TESIS DE LA UNIVERSIDAD DE ZARAGOZA

2014

79

José Ignacio Requeno Jarabo

Formal methods applied to the analysis of phylogenies: Phylogenetic model checking

Departamento
Informática e Ingeniería de Sistemas

Director/es
Colom Piazuelo, José Manuel

ISSN 2254-7606

Prensas de la Universidad
Universidad Zaragoza

Universidad Zaragoza
1542

Tesis Doctoral

FORMAL METHODS APPLIED TO THE ANALYSIS OF PHYLOGENIES: PHYLOGENETIC MODEL CHECKING

Autor

José Ignacio Requeno Jarabo

Director/es

Colom Piazuelo, José Manuel

UNIVERSIDAD DE ZARAGOZA

Informática e Ingeniería de Sistemas

2014

Tesis Doctoral

Formal methods applied to the analysis of phylogenies: Phylogenetic Model Checking

Autor: José Ignacio Requeno
Director: José Manuel Colom
Departamento de Informática de Ingeniería de Sistemas
EINA, Universidad de Zaragoza

We are all ignorant, but not all ignore the same things

— Albert Einstein

Agradecimientos

Me gustaría recordar que esta travesía no habría llegado a buen puerto sin la colaboración de diversos factores. En primer lugar, tienen un hueco importante en mi memoria las dos becas de investigación (I3A [171-03] y DGA [B117/10]) y proyectos de investigación del Ministerio ([TIN2008-06582-C03-02] y [TIN2011-27479-C04-01]) que han financiado todos estos años de esfuerzo. En segundo lugar, pero no por ello menos importante, las relaciones socio-personales también han influido de manera especial. Como es de rigor en estos casos, hago una mención particular a la familia, al director de tesis, a los amigos y compañeros del laboratorio de becarios que han creído en mí y en esta tesis desde el principio, y me han apoyado en los momentos de dificultad. Como esta enumeración personal suele degenerar en una lista interminable de nombres, fuente de celos, peleas y riñas cariñosas sobre el orden de aparición y quién ha colaborado más en desarrollo de los acontecimientos, he optado por cortar por lo sano y evitar envidias. Por ello, a <nombre>[A-Za-z]+<\nombre> en particular, y a todo el resto de héroes anónimos en general, os agradezco que hayais estado ahí ayudándome y guiándome en los momentos de debilidad. ¡Gracias!

Resumen

Los árboles filogenéticos son abstracciones útiles para modelar y caracterizar la evolución de un conjunto de especies o poblaciones respecto del tiempo. La proposición, verificación y generalización de hipótesis sobre un árbol filogenético inferido juegan un papel importante en el estudio y comprensión de las relaciones evolutivas. Actualmente, uno de los principales objetivos científicos es extraer o descubrir los mensajes biológicos implícitos y las propiedades estructurales subyacentes en la filogenia. Por ejemplo, la integración de información genética en una filogenia ayuda al descubrimiento de genes conservados en todo o parte del árbol, la identificación de posiciones covariantes en el ADN o la estimación de las fechas de divergencia entre especies según la velocidad en la variación de su sequencia genética. Consecuentemente, los árboles ayudan a comprender el mecanismo que gobierna la deriva evolutiva.

Hoy en día, el amplio espectro de métodos y herramientas heterogéneas para el análisis de filogenias enturbia y dificulta su utilización, además del fuerte acoplamiento entre la especificación de propiedades y los algoritmos utilizados para su evaluación (principalmente scripts ad hoc). Este problema constituye el punto de arranque de esta tesis en la que se propone como solución la posibilidad de introducir un entorno formal de verificación de hipótesis que, de manera automática y modular, estudie la veracidad de dichas propiedades definidas en un lenguaje genérico e independiente (en una lógica formal asociada) sobre uno de los múltiples softwares preparados para ello.

La contribución principal de la tesis es la propuesta de un marco formal para la descripción, verificación y manipulación de relaciones causales entre especies de forma independiente del código utilizado para su valoración. Para ello, se exploran las características de las técnicas de model checking, un paradigma en el que una especificación (propiedad) se formula en términos de una lógica temporal y se verifica su cumplimiento con respecto de una implementación representada mediante un modelo discreto definido como un sistema de transiciones. Se ha aplicado satisfactoriamente en la industria para el modelado de sistemas y su verificación, emergiendo del ámbito de las ciencias de la computación. En esta tesis se considerará model checking como una técnica formal, genérica y automática que, dado un modelo finito de estados de un sistema y una propiedad, sistemáticamente comprueba si dicha propiedad se satisface para un estado dado del modelo. Este proceso de verificación se realiza en tres fases: a) el modelado del sistema y las propiedades con un lenguaje descriptivo adecuado, b) la ejecución de la verificación (comprobando la validez de la propiedad con un software de model checking), y c) el análisis de los resultados (estudiando los contraejemplos si la propiedad falla).

Las principales ventajas y aportaciones del estudio de propiedades filogenéticas con este enfoque son que: a) se pueden considerar diferentes filogenias, b) se pueden

especificar propiedades complejas como la composición lógica de las demás, y c) el refinamiento de propiedades inválidas (así como el descubrimiento de otras nuevas) puede llevarse a cabo mediante la explotación de los contraejemplos.

Los métodos formales presentados aquí ofrecen un marco integrado para la verificación de propiedades filogenéticas utilizando un razonamiento simbólico y abstracto. Además, la introducción de las lógicas temporales añade independencia y modularidad a la definición de especificaciones biológicas con respecto a la formalización del modelo (filogenia): la misma propiedad puede ser exportada y evaluada en otras filogenias fácilmente. Por otra parte, la especificación con lógicas temporales es transparente a la tecnología subyacente de la herramienta genérica de model checking. Incluso, permite la reutilización del software existente de verificación, ahorrando costes de implementación y aprovechando la experiencia y tecnologías actuales.

Desde el punto de vista conceptual, a lo largo de esta tesis se desarrollan los puentes entre los términos biológicos que intentamos comprobar sobre una filogenia, junto con su especificación, interpretación y metodología de uso en el ámbito de model checking y las lógicas formales. El ámbito de aplicación abarca desde las propiedades topológicas o de secuencia clásicas (p.ej., la detección de clados o subárboles en función de polimorfismos comunes en las secuencias), hasta el estudio de relaciones cuantitativas y probabilistas más complejas (p.ej., la distribución de enfermedades endémicas en función de las regiones del árbol o la puntuación de filogenias con máxima verosimilitud). Dadas las características especiales del análisis de filogenias (principalmente su tamaño y la cantidad de datos etiquetados en las poblaciones), ha sido necesario adaptar la estrategia clásica de análisis y ajustar partes de las herramientas actuales para obtener rendimientos razonables.

Otro de los pilares esenciales de esta tesis es la utilización, en la mayor medida posible, de las herramientas existentes de model checking. Dichas herramientas se han desarrollado profusamente en los últimos 20 años en el contexto de "computer engineering" e "ingenieria del software" y cuentan con una amplia trayectoria acumulando experiencia y productos probados. Ejemplos paradigmáticos de esto son herramientas como NuSMV, de libre difusión, o SPIN, que incluso cuenta con un congreso dedicado que ha celebrado 21 ediciones hasta la fecha. La razón de adoptar esta estrategia es la de aprovechar toda esta experiencia previa y desarrollos consolidados en beneficio de lo que consideramos un nuevo domino de aplicación como es el análisis filogenético, incorporando a este dominio herramientas genéricas, generales y consolidadas que permiten la rápida adopción de la metodología. Esta tesis se propugna como sustitución de la metodología actual, que resulta más vertical, heterogénea y dispersa en cuanto a las habilidades requeridas en los filogenetistas para sus nuevas propuestas de análisis.

Se pondrá en evidencia en la tesis que las características especiales de este do-

minio, sobre todo las dimensiones de la información asociada a los nodos del árbol, requieren ciertas adaptaciones de estas herramientas. Dichas modificaciones no se refieren tanto a lo que es el kernel de la herramientas, que siguen siendo los mismos model checkers que existen hoy en dia, como a la organización de los workflows para la verificación de una propiedad. En este sentido, en esta tesis se analizarán dos estrategias fundamentales: a) la distribución de la carga de trabajo mediante diferentes estrategias de particionado del modelo entre réplicas coordinadas del mismo model checker básico, y b) la incorporación de sistemas de información que almacenen los datos fuera de la memoria principal para acelerar los procesos de verificación.

Las propuestas de la tesis están contrastadas por la comunidad científica mediante la siguientes publicaciones en conferencias y revistas internacionales. La introducción de model checking como entorno formal para analizar propiedades biológicas ha llevado a la publicación de nuestro primer artículo de congreso [26]. En [138], desarrollamos la verificación de hipótesis filogenéticas sobre un árbol de ejemplo construido a partir de las relaciones impuestas por un conjunto de proteínas codificadas por el ADN mitocondrial humano (ADNmt). En ese ejemplo, usamos una herramienta automática y genérica de model checking. El artículo de revista [143] resume lo básico de los artículos de congreso previos y extiende la aplicación de lógicas temporales a propiedades filogenéticas no consideradas hasta ahora. Los artículos citados aquí engloban los contenidos presentados en las Parte I–II de la tesis.

El enorme tamaño de los árboles y la considerable cantidad de información asociada a los estados (p.ej., la cadena de ADN) obligan a la introducción de adaptaciones especiales en las herramientas de model checking para mantener un rendimiento razonable en la verificación de propiedades y aliviar también el problema de la explosión de estados. El artículo de congreso [139] presenta las ventajas de rebanar el ADN asociado a los estados, la partición de la filogenia en pequeños subárboles y su distribución entre varias máquinas. Además, la idea original del model checking rebanado se complementa con la inclusión de una base de datos externa para el almacenamiento de secuencias. El artículo de revista [140] reúne las nociones introducidas en [139] junto con la implementación y resultados preliminares presentados [141]. Este tema se corresponde con lo presentado en la Parte III de la tesis.

Para terminar, la tesis reaprovecha las extensiones de las lógicas temporales con tiempo explícito y probabilidades a fin de manipular e interrogar al árbol sobre información cuantitativa. El artículo de congreso [142] ejemplifica la necesidad de introducir probabilidades y tiempo discreto para el análisis filogenético de un fenotipo real, en este caso, el ratio de distribución de la intolerancia a la lactosa entre diversas poblaciones arraigadas en las hojas de la filogenia. Esto se corres-

ponde con el Capítulo 14, que queda englobado dentro de las Partes IV–V. Las Partes IV–V completan los conceptos presentados en ese artículo de conferencia hacia otros dominios de aplicación, como la puntuación de árboles, y tiempo continuo. La introducción de parámetros en las hipótesis filogenéticas se plantea como trabajo futuro.

La propuesta metodológica realizada en esta tesis es completamente original, no existiendo ningún precedente al trabajo aquí presentado dentro del campo de la filogenia. En ese sentido, consideramos que hay una labor experimental a realizar junto a la comunidad de filogenetistas que permita recoger experiencias de uso, métodos de trabajo, y propiedades relevantes que permitan adaptar las metodologías aquí propuestas al entorno final para el que están pensadas.

Contents

List of Figures

List of Tables

Part I

Introduction and Formal Definition of the Framework

Chapter 1

Introduction

"You are to bring into the ark two of all living creatures, male and female, to keep them alive with you. Two of every kind of bird, of every kind of animal and of every kind of creature that moves along the ground will come to you to be kept alive. You are to take every kind of food that is to be eaten and store it away as food for you and for them." Noah did everything just as God commanded him.

Genesis 6:19-22

1.1. Biological Context: Phylogenies and Phylogenetic Analysis

The study and exploration of life is, for reasons as evident as they are difficult to formulate, one of the most deeply rooted constants in human thought. Understanding and classification are intimately related, and consequently it should come as no surprise that the branches of science that concern themselves with biological classification figure prominently in the history of modern science, from the revolutionary breakthroughs of Linnaeus and Darwin to the newer disciplines that have evolved in their wake.

Generally, the goal of taxonomy, into which Linnaeus breathed new life in the 18th century [113, 114, 115, 116], is to define and organize populations of organisms as taxonomic units (taxa) in hierarchies according to shared traits; in the domain of life, these may correlate in varying degrees to relations of descent. Additionally, systematics is informed by Darwin's 19th-century defense of common ancestor [54], by now universally accepted by the scientific community, and therefore seeks to propose phylogenies that reflect said existent descent relationships. Regardless

of their finer points, an overwhelming majority of taxonomic and systematics approaches commonly agree in their basic conception of evolution and classifications as hierarchical trees, whether their splits represent speciation events (mutations or sexual reproduction) and groups are founded on relevant attributes.

Phylogenetic trees, or phylogenies for short, continue to be useful abstractions for modeling and characterizing evolution over time [4]. Biologists consider them as feasible, traditional and wide-spread approaches of more powerful data structures such as phylogenetic networks or pedigrees. It is accepted that neither inheritance is single and infallible, nor descent strictly linear, with sexual reproduction and horizontal transfers of genetic material blurring this ideal model to some extent. Yet the lines are clearly visible, and the tree model is perfectly valid for vast regions of the postulated tree of life, if only we shift our focus from individuals to populations and beyond [18]. Biologists use phylogenetic trees in many different ways to solve both scientific and practical problems. For example, they use trees to make predictions about fossils or poorly-studied species, and to learn about the evolution of complex features, the order of evolution or the evolution of diversity [167].

During the last century, systematics and taxonomy have provided a wide range of methodologies, both theoretical and technical, which have contributed to a better understanding of evolution and the building of the tree of life. Among the former, Hennig's 20th-century cladistics [89] has proven a solid and durable theoretical methodology, aided in recent times by the development of computational phylogenetics. In addition, the concept of evolutionary distance allows for exploring tangible numerical relationships between sets of populations or individuals characterized by biological features such as DNA.

Computer science tools have upgraded the capabilities of biologists for the tree construction. A great amount of software packages for tree-building guides the power of computer science towards distance-based [148] (PHYLYP [70]) and character-based methods [74] (PAUP [161]) or maximum likelihood estimations [69, 71, 176] (RaXML [155], jModelTest [53]). In particular, the maximum likelihood estimations contribute with statistically consistent techniques for inferring the probability that the phylogeny accommodates a specific evolution drift, provided by the scientists in terms of DNA substitution models [117]. On these days, one of the most relevant challenges orbits around the validation of the inferred tree.

Both disciplines, taxonomy and systematics, make heavy use of empirical data, with proposition, verification and generalization of hypotheses over the reconstructed tree playing a central role in their application [130]. Once trees are inferred, the objective is to extract or to discover the implicit biological messages and structural properties underlying in the phylogeny under inspection. For in-

stance, the integration of genetic information in a phylogeny helps to the discovery of gene conservations, covariations or speciation dates and consequently the comprehension of the mechanisms that rule the evolution drift [75]. Nowadays, more and more applications rely on the existence of a support phylogenetic tree for the confirmation of biological hypothesis that are valuable for the scientific community. A small but representative portion of these researches combine phylogenetic trees (constructed via the mentioned tools using the information of the genome) with geographical or phenotypical data in order to trace the human migrations or the distribution of endemic diseases [92, 36]. Sometimes, these biological hypotheses also use the cladistic information[1] for studying the presence of the features in the state surroundings, i.e., inspecting the temporal locality (e.g., the conservation and covariation of sequences in zones of a tree [77]). The wide range of heterogeneous methods and tools used by biologists for the analysis of phylogenies and verification tasks recommends the possibility of researching in a generic framework for heterogeneous hypothesis testing over trees.

This suggests the potential of introducing formal logics for hypothesis verification by automatized means and, additionally, the symbolic manipulation of sets of phylogenetic data characterized by properties defined in these logics. The two former aspects are based on a) a methodology for a systematic construction of formal queries about the phylogenetic tree, and b) the availability of a general tool to answer the questions. The aim of this thesis is to propose a formal framework for describing, verifying and manipulating causal relationships of species, arranged as states, irrespective of the final structure (tree or network). The objective is the utilization of a phylogenetic tree for testing biological hypothesis and, indirectly, the use of the evaluation results to feedback the phylogeny and increase its quality. We try to overcome the lack of flexibility of conventional structural models for incorporating, combining and reusing evolutive rules at various levels of abstraction. To this end, we explore the features of model checking, a paradigm stemming from computer science based on temporal logics which has been successfully applied in industry for system modeling and verification [85].

1.2. Formal Verification Techniques: A Guided Tour

Traditionally, the formal methods research community has long advocated that by verifying a model of a complex system against a set of a priori specified properties we can gain greater assurance that the system behaves as desired. The process

[1] Cladistics consists of a biological classification of organisms that are grouped together based on their unique attributes.

of formalization does not give us any absolute guarantees about correctness, but it greatly increases our understanding of the system often by revealing inconsistencies, ambiguities, and incompletenesses in its design. In case of expensive, critical or safety systems, checking the system design and implementation against its model specifications using formal methods provides an enough confidence that the system won't crash or malfunction in a dangerous way after its construction.

Formal verification techniques are very familiar in the context of engineering disciplines: software engineering, protocol engineering, mechanical engineering and so on. In these fields, in the points related with the computer science, there are powerful developments and tools to support formal verification. Model checking is an automated and generic formal verification technique that, given a finite state model of a system and a formal property, systematically checks whether this property holds for (a given state in) that model. The model checking process consists of three phases: modeling both the system and properties with appropriate description languages, running the verification (checking the property validity with a model checking software) and analyzing the results (studying counterexamples if the property fails).

Bearing in mind that "any verification using model-based techniques is only as good as the model of the system", one of the most interesting strengths of model checking is the fact that "it is a potential push-button technology" because its use "requires neither a high degree of user interaction nor a high degree of expertise" [13]. Besides, there already exists a pool of generic and powerful software tools for the automated verification of properties [85]. They facilitate the decoupling of the verification process from the definition of properties and they mask the underlying implementation technology as well. In fact, this is an important quality that avoids the coding of particular verification algorithms and promotes the reuse and adaptation of tools with a long trajectory and community support.

The application of model checking techniques assumes that a system is abstracted by a set of discrete reachable states plus the transitions that allow the movement from one state to another. The objective is to place an intuitive abstraction for characterizing phylogenetic trees as transition systems, as well as specifying biological properties using temporal formulas, which capture the temporal nature of the phylogenetic tree. Implicitly, the speciation and mutation events along a path embed the notion of time. Occasionally, these temporal restrictions do not match with a chronological time but with a virtual clock that counts the number of transitions between states.

Evolutionary systems are very different in their structure and features from the abstractions of prototypes that are typically modeled for verification in industrial environments. One important difference between phylogenies and artificial systems designed by state-transition models is that we cannot arbitrarily change or modify

the phylogenetic structure according to our desire irrespective of the available information of the species (mainly the restrictions placed by the relation of their genomic sequences). The genome and the set of specifications that we evaluate over the tree report meaningful data that favor the continuous update and refinement of the tree topology.

Therefore the model checking approach that we propose for the context of phylogeny considers a phylogenetic tree as a *model* of a particular system endowing an evolutionary process in which biological properties expressed with formal logics can be *verified*. That is, a property is considered as a claim over the tree and then the verification determines the truth value for this claim. The evaluation of biological specifications validate, nullify or refine the phylogeny as a plausible model of the evolution.

Finally, the term of *model checking* from formal methods in computer science is often confused with the classic concept of *validation* in the field of phylogenetic analysis. In the biological context, validation focuses on determining the goodness of fit of a phylogenetic tree with respect to a DNA substitution model that hypothetically drives the evolution (see Section 15.2–15.3). Phylogenetists conceive the phylogenetic tree as a trace resulting from the long-term simulation of one of these mutation models, and then they score the fitness of the phylogeny accordingly. The misunderstood around the term *model checking* is probably motivated by the fact that DNA substitution models are commonly referred to as *phylogenetic models* (understood as a model that generates a phylogeny as output), while for us the phylogeny is considered as our model of the evolution over which we validate biological hypothesis and, incrementally, use the provided feedback to refine and update the tree. In fact, these two apparently contradictory concepts, *model checking* and *tree validation*, are complementary ideas. As we will see in further chapters, the application of formal methods in computer science for the tree validation (i.e., score the tree structure with respect to a DNA mutation model) is only a portion of all the potential of model checking techniques presented in this thesis. We will show how this validation can be integrated in and also be carried out by means of the proposed model checking techniques.

1.3. Related Work

The main significance of this thesis is in introducing the model testing framework to phylogenetics and possibly other computational biology communities. The importation of model checking techniques for the analysis of phylogenies is a conceptual and technological novelty that hasn't been introduced in this field before. Here, we present a whole different set of techniques to the computational phylogenetics community. Model checking offers a generic and formal framework for

evaluating phylogenetic properties. The wide range of heterogeneous methods and tools used by biologists for the analysis of phylogenies and verification tasks recommends the possibility of researching in a generic framework for heterogeneous hypothesis testing over trees: we try to offer uniformity.

Commonly, the study of phylogenetic relations starts with the DNA sequencing of the taxa, a process that is not completely free of errors [14, 10] and may disrupt the final result. Evolution causes differences in the DNA length between species [81, 82], even anomalous insertions/deletions (indels) of particular nucleic bases inside populations [133]. Thus, the alignment of genomes becomes necessary for a correct comparison of compatible DNA zones among individuals. Tools such as Clustalw [162] or Muscle [62] fill this gap, preparing the data for the reconstruction of phylogenetic trees. Nowadays, the construction of phylogenies is a key problem. Current phylogenetic inference methods comprise distances [148] (PHYLYP [70]), maximum parsimony [74] (PAUP [161]), maximum likelihood [69] (RaXML [155], jModelTest [53]) or Bayesian techniques [175] (MrBayes [147]), each one assessing the similarity of sequences with a complementary perspective and up-to-date computer software. Closely related taxa in the phylogeny are supposed to share a common structure and characters in the genome.

However after the inference of the tree, the phylogenetic analysis is carried out by disparate tools. Ape [67] and Arlequin [68] are R packages that help to the manipulation of phylogenies and apply statistical tests for genetics analysis in inter or intra-populations. For instance, BEAST [61] uses a Bayesian analysis of molecular sequences "for testing evolutionary hypotheses without conditioning on a single tree topology", but it is mainly limited to the estimation of the divergence dates and molecular substitution models. The study of comparative data (variation and distribution of characters, consensus trees, branch lengths, ...) is solved by Mesquite [120] or TNT [80]. In addition, adapted libraries for python such as Pycogent [105] includes functionalities for analyzing the correlation of nucleic bases and aminoacids in the sequences.

Beyond the mentioned software, the lack of generic tools for a true hypothesis testing involves the application of a separated statistical analysis for the set of DNA sequences and, subsequently, the interpretation over a phylogenetic tree. The process consists of the inspection of the tree topology and how the characters are arranged over there. A clear example of this is the search for correlated positions and compensations in the DNA, a study that employs a background phylogeny for discriminating noisy and true relations [77]. The general problem is not constrained to this peculiar example, but can be extended to the definition of complex relations between tree states and characters. Common questions in phylogeny are concerned about, for instance, the detection of reversions (back mutations) along a path, the extraction of the last common ancestor having a certain base shared by a set

of leaves, the aggregation of species in subtrees (clades) based on characteristic nucleotides, and even the detection of spurious or intruders in those clades. The verification of sound properties or the presence of anomalous patterns of nucleotic placement supposes a quality measure that warns about DNA sequencing errors or inconsistencies in the phylogenetic inference process.

Similarly, recent studies in phylogeography conjecture about human migrations [76, 36, 101, 166] (sometimes caused by the colonization with army invaders [179]) and its consequences in the language evolution [37, 29, 95], distribution of chronic and endemic diseases [57, 51, 129], and phenotypic adaptations (Tibetan people [19, 20, 25, 177], lactose [160, 163, 97, 92] and alcohol tolerance [72, 178]). Normally, these genetic modifications are motivated by environmental (temperature, oxygen) or cultural factors (milk-based diet, alcohol exposure) and they define haplotypes and populations as well.

The introduction of the model checking framework for hypothesis verification aims to avoid the necessity of coding ad hoc scripts (BioPerl [153], BioPython [49]) for particular investigations composing information of different areas. Nevertheless, the developments of model checking as a framework for learning phylogenetic properties have not been fully translated to the world of biology; or at least, many things must still to be done. Model checking has not been introduced in the context of phylogenetic analysis yet. Exclusively in computational biology, model checking has been applied to fields in which (mostly) quantitative properties over temporal data are analyzed [16]. Cellular and molecular interaction networks [128], including cell cycle kinetics [145], are the main areas of application, but problems as varied as the prediction of protein folding dynamics have been considered [107]. Further extensions involves the study of probabilistic relations in signaling pathway regulation [123], biological pathways [103, 104] or cell energy reactions [73]. Notice that temporal logics embedding the concept of evolutionary distance under the notion of time in phylogenetic trees have already been proposed in biological system modeling [56]. The existence of temporal logics capable of handling explicit time and probabilities enlarges the type and complexity of phylogenetic properties that the framework will process.

However, memory usage arises as a major limiting factor in the analysis of complex systems. Often, it is in association with vast state spaces, and therefore with long execution times. In this regard, the conventional monolithic techniques conceived for improving the performance in industrial model checking applications fail to manipulate the phylogenetic tree in an efficient way due to the inherent features of the biological data, mainly the huge ratio of labels per state (i.e., DNA sequence). This problem persists in spite of the many methods which have been devised to scale model checking procedures. Of these, symbolic model checking is perhaps the most widespread [99, 85]. There exist several general-purpose memory

techniques that can be applied to alleviate the problem of memory footprint, such as abstractions, partial reductions and symmetries (for a review, see e.g. [63, 31]).

Current efforts in this area revolve around two main topics. Firstly, compositional reasoning [47], itself a classic approach based on the verification of local properties associated to each of a collection of "components" (e.g., codons and genes in the context of biological sequences), proceeding incrementally to infer global properties of the system through a bottom-up strategy. Chief in importance among these techniques is the assume-guarantee paradigm [135], which operates by establishing a collection of assumptions about the environment of a component and verifying the latter subject to the former. Alternative approaches are exemplified by [78], where they focus the trouble from the point of view of temporal logic formula decomposition. However, all these methods are ineffective when applied automatically in isolation to tightly coupled systems made up from highly interdependent components [48]. Only a few theoretical works have been published in relation to the complexity and advantages of compositional model checking over these simple structures [39].

Secondly, we have methods that exploit the explosive availability of multicore (shared-memory) computers [96], fast interconnection networks [165] or MapReduce in clusters [22]. Generally, these operate by partitioning the system as defined by the Kripke structure and distributing the chunks among available computing units (both storage of the partial Kripke structure and computation of satisfiability of logic formulas [84, 30]). These would be applicable here, yet by themselves they are ineffective, as they address the size of the structure in number of states and not the complexity of each state, which is the other limiting factor in phylogenetic model checking. An approximation to this last group of methods is considered in Section 7.3. The adaptation and optimization of model checking techniques for large systems to the domain of phylogenetic analysis constitutes our second contribution.

1.4. Objectives of this PhD Thesis

The aim of this thesis is the introduction of formal methods techniques as an unifying formalism that permits the phylogeneticist to focus their efforts on phylogenetic analysis instead of focusing on implementation issues concerned with particular algorithms or tools. The symbolic manipulation of phylogenies and phylogenetic properties, represented in terms of temporal logics, helps with the extraction of meaningful biological information. The formalism is flexible enough to consider different kind of data structures and evolutionary models. It also represents an opportunity to handle the increasing complexity of the structural properties required by biologists, e.g. the detection of potentially deleterious mutations

or conserved regions in a clade. Moreover, the idea of hypothesis verification works in a closed-loop, helping the phylogeneticist to *refine* or *discover* properties using counterexamples obtained from unfulfilled properties. To this end, the introduction of model checking techniques is summarized under three different perspectives:

1. First of all, our goal is to separate the implementation (software tools) from the analysis of a phylogeny, denoted with a set of properties the user desires to check. The objective is to develop a different kind of phylogenetic analysis. Inside a model checking framework, the analysis and study of properties is realized independently of the software tool that makes the particular computations. That is, the model checker tool is a generic package that interprets the specification of a system and automatically realizes the computations to carry out the analysis. Then, the biologists can concentrate their activity in the specification and interpretation of the results. In addition, the model and specifications are usually defined as separated input files for the model checking tool, allowing the verification of the same properties over multiple trees and the interpretation of different results or behaviors.

   This is a very instrumental reason, and it sometimes presents a conflict between generality and efficiency. Nevertheless, there are many developments inside the formal verification domain that are transparent to the application domain and that can be translated to the context of phylogeny. Therefore, there is no need to return to the path that the computer scientists already visited before. For example, the research and implementation of methods for the distribution of computations in order to speed up the performance, or the extraction of efficient representations and symbolic manipulation of the information. The use of experienced software with up-to-date technologies and a good community support such as NuSMV [45] facilitates the study of phylogenetic properties. Our objective is to take advantage of these tools and adapt them to the domain of phylogenetics if necessary, instead of coding a completely new software or focusing on particular algorithms.

   On the other hand, current packages for phylogenetic analysis such as Mesquite [120], ape [67] or Pycogent [105] are becoming really complex with thousands of code lines. They are difficult to maintain, even when they include specific and adapted algorithms for this field. The attractiveness of model checking comes from the fact that it is completely automatic, generic and independent of the model and specifications (i.e., the learning curve for a user is very gentle).

2. The second cornerstone of our approach is the introduction of the symbolic manipulation of biological objects used to reason inside the world of phylogeny. In this sense, the introduction of formal definitions through logical

formulas, interpreted over the phylogenetic tree, allows to characterize sets that are not explicitly enumerated. These sets of objects are symbolically managed and it is possible to define specific calculus to manipulate them.

In a complex world with many levels of abstraction, such as the phylogeny, the ability to systematically manipulate components or objects represented in a symbolic way constitutes a key advantage. This requires some abstraction tool allowing reasoning with the right objects at each level, removing unnecessary details for the searched results. For example if we are trying to work in the field of cladistics, then one of the right objects to reason with is the set of clades. Hence, we need to have a symbolic representation of these objects and a calculus to reason with them.

3. Last but not least, we propose model checking for studying complex systems with single or multiple phylogenetic trees as input. Our objective is to obtain and to organize the information contained in these structural-logical models. Here we try to extract properties of a phylogeny, combine them, compare the results among trees and split the properties into smaller meaningful ones. In other words, we need an algebra to manipulate properties symbolically. This is a difference with respect to the classical verification framework, which restricts the analysis to the verification if a property is satisfied or not. We propose a classification of phylogenetic properties and a methodology for the specification of these hypothesis using temporal logics.

   In addition, our phylogenetic model is initially represented by a phylogenetic tree labeled with the genome of each taxon, but the states of the phylogeny can be also enriched with extra information complementary to that obtained from the DNA sequences or the output results obtained from the evaluation of previous properties, for example, with phenotypic information. Therefore, the model checking techniques work with properties that can be interpreted over different models (phylogenetic trees) or over phylogenies that maintains the structure but the information content is changed. We also work with models that include probabilities (Markov chains). Modifying the behavior of the logics and data structures, we can work in the future with phylogenetic networks or pedigrees as well.

To sum up, this approach allows the specification of properties in temporal logic in an unambiguous and mathematically strict way. Hence, biologists can formalize consensus definitions for controversial phylogenetic terms, mainly in quantitative analysis. The definition and analysis of a new phylogenetic property doesn't need the implementation of any new algorithm, which reduces the risk of coding errors and simplifies the software cycle of life. The same property can be exported to and tested in different models (phylogenetic trees) with minor changes, fixing them

for phylogenetic networks if necessary. It also offers counterexamples in case of a model fails to satisfy a property serving as indispensable for the debugging.

The previous arguments correspond to the conceptual advantages of model checking beyond the computational efficiency. In fact, we try to demonstrate that our contribution consists of the adaptation of the model checking framework as an unifying formalism to realize many different tasks under the same generic formal description languages and tools, offering the possibility to create different abstraction levels throughout the symbolic creation and management of objects.

The work introduced in the first theoretical sections is completed with a second part oriented to the usability and competitiveness of our approach. We have evaluated the performance of existing model checking tools in the field of phylogenetics. Due to the particularities of the biological context and the huge volume of data they have to manipulate, we have discovered that most of the software packages suffer from slowness. Then, we are forced to design, adapt, propose and implement new solutions for increasing the efficiency of our methodology in phylogenetics. Finally, the formalism is enriched with time and probabilities for model checking.

1.5. Organization and List of Publications

The contents presented in this thesis is supported by several publications in international conferences and journals. The introduction of model checking as a formal framework for analyzing biological properties led us to our first seminal paper [26]. In [138], we develop the verification of a particular phylogenetic example with proteins coded by human mitochondrial DNA (mtDNA). There, we used a generic and automated model checking tool. The journal paper [143] summarizes the basics of previous papers and extends the operational of temporal logics to new phylogenetic properties unconsidered until now.

Due to the peculiarities of the phylogenetic analysis (i.e., mainly huge trees with a considerable amount of information in the states, such as the DNA), the model checking tools need to be optimized in order to keep a reasonable performance and ease the state explosion problem. The paper [139] presents the advantages of slicing the DNA strings associated to the states, and the partition of the phylogeny into small subtrees. Besides, the original idea of sliced model checking is complemented with the addition of an external database for storing the sequences. The journal paper [140] gathers the notions introduced in [139] together with the implementation and preliminary results of [141].

At the end, the thesis extends the standard temporal logics with explicit time and probabilities so as to manipulate and interrogate the tree with quantitative

information. The paper [142] exemplifies the necessity of introducing probabilities and discrete time for the phylogenetic analysis of a real phenotype.

Therefore, the thesis is arranged in six parts. Following this introduction, Part I summarizes the essentials of phylogenetics and model checking. It explains the roots which bridge model checking and phylogenetic analysis: phylogenies as logical models, phylogenetic specifications as temporal logic formulas, and automated system verification via model checking tools. Later, Part II describes the logical specification of non-trivial structural properties of cladistic classification and sequence composition. It also details the key steps for implementing phylogenetic trees and biological properties within the scope of a Symbolic Model Verifier (SMV) tool in order to obtain both feasibility and performance criteria for our approach. Part III considers different compatible optimizations for scaling the framework for bigger systems, mainly the introduction of tree partitions, state slicing and external databases. A workflow describes the interconnection and cooperation of the modules implementing the mentioned techniques. Next, the logic and structures presented in Part II are extended for quantitative purposes (explicit time and probabilities) in Part IV–V. With a set of real examples, we motivate the interest of inserting explicit clocks and likelihoods in phylogenies. Finally, Part VI gathers the conclusions drawn from this research and outlines the future work. We highlight that the introduction of parameters in models and specifications allows the symbolic manipulation of relations, converging to a potential automatic discovery of properties and model exploration.

Chapter 2

Bridging Worlds: Phylogenetics and Model Checking

> *The fact is, what we're doing could be construed as, forgive me sir, collaboration with the enemy. Perhaps even as treasonable activity... Must we work so well. Must we build them a better bridge than they could have built for themselves?*
>
> — Maj. Clipton, *The Bridge on the River Kwai*

2.1. Introduction

Transition systems are considered powerful formal models for the study of concurrent systems [13, Definition 2.1]. They are machines composed of a set of (probably) infinite states labeled with information describing the system behavior at that point, and a set transitions among them. Formally, a state transition system is a pair (S, R) where S is a set of states and $R \subseteq S \times S$ is a binary relation (transitions) over S. Given a model of a system and a suitable logic to reason about its behavior over time, it is possible to achieve automated, exhaustive verification of properties of interest. In this section we show that phylogenies can be understood and represented by such models.

Phylogenetics and model checking can be bridged after reflecting on some considerations about the processes of modeling and specification. We begin by identifying the foundations of evolution with those of transition systems and continue

with the study of their temporal nature and their logical formulation. We conclude with an overview of verification under the model checking paradigm.

2.2. Evolution as a Transition System

At the highest level of abstraction, phylogenies postulate partial models of the evolution history of sets of living organisms. They can be represented as directed graphs, though phylogenetic trees are widespread representations and they suffice for most common purposes; phylogenetic trees will be adopted in this thesis.

Rooted labeled trees offer a realistic model of aggregated evolution, in which each vertex represents an inferred state of the evolution characterized by a population of related individuals who mate among themselves and are denoted by a common, compatible heritage (e.g., biological sequences such as DNA) or other information such as their attributes (e.g., morphological and physiological data). The vertex are arranged along the paths of the tree according to a evolutive process. The methodology presented here is extensible to other domains (trees or networks) and phylogenies annotated with different kind of information. Although phylogenetic trees don't match exactly with a gene tree [131, 132, 119], we start focusing on trees built from genes and genomic alignments for simplicity.

Transformative events that modify heritable information give rise to new states which are reflected in oriented parent-child edges in the graph. Transitions are comparable to "instantaneous" speciation events or mutations. Note that neither explicit time nor the ordering of child states are part of the model, which is consistent with the descent semantics of the tree. In this context, time is implicitly represented as a relation order that indicates the direction of the evolution drift caused by the speciation events.

Definition 1 (Rooted Labeled Tree)**.** *Let Σ be a finite alphabet and l a natural number. A phylogenetic tree over Σ^l is a tuple $P = (T, r, D)$, where:*

- *$T = (V, E)$ is a tree graph,*
- *$r \in V$ is its root, and*
- *$D : V \to \Sigma^l$ is a dictionary function that labels each vertex with its associated taxon information, mainly the genome sequence.*

Gene trees are typically built from finite sets of words of uniform length resulting from alignment algorithms. These words, which commonly represent present-time taxa, are required to be in bijective correspondence with the set of leaves of the trees. The inclusion of extinct species, which are improbably direct ancestors of extant taxa, are also located in terminal nodes. Ancient DNA sequences far

before $100,000$ years are unusual in the phylogeny due to its deterioration, but they can be inferred for the internal nodes of the tree using maximum parsimony for the direct common ancestors.

Restrictions can be added or removed to adjust the phylogeny as needed (i.e., including horizontal transference, sexual reproduction or multiple roots for phylogenetic networks), but in any case the nature of trees as *reactive systems* should become clear by now. They are composed of independent states of evolution that interact indefinitely with their environment. One of the most prominent features is their *state*, i.e., their hereditary information or a suitable portion thereof. Consequently, it is possible to naturally effect modeling and verification of evolutionary systems, and to this end data structures for the representation of transition systems become a very attractive solution. A Kripke structure provides a graph-based data arrangement that represents the behavior of the system and provides semantic information for querying properties in temporal logics [122].

Definition 2 (Kripke Structure)**.** *Let AP be a set of* atomic propositions*, i.e., boolean predicates that describe the observable properties of a state. A Kripke structure over AP is a finite transition system represented by a tuple $M = (S, S_0, R, L, AP)$, where:*

- *S is a finite set of states,*
- *$S_0 \subseteq S$ is the set of initial states,*
- *$R \subseteq S \times S$ is a total transition relation between states, i.e., for every state $s_i \in S$, there exists $s_j \in S$ such that $(s_i, s_j) \in R$, and*
- *$L : S \to 2^{AP}$ is the labeling function that associates each state with the subset of atomic propositions that are true of it.*

A Kripke structure models a system that is capable of an infinite number of behaviors or *paths*, infinite sequences of successive states $\pi = s_0 s_1 s_2 \ldots$ such that $s_0 \in S_0$ and $(s_i, s_{i+1}) \in R, i \in \mathbb{N}$. The set of possible executions (paths) in a structure can be unfolded into its *computation tree*[1].

Here we focus on Kripke structures that represent a certain tree as a (hopefully real) computation of the evolutionary process, or rather the set of computations that result in the hypothesized patterns of evolution. Relations between states and atomic propositions, and between tree branches and state transitions, are of utmost in importance, and raise some interesting issues that need to be addressed:

[1] A computation tree is a rooted tree-like structure in which the future is not determined by the current node and, therefore, there are many different paths reachable from this state.

- Ideally, the state of an evolution process is uniquely identified by its biological information. Sequences also determine the atomic propositions that form the basis of logical properties: the presence of a certain alphabet symbol at a given position. If separate vertices of the tree must share a sequence, their states can be enriched with auxiliary properties to preserve the unique identity of each.

- Once a one-to-one correspondence between tree vertices and states has been established, it remains to resolve the identification of branches as state transitions. Transitions are comparable to "instantaneous" speciation events.

- In sum, trees are essentially a present-time, local, tentative snapshot in the execution of a potentially infinite evolution system. In order to translate trees to the infinite-path semantics of Kripke structures, we need to add self-loops to terminal vertices so as to deadlock their states.

At this point, we can define a suitable *branching-time structure* for phylogenetic trees for the interpretation of temporal logic formulas which express properties in them. The most common state formula determines whether the present state is associated with a sequence $seq = \sigma_1\sigma_2 \ldots \sigma_l \in \Sigma^l$ (or possibly a partial sequence or a set of sequences). Sequences will be manipulated symbolically, as will sets, as the aggregation of their parts. In logical terms:

$$seq \equiv \bigwedge_{i=1}^{l} seq\,[i] = \sigma_i \tag{2.1}$$

These expressions related with the DNA compose the basic set of atomic propositions (AP) that are tagged to the states, that is, $AP = \{seq\,[i] = \sigma \,|\, \sigma \in \Sigma, i \leq l\}$. The set of AP accepts the inclusion of new information apart from the genome.

Definition 3 (Branching-time Phylogeny)**.** *A tree (per Definition 1) $P = (T, r, D)$ is univocally defined by the Kripke structure $M = (V, \{r\}, R, L)$, where:*

- *R is the transition relation composed of the set of tree edges (directed from r) plus self-loops on the leaves: $R = E \cup \{(v, v) : \nexists\,(v, w) \in E \wedge v, w \in V\}$ with $v \neq w$.*

- *L is the standard labeling function defined by AP, under which a state v mapped to $D(v) = seq$ with $seq = \sigma_1\sigma_2 \ldots \sigma_l$ satisfies the family of properties $seq\,[i] = \sigma_i, 1 \leq i \leq l$, plus a unique state identifier in the case of several states sharing the same atomic propositions.*

Roughly speaking, we can say that the Kripke structure reflects the parent-child relations in the original phylogenetic tree representing the evolution process. Self-loops in terminal nodes allow for infinite computation paths in the original

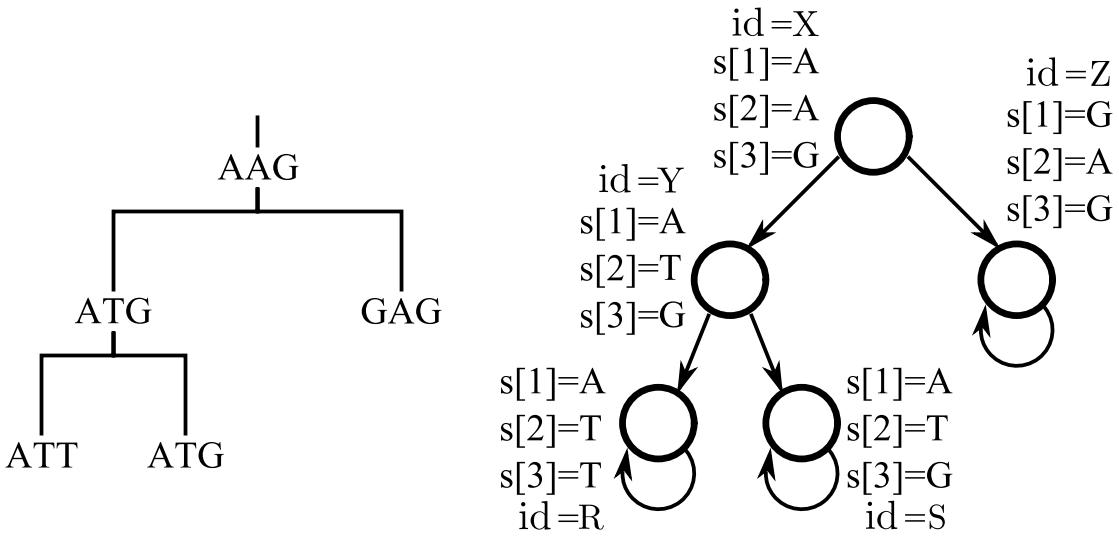

Figure 2.1: Translation from a phylogenetic tree to a Kripke structure.

phylogenetic tree. The labeling of states with atomic propositions in the transition system helps in the identification of nodes during the verification process. Thanks to the codification explained, the detection of a particular instance of a DNA sequence is translated to the selection of states satisfying a set of logical equalities with the form $seq\,[i] = \sigma_i, 1 \leq i \leq l$. In this way, we can manipulate sequences symbolically. In case of collision, a explicit definition of state identifiers ensures uniqueness. The set of atomic propositions has to be able to represent distinct populations that share a common sequence distinguished as different states.

Figure 2.1 illustrates this translation process from a (phylogenetic) rooted labeled tree (Definition 1) to a Kripke structure (Definition 2). It should be emphasized that our proposal can be used in future works for other phylogenetic structures such as phylogenetic networks [52] by reformulating Definitions 1–3 as required. Although they are more generic and powerful representations of the evolution process, the complexity for interpreting and manipulating specifications in temporal logic in a phylogenetic network places this data structure out of our scope in this thesis.

2.3. Temporal Logic as a Specification Language

Temporal logics are formal systems that allow the representation and manipulation of logical propositions qualified in terms of time [134, 122, 46, 63, 13]. In

the context of transition systems, they are used to define properties on sequences of transitions between states of a system through a convenient abstraction of it (in the present case, a specific kind of Kripke structure). For example, properties may express whether it is possible that one particular type of state may be reached at a particular point or whether a certain property will always hold.

Temporal logics can be classified according to how they treat sequences of events: whereas *linear-time logics* (LTL) [134] deal with individual paths $\pi = s_0 s_1 s_2 \ldots$, *branching-time logics* take into account the set of possible progressions from each state, hence reasoning globally about the computation tree. *Computational Tree Logic* (CTL) [46], a versatile exponent of the latter, has been widely adopted by the model checking community. Phylogenies represent evolutionary processes that are mainly branching in nature due to the (hypothetically independent) speciation events over the time. Branching-time logics in general, and CTL in particular, are remarkably well suited to the description of phylogenies because the structure of phylogenetic trees matches with the topology of a computation tree and CTL provides a set of operators for inspecting and manipulating those paths. We propose Phylogenetic Tree Logic (PTL) as a temporal logic close to CTL for the specification of evolutionary processes.

PTL reinterprets the quantifiers of first-order logic as *path quantifiers*, expressing the fulfillment of a property throughout all computation paths (**A**), or at least one computation path (**E**). These two must be immediately qualified by one of five *temporal operators*, of which three express the satisfaction of a property eventually in time (**F**), at all times (**G**), or in the next state (**X**); and two are conditional constructs in which a precedent is verified until a consequent comes into force (**U**), or until and including the moment when it does, if it does (**R**). A complete grammar and semantics of PTL formulas can be defined from a minimal and complete subset of logical operators (see Figure 2.2).

Definition 4 (Phylogenetic Tree Logic)**.** *A temporal logic formula ϕ is defined by the following grammar, where $p \in AP$:*

$$\phi ::= true \mid p \mid \neg\phi \mid \phi \vee \phi \mid \mathbf{EX}\,(\phi) \mid \mathbf{EG}\,(\phi) \mid \mathbf{E}\,[\phi \mathbf{U} \phi] \tag{2.2}$$

The formulas are checked against a structure M considering all paths π from a certain state s_0. Notice that $M, s_0 \vDash \phi$ means that s_0 satisfies ϕ. The semantics of well-formed formulas is as follows (let $\pi = s_0 s_1 s_2 \ldots$):

- $M, s_0 \vDash p \Leftrightarrow p \in L\,(s_0)$,
- $M, s_0 \vDash \neg\phi \Leftrightarrow M, s_0 \nvDash \phi$,
- $M, s_0 \vDash \phi \vee \psi \Leftrightarrow M, s_0 \vDash \phi$ *or* $M, s_0 \vDash \psi$,

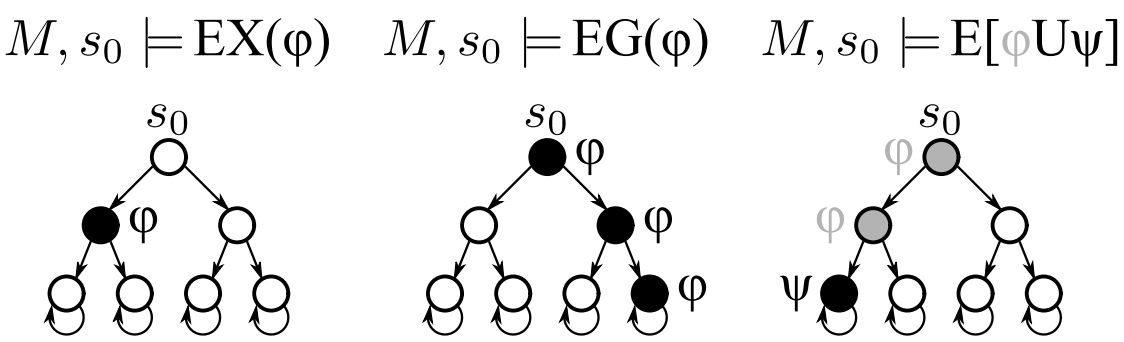

Figure 2.2: Evaluation of temporal logic operators.

- $M, s_0 \vDash \mathbf{EX}(\phi) \Leftrightarrow \exists \pi : M, s_1 \vDash \phi$,
- $M, s_0 \vDash \mathbf{EG}(\phi) \Leftrightarrow \exists \pi : M, s_i \vDash \phi, \forall i \in \mathbb{N}, i \geq 0$,
- $M, s_0 \vDash \mathbf{E}[\phi \mathbf{U} \psi] \Leftrightarrow \exists \pi, i \in \mathbb{N} : M, s_i \vDash \psi$ *and* $M, s_j \vDash \phi, 0 \leq j < i$.

The previous language is extensible with additional temporal logic operators that are calculated in terms of the previous ones:

- $\phi \wedge \psi \equiv \neg(\neg\phi \vee \neg\psi)$
- $\mathbf{A}(\psi) \equiv \neg\mathbf{E}(\neg\psi)$
- $\mathbf{F}(\psi) \equiv [true \mathbf{U} \psi]$
- $[\psi_1 \mathbf{R} \psi_2] \equiv \neg\,[\neg\psi_1 \mathbf{U} \neg\psi_2]$

A CTL formula ϕ represents a property that may be verified at certain states in the computation tree. In this context, a system M satisfies ϕ iff its initial state (phylogenetic root) does: $M, s_0 \vDash \phi$. We can extend the previous formula to a set of initial states: $\bigwedge_{s_0 \in S_0} M, s_0 \vDash \phi$.

A logic thus defined permits the formal expression of generic properties on evolving biological sequences in a form that is mathematically strict and compact, and their eventual automated verification without any additional programming involved. It can be used to verify hypotheses or desirable properties as well as to find out where in a phylogeny, if at all, such statements are true.

2.4. Model Checking as an Inference Framework

The operating principle behind model checking is simple: it is the execution of verification software in a computer to check the correctness of a system, given a

model of the system and a specification of its requirements, both provided by the user. In the event of failure to comply with the specification, the software outputs the scenarios which invalidate the property as counterexamples. Nevertheless, the state space explosion problem prevents the verification even for small systems, for which symbolic manipulation of systems and formulas are required [33]. The model checking technique is, thus, only limited by the complexity and expressiveness of the selected logic.

Most importantly, the use of model checking techniques is completely transparent to the system under verification, as they are domain and interpretation independent. This means that phylogeneticists can shift their efforts from implementation issues to logical modeling (establishing required or desirable properties before model checking) and results analysis (observing the success or failure of the process and ascertaining its significance after model checking). The main goal of introducing model checking in the field of phylogeny is the evaluation and inference of biological properties using the tree.

The feedback provided by the inquiry of hypothesis in the phylogeny helps to refine our knowledge about the evolution and, indirectly, update the phylogenetic structure. Particularly, the counterexamples are useful if the evaluation results are aberrant with those that are expected in the scientific community. For example, we can inspect tree regions that are supposed (by biological reasons) to keep an invariant portion of DNA. If we find a falsification in the conservation of this pattern, it rises an alert of DNA sequencing or phylogenetic reconstruction errors (for more details, see Section 4.3).

Going into greater detail, the key step of the verification process consists of the generation of a satisfiable set (i.e., the set of states of a Kripke structure for which a property holds). The algorithms of model checking are based on set theory. The definition and manipulation of sets is done symbolically in terms of a temporal logic. The evaluation process starts with the input of a set of initial states characterized by a formula in temporal logic and a reachability relation provided by the Kripke structure. The model checking algorithm computes reachable states from/to that set in an iterative and symbolic way until a greatest (least) fixed point is reached [63]. We arrive to a greatest (least) fixed point when the model checking algorithm cannot update the result set. The counterexamples are obtained by tracing back the appropriate path over the complementary set.

Take the following example. Our aim is the selection of a path whose nodes conserve the genome in the first position of the DNA string, for instance, $seq[1] = A$. In temporal logic notation, we try to solve $M, s_0 \vDash \mathbf{EG}(seq[1] = A)$, with s_0 the root of the phylogeny. For homogeneity, we suppose that the common ancestors have an inferred DNA. The greatest fixed point algorithm starts considering all the states of the Kripke structure, and, next, restricts the set to those nodes

where $seq[1] = A$ holds. The procedure reduces repeatedly the set of states: in each iteration, the set is limited to the elements that satisfy $seq[1] = A$ and that, according to the reachability relation of the phylogeny, can be reached by the nodes obtained in the previous step. The algorithm finishes when it cannot update the result set anymore. The property is true if the greatest fixed point algorithm incorporates the root s_0 in the result set. The least fixed point operates inversely, increasing the set in each operation.

Summarizing, there is a direct correspondence between resolving boolean formulas of temporal logic and manipulating sets of states. In fact, a temporal logic formula characterizes the states of the tree satisfying the imposed relations. The states of the tree are coded with the same variables used in the atomic propositions, and, therefore, the model checking algorithms can manipulate sets symbolically by means of temporal formulas.

Chapter 3

Conclusions

"Assume a spherical cow of uniform density."

—

3.1. Foundations of this Thesis

Phylogenies are useful structures for testing biological hypothesis. Nonetheless, the data on which trees are based is usually noisy, mainly for trees with exclusively genomic information [14, 10]. First of all, the DNA and protein sequencing in the laboratory, though highly automatized and accurate in recent ages, are not completely exempt of errors whose unfaithful signal propagates to further steps of phylogenetic analysis. In addition, ancestral DNA is barely stable and its degraded conservation prevents its use far beyond $100,000$ years, which limits the application of genomics to present-time taxa arranged in the tips of the phylogeny. Nevertheless, some methods offer an estimated value of the DNA for the internal nodes and draw the edge length proportional to the time (for instance, parsimony [74]).

Secondly, the large computational cost of aligning sequences and inferring the *true* phylogenetic tree with current software tools restricts the input to only a few simultaneous genes or characters per phylogeny instead of the complete genome. This fact amplifies the bias and distortions the complete image of the evolution drift in the DNA. Even, different methodologies for tree reconstruction provide candidates with sound background and same theoretical consistency but that are dissimilar to each other [159].

Hence, it is almost impossible to ensure with great certitude if the current phylogeny corresponds to the *true* tree of life because of the ambiguous, incomplete or partial information the scientist work with. Although the phylogenetic trees

obtained from single genes are useful for specific studies, a reliable phylogeny of species must solve theses biases using consensus trees, phylogenetic comparative methods and a compendium of information collected from disjoint research areas (e.g., animal migration or phenotype data) in order to enrich the tree model.

Therefore, there isn't a single phylogenetic tree but a summary of possible trees that are continuously revised and refined every time we gain a better comprehension of the evolution: the phylogeny is not a static model of the evolution. This fact leads to the periodic appearance of heterogeneous tools and methods for building trees or evaluating properties over them, which complicates the analysis of phylogenies. As demonstrated in the related work, the lack of generic tools for a true hypothesis testing involves the application of a separated statistical analysis for the set of DNA sequences and, subsequently, the interpretation over a phylogenetic tree. For more complex problems like phylogeography, the necessity of specific implementations and scripts is real.

In any case, the phylogenetic tree is still an acceptable model for characterizing the evolution process, which is captured by the states of a hypothetical transition system reflecting a specific step of the speciation process, and the transitions themselves, which describe mutations and reproduction events. The introduction of model checking techniques alleviates the problem of verifying multiple properties over trees. Model checking is an automated generic verification technique that, given a finite state model of a system and a formal property, systematically checks whether this property holds for (a given state in) that model or returns a counterexample if it is invalid. Making the corresponding associations, we have built a bridge between two such apparently removed worlds as phylogenetics and formal verification using model checking techniques. To this end, a phylogenetic tree needs to be reinterpreted as a transition system in order to assimilate it to the Kripke structure used by model checking techniques.

The use of sound and strict temporal logics adds formalism to the definition and verification of properties over those structures, as well as the classification of phylogenetic hypothesis according to the type of information they work with. It also facilitates the introduction of generic and automatic model checking tools, which decouple the specification and test of models and properties from the particular implementation language. In addition, the existence of good tools with a robust historical background and community support avoids the coding of algorithms *de novo*: the adaption of model checking tools to phylogenetic analysis revolves around the modification and customization of specific traits. The modularity (separation of the model from the specifications) simplifies the exportation and inspection of properties in other phylogenies and the application of partial results.

The inquiry of phylogenetic properties allows the discovery of new information.

Probably, they alert with counterexamples that report erroneous relations in the phylogeny. The result of testing phylogenetic properties expressed in temporal logics may confirm or refute biological hypothesis about the speciation drift, which assesses the tree quality with respect to different criteria. For example, rootedness is an important factor in practice and should not be taken for granted, even for simple phylogenetic trees. An overwhelming majority of phylogeny reconstruction methods, including all those in common use, produce as their results undirected trees, which lack a distinct root unless ascertained by external means, e.g. by using outroots.

To this end, model checking is useful as a method to locate the root and discriminate among several potential candidates. Given a set of properties, model checking will consist of detecting the set of potential roots that verify them. That is, consider a property to verify on an unrooted phylogeny. For instance, we want to inspect the roots whose mean distance to the leaves is over a certain threshold, i.e., is the tree balanced if we take that node as root? (see Chapter 12.2). It is possible to determine the set of nodes which can be the root of the tree according to that property by attempting to verify the property taking each of the nodes of the tree as its root. This involves the symbolic manipulation of sets of states by means of the propositions they satisfy. The total cost of verification is linear with respect to the number of potential roots, which nonetheless is composed of completely independent problems (though they may have subproblems in common, etc.). This can be parallelized with perfect speed up.

Each property in a collection restricts the possible locations of the root to a subset of the nodes of the tree. We may conclude that a node is consistent with the rootedness of the tree under all those properties if it is acceptable as a root for each and every one of the properties. The number and type of properties validated over the tree gives a way to judge the strength of the root under consideration. The feedback returned by the model checking process helps to update the phylogenetic structure.

In spite of the mentioned advantages for defining and evaluating phylogenetic hypothesis, the model checking techniques has never been introduced in the domain of phylogenetics yet. They are mainly employed for industrial domains and particular operations. Model checking is barely introduced in computational biology, where it has been applied to fields in which (mostly) quantitative properties over temporal data are analyzed. This fact contrasts with the principal limitations of phylogenetic analysis. The goals of introducing model checking techniques in phylogenetics have been a) the presentation of a formal framework for the investigation of biological hypothesis that solves the lack of independence between specification and verification, and b) the use of generic and powerful tools for the validation of those properties and the symbolic manipulation of states instead of

the codification of new software.

As final remarks, temporal logics assume a well-defined flow of time, and thus are dependent on an oriented graph. Kripke structures are oriented by definition, i.e., there exists a initial state at least. Branching-time temporal logics define properties over the computation tree of a Kripke structure. A rooted phylogenetic tree matches with the notion of a computation tree because the speciation events are similar to the transitions and the taxa correspond to the states of the computation tree. In case of unrooted phylogenies, it is necessary to either adapt a temporal logic to accept uncertainty in the flow of time, or the model checking algorithm. Clearly, the Kripke structure needs to be generalized. In any case, it is possible to overcome this problem by defining a partial orientation in regions where the evolution flow is obviously known.

In sum, the phylogenetic tree is still a feasible and useful model over which we inspect biological hypothesis despite the limitations and drawbacks it has. The inherent hierarchy of the data structure allows the adaptation and optimization of the model checking techniques to this domain. Besides, the use of a phylogeny as a model of the evolution contrasts with the use of DNA substitution models, which are considered as the default models for analyzing the principles operating behind the macro evolution process. More complex and refined data structures like phylogenetic netwoks can be considered in the future.

3.2. Hypothesis: Phylogenetic Trees versus Phylogenetic Networks

As well as phylogenetic trees, other structures such as phylogenetic networks are suitable for phylogenetic analysis because they represent a generalization over trees [52]. In particular, they exhibit an extra level of abstraction for modeling evolution based on the inclusion of potential multiple roots, horizontal transferences, sexual reproduction and the existence of cycles.

In fact, the inclusion of cycles in the phylogenetic network is an useful idea. This involves the addition of new semantic information to the phylogeny. For example, a phylogenetic network can be understood as a compact representation of a set of phylogenetic trees where branches and nodes overlap in the same structure [172]. Thus, the verification process focuses not only on the validity of the logical proposition but the detection of trees in which they hold. The identification of which nodes belong to each tree is easily solved by defining the membership with a explicit variable in every node. This feature represents an efficient way of processing several trees together.

Cycles in a phylogenetic network could also stand for the horizontal transference

of fragments of sequences (genes) between individuals of simple organisms like bacteria, while they would represent sexual reproduction for the rest of animals and plants. In this case, there would be only one phylogeny under study but the complexity of the verification process would be increased due to the explosion of paths.

As the notion of cycles is already included in the definition of Kripke structures, and both phylogenetic interpretations of cycles are compatible with the semantics of temporal logic, phylogenetic networks are also feasible in the context of model checking. Relevant phylogenetic properties expressed in temporal logics can be imported to phylogenetic networks with fairly minor changes.

Then, phylogenetic networks are more powerful and expressive. However, we maintain the use of phylogenetic trees in the rest of this dissertation for simplicity and compatibility with previous studies: phylogenetic trees continue to be useful and widespread abstractions of the evolution over time. This thesis places the essentials for the introduction of the verification process of phylogenetic properties using model checking, and therefore trees suffices for the first steps. Besides, a tree-like structure simplifies the use and adaptation of model checking. In any case, we desire to remark that the work developed here is flexible enough for the extension of phylogenetic analysis to other data structures in the future, such as networks.

Part II

Phylogenetic Properties: A Logic for Qualitative Logical Properties

The inspection of properties over a phylogeny needs the adoption of a temporal logic capable of capturing the essence of these biological hypothesis. The specification of phylogenetic properties with temporal logic is presented here for the first time. The existence of coalescence, conservation or covariation of sequences, gene inversions, reversions and a complete set of mutation patterns can be analyzed with a basic temporal logic operating over the paths of the tree. They allow the study of the topological structure, obtaining meaningful information about the peculiar nucleic arrangements and aggregations over the tree, and possibly examining the phylogenetic quality by recognizing anomalous characteristics. The first chapter of this part is dedicated to classify the kind of properties that can be represented with a branching-time temporal logic and illustrate the methodology for its definition.

By default, we organize the properties in three families depending on the phylogenetic information they require: a) topological or tree properties, that need the structural information of the phylogeny, b) sequence or state properties, that work with the sequences attached to the nodes, or c) a combination of both. For each case, we proceed incrementally with a modular composition and specification of subproperties in order to show every step of the design.

Subsequently, the next step faces the verification of those phylogenetic properties over a specific model checking tool. There, we show how to translate the phylogenetic tree into the syntax of a generic software and code the specifications accordingly. In case of failure to accomplish the property, a counterexample is returned by the tool. We use an example to clarify the interpretation of the counterexample output. Finally, the performance results demonstrate the feasibility of our approach over human mitochondrial DNA data and sketches future work.

Chapter 4

A Temporal Logic for Phylogenetic Analysis

No, no, you're not thinking; you're just being logical.

— Niels Bohr

4.1. Introduction

This chapter describes a methodology to specify biological properties of phylogenies employing the previous logical framework. Syntactically, a non-trivial property can be broken down into simpler, meaningful ones, and synthesized from these. The benefits of such logical decompositions are twofold: first, they simplify formalization and favor readability; second, modular properties can be reused in complex constructs, and variations on a formula can be produced by local adjustments.

The main idea is to apply formulas in temporal logic to extend the aptitudes of current phylogenetic analysis tools and cover the increasing necessities of biologist for the verification of hypothesis (see Section 1.3). These tasks include the detection of common ancestors, monophyletic clades [64], reversions (back mutation) [65, 1], and probably extend the hypothesis testing with quantitative information for the study of speciation rates (extinction and diversification) and mutation rates in DNA substitution models [117, 53].

Four classes of queries can be identified: *global*, *topological* or *tree properties*, in which the structure of the phylogeny itself is placed under scrutiny and relations between taxa inspected; *local* or *sequence properties*, where compositional sequence features take center stage, possibly aided by the placement of constraints; *a combination of both*, where the tree topology and the sequence alignment are

simultaneously required; and *quantitative properties*, that extends the preceding properties with explicit time and probabilities. In the following sections we describe the relevant properties of each group and exemplify their formal modeling (quantitative properties are studied apart in Part IV–V). Table 4.1 summarizes relevant phylogenetic properties according to the classification introduced.

4.2. Tree Properties

Tree properties are mainly of a cladistic nature, asking about the organization of the phylogenetic tree in compatible regions and the biological characteristics that are satisfied in them [89]. One of the most frequent basic queries is whether a set of extant organisms S under study constitutes a *monophyletic group* or *clade* in a phylogeny. That is, does the phylogenetic tree contain a subtree that has exactly those organisms as its leaves? Formally, the PTL formula over the Kripke structure will be true if there exists somewhere in the tree a reachable state (**EF**) which is the root of the group S, and thus: a) everything that has to be in, is in; and b) everything that has to be out, is out (there are no outsiders, or conversely, only insiders can be found inside):

$$clade\,(S) \equiv \mathbf{EF}\,(in\,(S) \wedge out\,(S)) \tag{4.1}$$

The inclusion rule states that for each unique sequence s of the set S (and-logic $\bigwedge$) there is a path that finds it as its leaf; the exclusion rule demands that all paths end in a leaf from the same set. Within infinite computations, leaves are found either through a *terminal* boolean variable or through the pattern $\mathbf{F\,AG}\,(s)$, since by construction whole uniform computation subtrees can only be found precisely as a consequence of leaf states:

$$in\,(S) \equiv \textstyle\bigwedge_{s \in S} \mathbf{EF\,AG}\,(s) \tag{4.2}$$

$$out\,(S) \equiv \mathbf{AF\,AG}\left(\textstyle\bigvee_{s \in S} s\right) \tag{4.3}$$

As noted, individual properties in isolation have useful semantics in their own right. Here, $in\,(S)$ is satisfied by all containing clades, and $out\,(S)$ by all strict subclades. In particular, the roots (non-terminal nodes) satisfying $in\,(S)$ define the group of *common ancestors* (CA):

$$CA(S) \equiv \neg terminal \wedge in(S) \tag{4.4}$$

By default, the set of taxa, and consequently their sequences, are located in the tips. However, if the phylogeny is extended with both ancestral and leaf sequences, the structure of the clade property remains unaltered. Only the inclusion and

exclusion rules need to be tuned, so that target sequences must be found anywhere in the subtree, and the whole subtree is free from intruders, respectively:

$$in'(S) \equiv \bigwedge_{s \in S} \mathbf{EF}(s) \tag{4.5}$$

$$out'(S) \equiv \mathbf{AG}\left(\bigvee_{s \in S} s\right) \tag{4.6}$$

In this example, the temporal formula $in(S) \wedge out(S)$ also describes a characteristic function that symbolically defines the set of organisms belonging to the clade by the intersection of states satisfying the inclusion and exclusion properties. In contrast, the evaluation $clade(S)$ returns a boolean value that tells whether the initial state of the Kripke structure satisfies the phylogenetic property. Thus, we emphasize the two aspects of temporal logics: as a symbolic representation for manipulating sets, and as a boolean function.

A more challenging question in phylogenetics is whether, given a phylogenetic tree and a partition of its leaves (which arises from the application of a sequence classification scheme), the latter constitutes a *haplogroup classification* in the tree. Haplogroups are aggregations of related haplotypes. An haplotype is a set of states of the phylogenetic tree that are identifiable by characteristic polymorphisms (i.e., point mutations that act as genetic markers). Thus, they define genetic populations and usually mark these geographically as well [168]. Their study is focused on the non-recombining regions of the genome, in particular mitochondrial DNA, where the original cladistic notation for haplogroups originated [144].

Essentially, a haplogroup together with the set of populations (child haplogroups) that have sprung from it over time must form a clade. In other words, a haplogroup is a *nested clade*: its members occupy all the leaves of a subtree, except possibly a number of sub-subtrees, which are completely devoid of members and themselves have a nested clade structure. A phylogeny is an acceptable classification if every part has a haplogroup-like structure:

$$classifier(S_1, S_2, \ldots, S_h) \equiv \bigwedge_{i=1}^{h} haplogroup(S_i) \tag{4.7}$$

Checking this property is trivial if the relations between parent and child haplogroups are known, symbolically by means not of a formula, but of a function $children(S)$. The function $children(S)$ may be extended to determine whether a suitable set of child haplogroups exists:

$$haplogroup'(S) \equiv clade(S \cup children(S)) \tag{4.8}$$

However, it is often interesting to allow for flexibility in the haplogroup placement in the ongoing study, the refinement of coarse haplogroups and the exploration of alternative hypotheses. Whereas Equation 4.8 may be extended to determine whether a suitable set of child haplogroups exists, it suffices to check the

local haplogroup-like quality of each individual part without resort to any additional information beyond the composition of the target part.

Formally, haplogroup-likeness is a relaxation of *clade* (S) through its local structure. While the inclusion rule is preserved, as is the general search for the root of the haplogroup, the exclusion rule is replaced by the nested clade property. Thus, all paths eventually reach a point (**AU**) where they either arrive at a member subclade or at the root of a foreign clade, always through ancestral nodes ascribed to the haplogroup in question (for this, a membership function h_i must be provided for each S_i):

$$haplogroup\,(S,h) \equiv \mathbf{EF}\,(in\,(S) \wedge nested\,(S,h)) \tag{4.9}$$

$$nested\,(S,h) \equiv \mathbf{A}\,[h\,\mathbf{U}\,out\,(S) \vee nesting\,(S)] \tag{4.10}$$

$$nesting\,(S) \equiv \mathbf{AF}\,\mathbf{AG}\left(\neg \bigvee\nolimits_{s\in S} s\right) \tag{4.11}$$

Notice that $nesting\,(S)$ is the opposite of $out\,(S)$. Obviously, terminal haplogroups (i.e., clades) are accepted by the formula. In cladistic terms, a local haplogroup-like structure corresponds to the concept of *polyphyletic group*. In this case, ancestral membership information becomes necessary because the leaf content of the polyphyletic group is indistinguishable from the *paraphyletic group*.

Nevertheless, the assumption that ancestral members of a haplogroup satisfy its defining properties is certainly reasonable. As before, the incorporation of ancestral taxa derives a related family of properties which allow a more comprehensive evaluation of the process of evolution.

Thus, the concept of clade, paraphyly and polyphyly, can be redefined using ancestral membership information by means of the *most recent common ancestor* (MRCA)[1], which is a restriction of the common ancestor set:

$$MRCA(S) \equiv CA(S) \wedge \neg\mathbf{EX}(CA(S)) \tag{4.12}$$

For example, the new reformulation of the clade property means that all the members of the monophyletic group share the same closest common ancestor and the resulting set is free of intruders:

$$clade(S) \equiv MRCA(S) \wedge out'\,(S) \tag{4.13}$$

Polyphyly encompasses a set of elements that share similar traits derived by convergent evolution or reversions. In addition, a polyphyletic group is a "group whose members' last common ancestor is not a member of the group" [64]. The set nodes of the phylogenetic tree that validate the following formula form a polyphyletic group:

[1]In gene genealogy, coalescent theory tries to trace back all alleles of a gene shared by all members of a population to a single ancestral copy, i.e., the MRCA [94].

$$polyphyletic(S) \equiv MRCA(S) \notin S \tag{4.14}$$

The paraphyletic group is a recursive specialization of the clade property. It consists of "all the descendants of a hypothetical closest common ancestor minus one or more monophyletic groups" [64]. The detection of nested clades must be done explicitly with a previous declaration of the internal clades that we want to exclude from the paraphyletic group. That is, all the elements of S should be arranged into subgroups that define up to h disjoint clades and every node of the set S must belong to one of them. Thus, a paraphyletic group is formed from the closest common ancestor plus a selection of clades. The verification of the following formula gives the paraphyletic root:

$$\begin{array}{rcl} paraphyletic(S, S_1, \ldots, S_h) & \equiv & MRCA(S) \wedge \\ disjoint(S, S_1, \ldots, S_h) & \wedge & \mathbf{AG}(\bigvee_{i=1}^{h} clade(S_i)) \end{array} \tag{4.15}$$

Further topological-related properties of phylogenetic trees are described in [64]. In particular, synapomorphies, symplesiomorphies and autopomorphies focus on the depth of the paths with derived traits while apomorphies, plesimorphies and homomorphisms (non homology) are interested in the point of divergence of derived and ancestral traits in the hierarchy of evolution (Figure 5.1). In the picture, dark dots represent derived characters and white nodes represent the ancestral ones.

A taxonomy of important biological properties is summarized in Table 4.1, together with their temporal logic formulas. The notation seq_s is a compact representation for specifying the sequence associated to the node s, denoted as $M, s \vDash seq$. In some cases (i.e., apomorphies or synapomorphies) we need to expand the PTL logic to include the past operator $\mathbf{X}^{-1}$, that means the previous (parent) node of the current state. The extension of temporal logics with past operators has already been described in the literature [112]. The CTL path quantifier of $\mathbf{X}^{-1}$ is not necessary because the phylogenetic trees has a unique ancestor. In the case of phylogenetic networks, **A** and **E** path quantifiers shall be considered.

Finally, $distance(S)$ is a distance metric that estimates the evolutionary divergence of a set of species. Although not strictly necessary, the definition of topological properties in phylogeny are simplified using distances. Properties such as parallel and convergent evolution, that are clearly distinguishable if the information about the traits of internal nodes is available, are complex to differentiate when the phylogeny lacks of this internal information. Two independent lineages whose leaves are similar with respect to a particular trait describe a parallel evolution if the ancestors considered are also similar, and convergent if they are not [118, 180]. Nevertheless, all organisms share a common ancestor more or less recently, and the selection of these ancestors at different deeps may change the consideration of

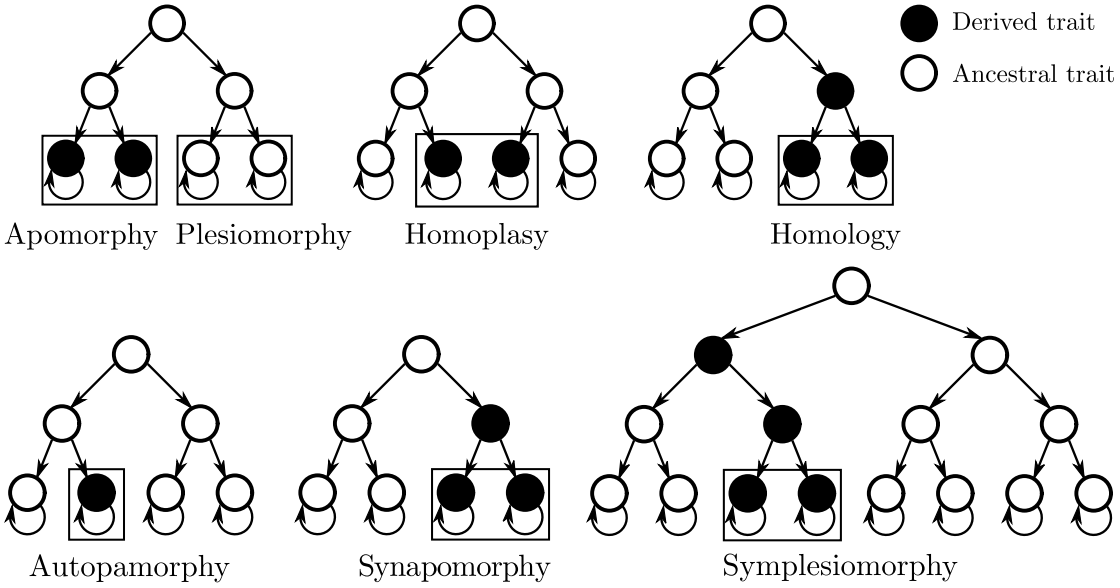

Figure 4.1: Boxed nodes indicate the states where the corresponding property is held.

the lineages as parallel or convergent [11]. Hence, these definitions of parallel and convergent evolution pose the next question: how far back shall we to look in the evolutionary time? The introduction of time constraints in the logical definition of phylogenetic properties nuances the concept of parallel and convergent evolution. The definition of more quantitative properties with time and probabilities is explained in Part IV–IV.

4.3. Sequence Properties

In general, sequence properties are based on state formulas, i.e., those that are evaluated within a node and without resort to temporal operators. Usually, they are composed of simple PTL patterns whose application scope is restricted to the surrounding nodes or the entire phylogeny. Such types of state formulas will be called *patterns(p)*. They offer a powerful descriptive formalism for formulating general restrictions without the limitations of ad hoc approaches. Often, these properties may be used not necessarily to forbid patterns, but as queries and alerts to signal unusual, possibly anomalous behavior, and mark it for further study. Their infraction can be interpreted as either the discovery of a evolutive novelty (advantageous or not), or spurious errors caused by diverse factors (sequencing fails, incongruousness in the phylogenetic inference method).

Type	Name	Ref	PTL Formula
T	CA	Sect. 4.2	$CA(S) \equiv \neg terminal \wedge in(S)$
T	MRCA	Sect. 4.2	$MRCA(S) \equiv CA(S) \wedge \neg \mathbf{EX}(CA(S))$
T	Monophyly	[64]	$clade(S) \equiv MRCA(S) \wedge out'(S)$
T	Polyphyly	[64]	$polyphyletic(S) \equiv MRCA(S) \notin S$
T	Paraphyly	[64]	$paraphyletic(S, S_1, \ldots, S_h) \equiv MRCA(S) \wedge \mathbf{AG}(\bigvee_{i=1}^{h} clade(S_i)) \wedge disjoint(S, S_1, \ldots, S_h)$
T	Homology	[64]	$homology(S, col) \equiv \exists \sigma \in \Sigma,\ \forall s \in S\,(seq_s[col] = \sigma) \wedge \left(seq_{MRCA(S)}[col] = \sigma\right)$
T	Homoplasy	[64]	$homoplasy(S, col) \equiv \neg homology(S, col)$
T	Parallel Evolution	[64]	$parallel(S, col) \equiv homoplasy(S, col) \wedge distance(S) \leq threshold$
T	Convergent Evolution	[64]	$convergent(S, col) \equiv homoplasy(S, col) \wedge distance(S) > threshold$
T	Apomorphy	[64]	$apomorphy(S, col) \equiv \exists \sigma \in \Sigma,\ \forall s \in S\,(seq_s[col] = \sigma) \wedge (seq_{\mathbf{X}^{-1}s}[col] \neq \sigma)$
T	Plesiomorphy	[64]	$plesiomorphy(S, col) \equiv \neg apomorphy(S, col)$
T	Synapomorphy	[64]	$synapomorphy(S, col) \equiv homology(S, col) \wedge \left(seq_{\mathbf{X}^{-1}MRCA(S)}[col] \neq \sigma\right)$
T	Symplesiomorphy	[64]	$symplesiomorphy(S, col) \equiv homology(S, col) \wedge \neg synapomorphy(S, col)$

Table 4.1: (1) Summary of the most important phylogenetic properties (Type T: tree; S: sequence; Q: quantitative).

T	Autapomorphy	[64]	$autapomorphy(s, col) \equiv \exists\sigma \in \Sigma, \ (seq_s[col] = \sigma) \wedge (seq_{\mathbf{X}^{-1}s}[col] \neq \sigma) \wedge terminal(s)$
S	Covariation	[26]	$covariation(i, j) \equiv \mathbf{EF}\,\mathbf{AG}\,(seq[i] = \sigma_i \rightarrow seq[j] = \sigma_j)$
S	Conservation	[26]	$conservation(i, j) \equiv \mathbf{EF}\,\mathbf{AG}\,(seq[i, \ldots, j] = \sigma_i \ldots \sigma_j)$
T & S	Back Mutation	Sect. 4.4	$\begin{aligned} detectBM &\equiv \forall j \in \{1, l\},\ \mathbf{AG}\,(\neg hasBM\,(j)) \\ hasBM\,(col) &\equiv \exists\sigma \in \Sigma,\ (seq[col] = \sigma) \\ &\wedge\ \mathbf{EF}\,(seq[col] \neq \sigma \wedge \mathbf{EF}\,(seq[col] = \sigma)) \end{aligned}$
T & S	Multiple Nucleotide Polymorphism	Sect. 4.4	$\begin{aligned} detectMNP &\equiv \forall j \in \{1, l-1\}, \\ &\mathbf{AG}\ (\neg startsMNP\,(j)) \\ startsMNP\,(col) &\equiv \exists\sigma, \tau, \upsilon \in \Sigma,\ (seq[col-1] = \sigma) \\ &\wedge\ (seq[col] = \tau) \\ &\wedge\ (seq[col+1] = \upsilon) \\ &\wedge\ \mathbf{EX}[(seq[col-1] \neq \sigma) \\ &\wedge\ (seq[col] \neq \tau) \\ &\wedge\ (seq[col+1] \neq \upsilon)] \end{aligned}$
Q	Distance[a]	Part IV–V	$distance(d, event) \equiv \mathbf{AF}^{\leq d} event$

Table 4.2: (2) Summary of the most important phylogenetic properties (Type T: tree; S: sequence; Q: quantitative).

[a] For the syntax of the quantitative formula, see Section 11.2

As a common understanding, closely related members share analogous genomes, satisfying identical patterns and properties. For example, the preservation of biochemical properties is essential for maintaining the stability of the resulting protein and the viability of an organism, placing important constrains to the evolution [86]. The amino acids conform the raw material of the protein, and consequently they determine the three-dimensional structure and singularities of the molecule. Proteins develop crucial functions in living organisms. The modification of a nucleic base, and indirectly its associated amino acid, may bring a recalibration in the molecular morphology which damages its functionality and propagates the instability to a chain of biological reactions [50, 173]. Therefore, natural selection favors the persistence of the molecular structure limiting the mutations to compatible amino acids. The existence of synonymous substitutions[2] provides some flexibility to the change of nucleic bases, but in any case the conservation of biochemical properties and sequences reaffirms the use of patterns.

When a nucleic modification inevitably changes the protein morphology or alters the biochemical equilibrium, the effects in the organism are unpredictable. In the worst case, the mutation becomes lethal and punishes the survival of the individual. In the best situation, it is ubiquitous or a new evolutionary advantage arises at the conclusion of this process. In an intermediate context, a single mutation occasionally triggers a group of (intra or inter-gene) changes that neutralize the consequences of the original modification in a short period of time. In other words, they try to compensate the attributes of the new amino acid in order to maintain the biochemical equilibrium. This character coevolution is often captured in the vicinity of the phylogeny [77]. The number and degree of pattern violations (nucleotide and biochemical conservation, protein shape) together with their circumstances assists to the discernment of the repercussions caused by the mutation.

According to the previous comments, patterns represents global correctness constraints that are supposed to hold across the whole phylogeny. They can be categorized as follows:

- *Conservation* is modeled as a restriction on the symbols that can occur at a given position in a sequence. Commonly, the pattern will codify a unidimensional boolean table that classifies each symbol as permissible or impermissible. However, it is possible to define general families of compatible elements, not bounded to specific positions, and restrict their usage to exactly one of these positions, among other extensions.

- *Covariation* imposes a relation of dependence between two (or more) positions in a sequence. It can be represented as a bidimensional boolean table

[2]Codons that transcript the same amino acid

which states, for each symbol in the first column, the set of symbols that may appear in the second column. Typically, for the property to be meaningful, associations between symbols will be sparse.

- A combination of both.

A global pattern thus defined is easily verified by extending it over the computation tree:

$$global\,(p) \equiv \mathbf{AG}\,(p) \tag{4.16}$$

Exceptions to the aforementioned properties may in fact indicate suspicious or potentially deleterious mutations, which are of great interest in applied phylogenetic studies [129]. Furthermore, known or suspected mutations of this kind can be explicitly modeled as patterns and their positioning in a phylogeny assessed. In particular, those affecting important metabolic functions are expected to prevent or hinder the reproduction of the organism, and consequently should be confined in or near terminal leaves.

Just as some mutations may ordinarily be forbidden altogether as global patterns, observed and feasible deleterious mutations may be permitted subject to certain restrictions. Specifically, it may be demanded that, if a hazardous pattern appears, it has no offspring, i.e., it is a leaf in the phylogeny; or, to provide some flexibility, it may be allowed that all descendants, if any, are reached in at most k steps ($\mathbf{AX}^k$):

$$terminal\,(p) \equiv \mathbf{AG}\,(p \rightarrow leaf) \tag{4.17}$$

$$terminal\,(p, k) \equiv \mathbf{AG}\left(p \rightarrow \mathbf{AX}^k\,(leaf)\right) \tag{4.18}$$

In this case, leaves (self-loops in the Kripke structure) must be detected without reference to any particular sequence. This is easily achieved by performing an equality comparison between the valuations of AP of the target state and all its successors:

$$leaf \equiv \bigwedge_{p \in AP} p \leftrightarrow \mathbf{AX}\,(p) \tag{4.19}$$

This last example is representative of properties that perform *conditional explorations* of the phylogeny. Lineage-specific verification represents a further step forward, where patterns would be used to define the sets of states of interest. Notice that pattern-based checking of haplogroup classifications falls within this category. Single nucleotide polimorfisms (SNP's) define genetic markers that differentiate members of the same biological specie.

4.4. A Combination of Tree and Sequence Properties

Some properties do not fit exclusively into one of the previous classifications but are a mix of complex sequence and tree properties. At first sight, it is reasonable to consider relatively simple properties based on a tree topology and an associated sequence alignment, exemplified by the following real properties. Suppose for now that the alignment comprises a number of cladistic characters indexed 1 through l, sequences seq are words of length l over an alphabet Σ, and $seq[i] = \sigma$ ($seq[i] \neq \sigma$) means that $\sigma \in \Sigma$ appears (not) in position i in a state sequence (recall the AP definition in Definition 3). For readability and compactness in these examples, we will refer to the atomic propositions $seq[i] = \sigma$ ($seq[i] \neq \sigma$) of a state sequence as σ_i ($\overline{\sigma}_i$). In the case of finite domains, such as the set of DNA sequences, the evaluation of logical quantifiers $\forall$ and $\exists$ can be substituted by multiple instances of boolean formulas connected by the $\bigwedge$ and $\bigvee$ operators.

Consider the following example. It determines whether a given tree is free of back mutations, which we abbreviate BM (equivalently, it detects those points in the tree where back mutations occur, if any).

$$detectBM \equiv \bigwedge_{j=1}^{l} \mathbf{AG}\left(\neg hasBM\left(j\right)\right) \tag{4.20}$$

$$hasBM\left(col\right) \equiv \bigvee_{\sigma \in \Sigma} \sigma_{col} \wedge \mathbf{EF}\left(\overline{\sigma}_{col} \wedge \mathbf{EF}\left(\sigma_{col}\right)\right) \tag{4.21}$$

In these two formulas we present a non-trivial modeling example of a cladistic property with a heavy use of sequence data. The goal is to detect back mutations in the tree, which may be encouraged by the recovery of an ancestral phenotype or a random walk in a low selection pressure zone in the DNA [65, 1]. To this end, we need to formalize the concept of back mutation: given a node in the tree (which itself defines a subtree) and an alphabet Σ, there is a back mutation in that subtree involving a position of the alignment, say j, if at some point in some descending path from the node we find in position j a different symbol ($\overline{\sigma}_j$) than that found in the root of the subtree (σ_j), and if at some point in the subtree hanging from that intermediate the symbol from the root reoccurs. The formula $hasBM\left(col\right)$ models this condition by nesting **EF** operators (a node satisfies a property which eventually some other descendant does not satisfy, but is fulfilled once again at some point in the future) and repeating the check for every symbol that may occur in the node. Finally, the global formula $detectBM$ iterates the check over the positions of the alignment and extends it to all tree nodes.

The second example detects a different family of complex mutations: those which affect groups of consecutive positions, a pattern of particular relevance to

protein-coding regions. We dub this *multiple nucleotide polymorphism* (MNP), as opposed to SNP.

$$detectMNP \equiv \bigwedge_{j=1}^{l-1} \mathbf{AG}\left(\neg startsMNP\left(j\right)\right) \tag{4.22}$$

$$startsMNP\left(col\right) \equiv \bigvee_{\sigma,\tau,\upsilon\in\Sigma} \left(\sigma_{col-1} \wedge \tau_{col} \wedge \upsilon_{col+1} \wedge \mathbf{EX}\left(\overline{\sigma}_{col-1} \wedge \overline{\tau}_{col} \wedge \overline{\upsilon}_{col+1}\right)\right) \tag{4.23}$$

where for brevity the special treatment required when $col = 1$ has been omitted. In this case, σ, τ and υ are characters of the state sequence *seq* in positions *col*, $col - 1$ and $col + 1$ respectively. The property *startsMNP* can be extended to accept these characters as input parameters. It is clear that any verification of properties from the alignment (e.g., covariation) can be fulfilled in the phylogeny.

Generally speaking, haplogroups are defined as clades whose members share a common mutation denoted by a SNP or MNP. The parametrization $startsMNP\left(col, \sigma, \tau, \upsilon\right)$ will work as the membership function h_i defined in Equation 4.9–4.11 as it will label the clade root (*MRCA*) which has the nucleotides σ, τ, υ around the position *col*. Given a set of clades and a known MNP, the detection of the most likely haplogroup will consist of the selection of the root with the greatest subtree satisfying the *detectMNP* property. Next, we must check the conservation of the MNP for all the populations inside the clade so that no spurious or external haplogroups (*startsMNP*) corrupt the current subtree. Using model checking packages, a fixed point algorithm will symbolically compute the set of states satisfying *startsMNP* for each candidate of column and characters. The objective for the haplogroup fitness is the selection of the ancestor with the maximum group of descendants verifying the property. Other related biological properties would have different maximization (minimization) purposes.

Chapter 5

Tools and Experiments

-"Alright," said Deep Thought. "The Answer to the Great Question..."
-"Yes...!"
-"Of Life, the Universe and Everything..." said Deep Thought.
-"Yes...!"
-"Is..." said Deep Thought, and paused.
-"Yes...!"
-"Is..."
-"Yes...!!!...?"
-"Forty-two," answered Deep Thought, with infinite majesty and calm.
-"Forty-two!" yelled Loonquawl. "Is that all you've got to show for seven and a half million years' work?"
-"I checked it very thoroughly," said the computer, "and that quite definitely is the answer. I think the problem, to be quite honest with you, is that you've never actually known what the question is."

— Douglas Adams, *The Hitchhiker's Guide to the Galaxy*

5.1. Introduction

The use of model checking techniques is domain-independent: given a phylogenetic tree and a specification of its biological properties, the verification software automatically checks the correctness of the system. In the event of failure to comply with the specification, the software outputs the scenarios which infringe the property as counterexamples.

This chapter starts introducing a list of model checking tools adapted for different temporal logics. It also describes the implementation of the branching-time phylogenetic tree (Definition 3) into a model checking verification tool and evaluates its performance using phylogenetic properties. More in detail, we evaluate a set of back mutation properties over a real data set with the NuSMV tool [45]. In future sections and chapters we will also use the model checking software PRISM [124]. Although there exist multiple model checking tools as explained in Section 5.2, NuSMV and PRISM suffice for our requirements.

5.2. Model Checking Tools

The consideration of the different types of temporal logics involves the implementation of a pool of model checker packages for supporting them (see Table 5.1). With respect to stochastic and quantitative temporal logics, PRISM is a model checker adapted for "formal modeling and analysis of systems that exhibit random or probabilistic behavior" [124]. In addition, PRISM has been successfully applied in a nearby application domain such as biological pathways [103, 123, 104, 73]. It seems suitable for the statistical analysis and quantitative evaluation of properties (e.g., mutations).

UPPAAL is a commercial software package that offers "an integrated tool environment for modeling, validation and verification of real time systems modeled as networks of timed automata" [21] . This feature can be exploited in biology so as to provide an explicit definition of evolution clocks. Furthermore, NuSMV [45], a powerful tool for verifying CTL formulas, includes quantitative extensions related to the computation of paths with maximum (minimum) lengths between sets of nodes satisfying a property. It also includes a simple database manager for reusing the results of previous properties. In addition, the classic LTL open source model checker SPIN offers a framework that can be tuned as desired [93]. Sliced model checking presented in Section 7.3 can be coded over these two packages [139, 143].

In conjunction with the previous logics, on-the-fly model checking improves performance by exploring the state space on demand [24]. One of the best model checkers in this field is PROD [169]. Moreover, multicore and network-oriented tools such as DiViNe [15] (for LTL), Murphy [126] or PVeStA [5] (stochastic) are available in order to speed up efficiency.

Finally, TLQSolver is a model checker that manages CTL queries and allows the mining of properties that match a specific pattern [40]. The definition of patterns helps to select sets of nodes of the phylogenetic tree satisfying an abstract relation in a similar way to an SQL query. These result sets can then be analyzed in order to extract the differences among them.

Name	Main feature	Properties Language	Platform
NuSMV [45]	OpenSource	LTL, CTL, RTCTL, PSL	Windows, Unix, MacOS
PROD [169]	On-the-fly Model Checker	CTL	Linux
SPIN [93]	Generic Model Checker	LTL	Windows, Unix
DiViNe Tool [15]	Distributed Multicore Model Checker	LTL	Unix
Eddy Murphi [127]	Distributed Multicore Model Checker	Assertions	Unix
PVeSta [5]	Statistical Parallel & Multicore Model Checker	PCTL	Windows, Linux, MacOS
PRISM [124]	Probabilistic & Quantitative Logics	PCTL, CSL, LTL, PCTL*	Windows, Linux, MacOS
UPPAAL [21]	Commercial Software for Real Time Systems	TCTL	Windows, Linux
TLQSolver [40]	Temporal Logic Query Checker for Mining Properties	Query CTL	Linux

Table 5.1: List of some available model checkers.

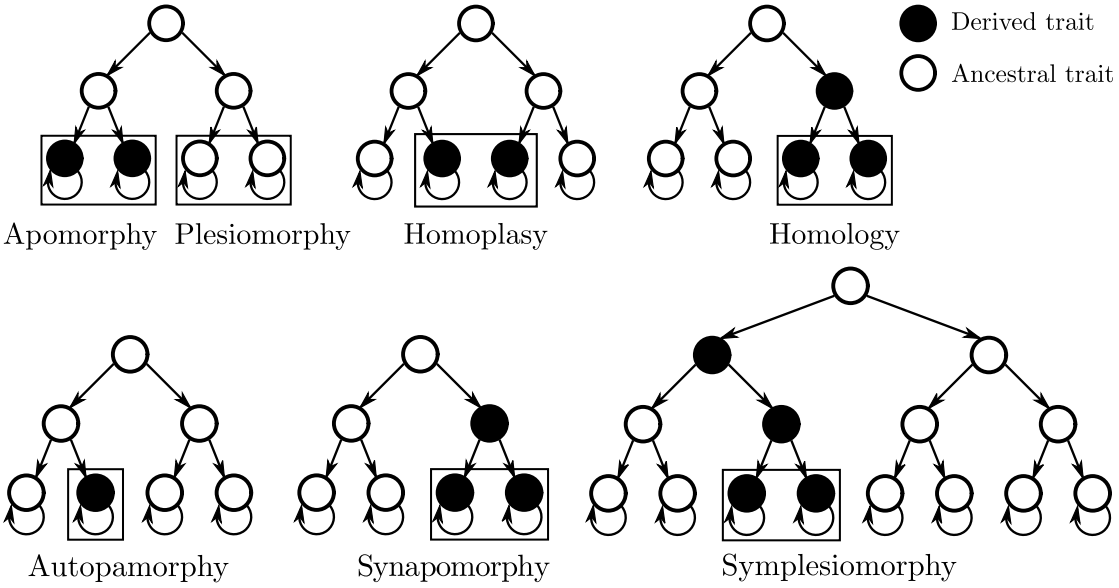

Figure 5.1: Boxed nodes indicate the states where the corresponding property is held.

5.3. Codification of Branching-time Phylogenies

We have used two different kind of model checkers for the implementation of the branching-time phylogenetic tree. First of all, NuSMV [45], a well-known model checking software tool compatible with Cadence SMV [125, 138], is used for the evaluation of qualitative properties defined in temporal logic. Secondly, PRISM [124] is used for the evaluation of quantitative and probabilistic properties in Part IV–V. They are both freely available and they support the implementation of the whole spectrum of biological properties defined in Table 4.1.

A description of the Kripke structure and the atomic propositions in the NuSMV syntax must be provided by the user as the input for the model checker. To this end, we precompute the sequence alignment and a phylogenetic tree. The translation from the phylogenetic tree to the NuSMV syntax has been performed automatically by a BioPerl script [153]. The script can be upgraded in order to include extra features such as the generation of multiple instances of phylogenetic trees, bootstrapping and so on. The inclusion of vector constructs, macros and a rich support for various logics operators in the NuSMV syntax facilitates a compact characterization of DNA and protein sequences as strings of characters.

Figure 5.2 shows the implementation of the branching-time phylogenetic tree of Figure 2.1 in SMV code. The main module describes the topology of the evolutionary tree, where the names of the tree nodes (taxa) are defined symbolically

$(X, \ldots, S)$, and id variables label the states. In this case, only the DNA string is defined but extra information could be considered in future works. The *init* and *next* clauses are used to mark the root of the tree and the successors of a given state. The second part of the description consists of a function returning the DNA string associated to each node.

Usually, model checking packages store the previous description of the transition system in an Ordered Reduced Binary Decision Diagram (ORBDD) [33]. These data structures are efficient graph representations for characterizing and manipulating sets or relations using boolean functions. Internally, each state of the Kripke structure is characterized by a set of atomic propositions showing the DNA associated to that state (i.e., the taxon sequence $seq_{i\sigma} \equiv (seq[i] = \sigma)$). Every atomic proposition can be expressed with a boolean variable indicating whether or not the character σ is present in the i^{th} position of the DNA sequence. Thus, those combinations that make true a boolean function $f(seq_{1A}, seq_{1C}, \ldots, seq_{lT})$ will identify the DNA sequences with length l belonging to the phylogenetic tree.

This feature provides an easy scheme for the manipulation of sets in a symbolic way because a string can be identified by means of a logical expression. It should be noted that the verification of a phylogenetic formula is intimately related to the identification and manipulation of sets satisfying a boolean function characterized in temporal logic. In fact, the last step of the verification process often consists of checking that the initial state (the tree root) belongs to the result set. However, the ordering of the boolean variables in the representation of the characteristic function has a high influence in the data structure and it determines the size (number of nodes) of the diagram. In the next section, we verify phylogenetic properties using the description depicted in Figure 5.2 as input data. We aim to discover the performance trends and analyze the impact of the variable ordering.

5.4. Performance Results for Monolithic Verification of Phylogenetic Properties

The performance evaluation of our system has been measured with human protein alignments retrieved from GenBank [23]. In particular, we selected genes of respiratory complex I encoded in mitochondrial DNA (mtDNA). We have chosen them because they are biologically interesting and varied in length, which makes them suitable for a complete performance analysis. This data set includes ND5, one of the biggest genes in mtDNA. Thus, the experimental results will define approximate upper bounds which can be used as a sound reference for mtDNA experiments. All tests have been run on a scientific workstation Intel Core 2 Duo E6750 @ 2.66 GHz, 8 GB RAM and Linux. Notice that the NuSMV uses a single

core.

We start analyzing the time and memory usage for the construction of the phylogenetic Kripke structure labeled with protein sequences. Table 5.2 and Table 5.3 show the time and memory consumption with respect to the sequence length and alignment size.

	Alignment Size					
	750	1000	1250	1500	1750	2000
ND4L (98)	13.5	27.6	46.0	67.4	89.6	124.7
ND3 (115)	7.0	31.4	53.6	78.1	105.7	142.8
ND6 (174)	18.1	26.3	71.6	123.2	160.8	205.9
ND1 (318)	8.8	13.2	84.2	203.3	246.7	371.6
ND2 (347)	10.8	13.9	75.8	217.5	322.5	420.0
ND4 (459)	14.2	22.4	29.4	191.3	417.4	499.0
ND5 (603)	19.0	92.1	106.8	314.9	518.5	728.8

Table 5.2: Seconds needed for the creation of the Kripke structure and the storage of protein sequences.

	Alignment Size					
	750	1000	1250	1500	1750	2000
ND4L (98)	102	123	146	170	194	211
ND3 (115)	111	135	161	193	218	244
ND6 (174)	161	199	233	267	300	351
ND1 (318)	275	349	409	472	533	595
ND2 (347)	307	388	454	520	586	679
ND4 (459)	423	512	598	712	798	890
ND5 (603)	568	682	823	940	1054	1168

Table 5.3: Megabytes needed for the creation of the Kripke structure and the storage of protein sequences.

The time increases quadratically with the number of sequences and linearly with the gene length, except for some erratic behavior due to the use of BDD diagrams (Figure 5.3). On the other hand, the memory usage increases sublinearly with the number of sequences and the gene length, which is very encouraging from a computational point of view. It is possible that these moderate trends are partly due to the use of highly conserved genes and closely related sequences. Nevertheless, the huge amount of time and memory required for the worst case indicates that the codification of data should be optimized.

The time required for the verification of a single temporal logic formula is extremely variable, as it depends on the complexity, the model checker search strategy (e.g., depth or breadth first search) and the occurrence of interruptions caused by counterexamples. In this example, we have tested the detection of back mutations (Equation 4.20) for the first 98 positions of the protein alignment. We have verified the equation for five different aminoacids, reaching a total of 490 evaluations. The time required for the initialization and evaluation of the batch of properties in NuSMV raises to 1729.33 seconds in ND5 (2000 tips).

The study and verification of structural properties only requires the topological information of the tree. Then, the DNA labels can be omitted from the associated Kripke structure, which leads to a resource-saving representation in the model checking tool. For example, the initialization of a binary phylogenetic tree with 2000 tips and no DNA tags consumes 19.2 megabytes and 4 seconds in NuSMV.

As well as the execution of properties in batch mode, NuSMV supports the addition of new phylogenetic properties via the command line in an interactive way. This feature simplifies the initialization phase and avoids multiple loads of the phylogenetic tree. NuSMV also includes a toy database manager for reusing the results of previous properties.

Figure 5.4 illustrates a counterexample returned by NuSMV when detecting a false property in the specifications. The trace corresponds to the result obtained for the evaluation of one of the previous back mutation properties for the first position of the DNA alignment. That is:

```
AG !(dna[1] = A & EF (dna[1] != A & EF dna[1] = A))
```

The use of the complementary expression $\mathbf{AG}\neg(x)$ instead of the original version $\mathbf{EF}(x) \equiv \neg\mathbf{AG}\neg(x)$ guarantees that the returned counterexample will match with one of the possibly multiple back mutations of the tree. The counterexample starts with a path emerging from the root (inode0) that finds a change to the nucleotide C in one of its direct descendants (inode3)[1]. Later, the verification process continues with the following state, a leaf that is associated to the taxon HQ286323. Finally, the model checker tool finds that the path reverts the nucleotide in that specie and it becomes again an A. The reversion located in that taxon can be caused by multiple biological factors. For example, a plausible origin of this back mutation is a combination of genomic changes in HQ286323 that makes unstable the DNA molecule if `dna[1]` doesn't come back to A (see Section 4.3). The evaluation of complementary specifications over the phylogeny may discard erroneous justifications and increment the overall knowledge about the constraints in the sequences.

[1]The notation inode is referred to the internal nodes of the tree.

These last results are only particular examples of verification, but they offer an insight into temporal costs and memory requirements for future experiments. It seems that the sequence length will be the main bottleneck if we straightforwardly apply our framework to bigger (e.g., nuclear) genes and phylogenies. Although these preliminary results might be discouraging in terms of efficiency, some techniques such as the exportation of DNA to external databases [143], and sliced [139] or distributed model checking [127, 15, 5] can be applied to relieve the memory constriction, fade out the initialization time and scale up the system. We emphasize that the implementation of these solutions is not completely optimized yet for this field, but the first performance tests suggest an acceptable trend in efficiency and scalability for the verification of boolean properties. Future optimizations will review all these aspects.

```
MODULE main
VAR
  /* States that represent taxa in the phylogenetic tree */
  taxon: {X,Y,Z,R,S};
  /* Function labelling with DNA the current node */
  sequence: process dna_taxon(taxon);
ASSIGN
  init(taxon) := X;            /* Tree root */
  next(taxon) :=               /* Successors of each node */
      case
         /* Nodes Y and Z are the successors of X */
         taxon=X: {Y, Z};
         /* Nodes R and S are the successors of Y */
         taxon=Y: {R, S};
         /* Self-loops in leaf nodes */
         TRUE: taxon;
      esac;

MODULE dna_taxon(taxon)
  /* Definition of the array of characters and the DNA alphabet */
  dna: array 1..3 of {-,A,C,G,T}
ASSIGN
  /* DNA for the tree root */
  init(dna[1]) := A;
  init(dna[2]) := A;
  init(dna[3]) := G;

  /* DNA for the rest of taxa */
  next(dna[1]) :=
  case
      taxon=Y: A;
      taxon=Z: G;
      taxon=R: A;
      taxon=S: A;
      TRUE: A;
  esac;
  ....
  next(dna[3]) :=
  case
      taxon=Y: G;
      taxon=Z: G;
      taxon=R: T;
      taxon=S: G;
      TRUE: G;
  esac;

  FAIRNESS
   running
--    SPEC EF AG dna[1] = A
```

Figure 5.2: Mapping of the phylogenetic tree of Figure 2.1 in SMV.

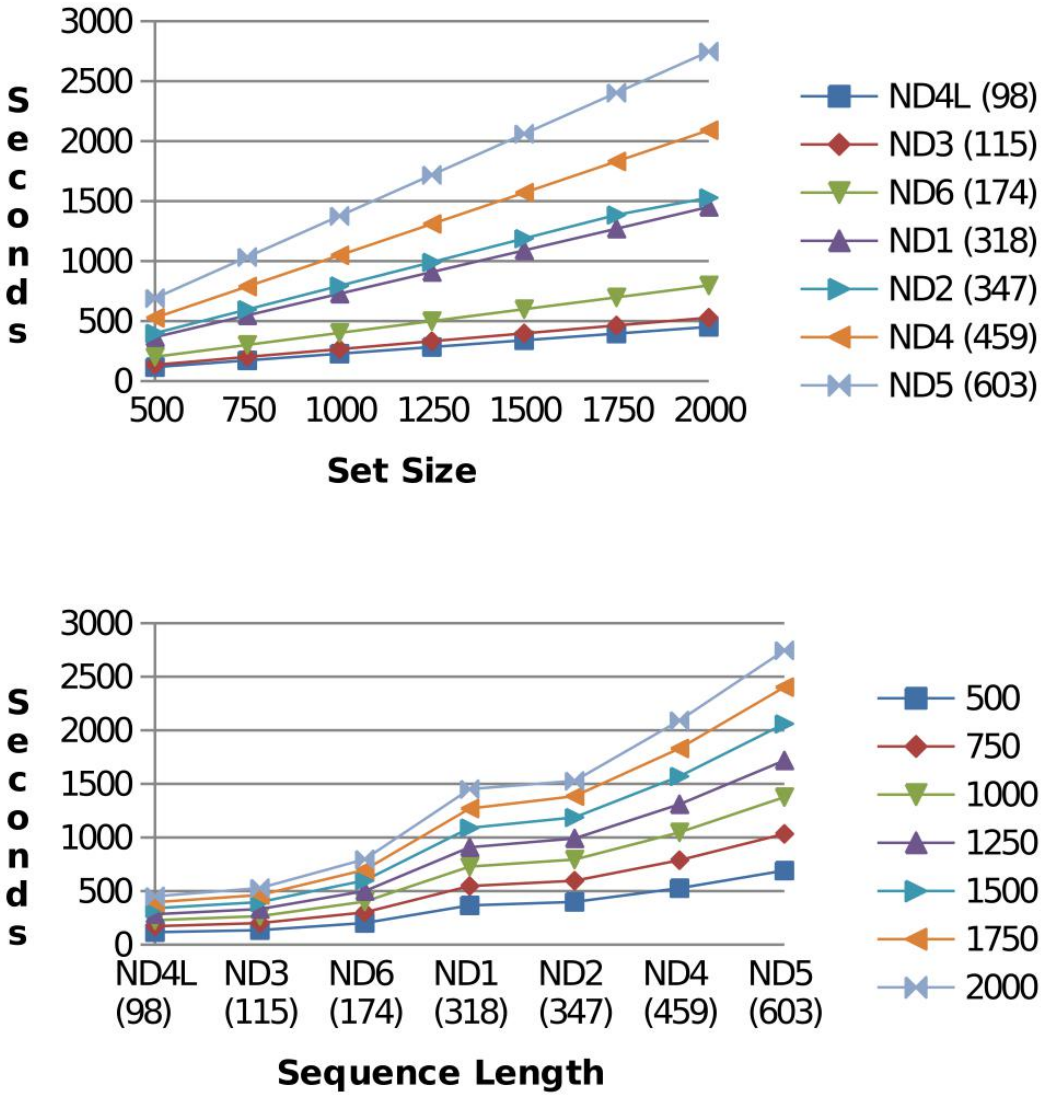

Figure 5.3: **(a)** Time is linear with respect to set size and **(b)** quadratic with respect to sequence length.

```
*** This is NuSMV 2.5.4 (compiled on Fri May  4
*** 09:46:59 UTC 2012)
*** Enabled addons are: compass
*** For more information on NuSMV see <http://nusmv.fbk.eu>
*** or email to <nusmv-users@list.fbk.eu>.
*** Please report bugs to <nusmv-users@fbk.eu>

*** Copyright (c) 2010, Fondazione Bruno Kessler

*** This version of NuSMV is linked to the CUDD library
*** version 2.4.1
*** Copyright (c) 1995-2004, Regents of the University of
*** Colorado

-- specification AG !(dna[1] = A & EF (dna[1] != A
   & EF dna[1] = A)) IN sequence is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
-> State: 1.1 <-
  taxon = inode0
  sequence.dna[1] = A
  ...
-> State: 1.2 <-
  taxon = inode3
  sequence.dna[1] = C
-> State: 1.3 <-
  taxon = HQ286323
-> State: 1.4 <-
  sequence.dna[1] = A
  ...
```

Figure 5.4: Counterexample of a back mutation property.

Chapter 6

Conclusions

> *Logic will get you from A to B. Imagination will take you everywhere.*
>
> — Albert Einstein

After the initial motivation and the introduction of the technical and conceptual basis in Part I, the next step consists of the specification and evaluation of phylogenetic properties. In the first chapter of this part, we have presented a classification of common phylogenetic properties according to the information they inspect. In particular, we have detected three kind of formulas: a) topological or tree properties, that need the structural information of the phylogeny, b) sequence or state properties, that work with the sequences attached to the nodes, or c) a combination of both. Moreover, we give a brief tutorial showing how to proceed methodologically for the incremental building and composition of specifications for each family. A taxonomy of important biological properties is summarized in Table 4.1, together with their temporal logic formulas. The inclusion of explicit time distances in certain properties motivates the addition of quantitative features in future sections.

The second chapter of this part is devoted to the experimentation. We start with the state of the art of model checking tools and the selection of the best one that fits for our requirements (mainly, the support of branching-time temporal logics such as CTL). To this end, we select NuSMV although others such as PRISM are suitable for future studies in Part IV–V. We continue with the modeling of the phylogeny and the specification of properties in NuSMV syntax. During the experimentation, we have evaluated formulas that search for back mutations in the tree.

We have seen that the initialization phase (creation of the associated Kripke structure) is much costlier than the verification process of a single formula. The experimental results show that the initialization time increases quadratically with

the alignment size and linearly with the sequence length. Additionally, memory consumption is sublinear in both cases. Despite the resources needed for model checking, it is still competitive thanks to the symbolic manipulation of huge amounts of data.

The data set used there consists of proteins coded by genes from the mtDNA genome, which are substantially smaller than those from nuclear DNA. As the temporal cost increases mostly with respect to the sequence length, the phylogenetic analysis of large genes and genomes becomes the major bottleneck. Thus, scaling the model checking verification process both in time and memory is one of our research priorities for the next parts. The difficulty of implementing a completely new and adapted software, the existence of few (but good and powerful) model checking tools such as NuSMV or PRISM, and the availability of its source code favors the reuse and customization of current tools instead of concerning with new implementations.

Part III

Model Checking Adapted to the Phylogenetic Context: Introducing Concurrency and Solving some Bottlenecks

From all the human genome, approximately the 99.9995% belongs to nuclear DNA whilst the remaining 0.0005% corresponds to the mitochondrial DNA used for the experimentation in Section 5.4. The cells of some salamanders may contain up to 40 times more DNA than those of humans [83]. Conversely, the number of available records in GenBank is currently doubling approximately every 35 months [23]. Hence, the extension and scalability of model checking for bigger alignments and phylogenies represents a dramatic question.

Here, we try to solve all these problems by applying an strategy of divide and conquer. This part contains two chapters. The first one is devoted to the presentation of technical and algorithmic solutions for the problem of model checking with large genomes. This includes the use of external databases for storing the sequences and the distribution of model checking computations. Instead of saving the complete genome as atomic propositions on the phylogenetic tree, the states are labeled with pointers to the database, leading to a light memory-consuming version of the Kripke structure.

Additionally, distributed model checking is divided in two subsections, explaining both the structural partitioning of the tree and the slicing of the states with DNA as feasible options for scaling the system. These solutions are compatible and they can be applied together. The structural partitioning of the tree attacks the problem by grouping the nodes in chunks and sending the partial data structure and specifications to computational units. Conversely, the slicing of the states maintains the Kripke structure but generates multiple copies of the original tree, each one with a portion of the DNA.

After this, the second chapter faces the implementation of all these techniques and integrates them inside a workflow. We evaluate a set of back mutation properties over a real phylogeny with thousands of nodes, increasing the complexity with respect to previous experiments. The experimentation results show the feasibility of our approach, showing competitive time and memory requirements.

Chapter 7

Sliced and Distributed Model Checking

Nothing is particularly hard if you divide it into small jobs.
— Henry Ford

7.1. Introduction

As a prominent advantage, model checking allows us to uncouple software tools from the definition of properties and it also hides the underlying implementation technology. Besides, these properties can be exported and evaluated in other structures (i.e., trees or networks) so as to compare the results and define metrics. Nevertheless, the performance is penalized when scaling the system for large phylogenies and alignments (Section 5.4). The underlying problem of standard model checking is the great amount of phylogenetic data it has to deal with: the information associated to each node of the tree is strongly related to the DNA sequence of the specie (up to millions of nucleotides). Currently, close to 2×10^6 species have been cataloged [110], but genomes exhibit an enormous range of sizes and complexity, with *Homo sapiens* ranking at 3×10^9 base pairs.

In order to cope with this problem, two strategies are presented here. The first one consists of partitioning the graph structure into a set of subgraphs, each one representing a subproblem of verification so as to speed up the computation time and distribute the memory consumption. Two subtactics are considered depending on the division method: the partition of the tree into subtrees, each one managed by a different model checker (Section 7.3); or the slicing of the tree, each slice containing a copy of the original tree but only a portion of the DNA sequence (Section 7.3). The generic techniques based on distributed model checking were

presented in [31], but the inherent acyclicness and structure of trees facilitates the introduction of new simplifications and adaptations that has been barely developed in the community yet. We also present the novelty of sliced model checking as an adaptation to the context of phylogenetics.

The second strategy is based on uncoupling the DNA information of the phylogenetic tree and exporting the alignment to an external tool specialized in the management of large information systems, for example, a database (Section 7.4). This leads to a light tree structure labeled with pointers to the elements of the database that can be efficiently manipulated by current model checking tools. The database strategy is firstly introduced here for the domain of phylogenetic model checking.

In this chapter, we explain the best of the two worlds. We integrate these methodologies (distribution and databases) and show that a combination of both strategies allows us to work with real phylogenies. The introduction of all these methodologies need the presentation of the sequential model checking algorithms for the contextualization.

7.2. Model Checking Algorithms

Current model checking tools load the model and the temporal logic formulas, evaluate them and compute a counterexample if so required [63, 13]. It is possible to reuse known properties and even results, if these are available. Evaluation results may discover new meaningful information that will be reused in subsequent refinements of the phylogenetic tree.

Verification of temporal formulas is formalized under the framework of set theory; we follow this convention throughout the section. In fact, the traditional model checking algorithm is usually presented as a recursive function which computes the set of states satisfying a CTL formula in a Kripke structure (Algorithm 1).

In order to evaluate the temporal operators $\mathbf{EG}(\psi)$ and $\mathbf{E}[\psi_1 \mathbf{U} \psi_2]$, the greatest and least fixed points are computed, respectively. Both fixed point sets can be obtained as the result of a breadth-first search (Algorithm 2). In particular, the call $\mathit{fixedpoint}(M, Sat(M, \psi), \emptyset, S)$ produces the greatest fixed point, and $\mathit{fixedpoint}(M, Sat(M, \psi_2), Sat(M, \psi_1), \emptyset)$ produces the least fixed point.

The time complexity of verifying a CTL formula ϕ against a Kripke structure is linear in $|\phi|$ and the size of the Kripke structure, with $|S|$ the number of states, $|R|$ the number of transitions and $|\phi|$ the number of logical connectives and temporal operators of the formula [63, 149]. More generally, the complexity is $\Theta((|S|+|R|)*|\phi|)$. The decomposition of a specification for the verification of the integral parts favors the distribution of the computations in next generation algorithms.

Algorithm 1 Algorithm $Sat(M, \phi)$

Require: $M = (S, S_0, R, L, AP)$ is a Kripke structure
Require: ϕ is a CTL formula
Ensure: A subset of states of S that satisfies ϕ
  if $\phi \equiv \top$ **then return** S {Set of states from the Kripke structure M}
  else if $\phi \equiv p \in AP$ **return** $\{s : p \in L(s)\}$
  else if $\phi \equiv \neg\psi$ **return** $S \setminus Sat(M, \psi)$
  else if $\phi \equiv \psi_1 \vee \psi_2$ **return** $Sat(M, \psi_1) \cup Sat(M, \psi_2)$
  else if $\phi \equiv \mathbf{EX}(\psi)$ **return** $\{s : (s, s') \in R, s' \in Sat(M, \psi)\}$
  else if $\phi \equiv \mathbf{EG}(\psi)$ **return** *fixedpoint* $(M, Sat(M, \psi), \emptyset, S)$
  else if $\phi \equiv \mathbf{E}[\psi_1 \mathbf{U} \psi_2]$ **return** *fixedpoint* $(M, Sat(M, \psi_2), Sat(M, \psi_1), \emptyset)$
  end if

Algorithm 2 Algorithm *fixedpoint* $(M, Sat(M, \phi), Sat(M, \psi), Init)$

Require: $M = (S, S_0, R, L, AP)$ is a Kripke structure
Require: $Sat(M, \psi)$ and *Init* are the sets of initial states (returned by calls to Sat algorithm) and $Sat(M, \phi)$ is the set of final states
Ensure: A set of states that represents paths going from *Init* (or $Sat(M, \psi)$) to $Sat(M, \phi)$
  $New \leftarrow Init$
  repeat
    $Old \leftarrow New$
    $New \leftarrow Sat(M, \psi) \cup (Sat(M, \phi) \cap \{s : (s, s') \in R, s' \in Old\})$
  until $New = Old$
  return New

7.3. Distributed Model Checking

State of the Art

Sometimes, the storage in local memory of the phylogenetic model together with the atomic propositions (both initial and new discovered properties) leads to a memory bottleneck in the model checker tool. It appears when characterizing taxa from phylogenetic trees as states in Kripke structures, that is, when labeling states of the transition system with long DNA or protein sequences (Section 5.4). In addition, the characterization of properties with temporal logics produce large sets of atomic propositions to be verified sparsely.

Thus, memory usage arises as a major limiting factor in the analysis of complex systems, often in association with vast state spaces, and therefore with long execution times. In this regard, the conventional monolithic techniques conceived

for improving the performance in industrial model checking applications fail to manipulate the phylogenetic tree in an efficient way due to the inherent features of the biological data, mainly the huge ratio of labels per state (i.e., DNA sequence). This problem persists in spite of the many methods which have been devised to scale model checking procedures. Of these, symbolic model checking is perhaps the most widespread [99, 85]. There exist several general-purpose memory techniques that can be applied to alleviate the problem of memory footprint, such as abstractions, partial reductions and symmetries (for a review, see e.g. [63, 31]).

Current efforts in this area revolve around two main topics. Firstly, compositional reasoning [47], itself a classic approach based on the verification of local properties associated to each of a collection of "components" (e.g., codons and genes in the context of biological sequences), proceeding incrementally to infer global properties of the system through a bottom-up strategy. Chief in importance among these techniques is the assume-guarantee paradigm [135], which operates by establishing a collection of assumptions about the environment of a component and verifying the latter subject to the former. Alternative approaches are exemplified by [78], where they focus the trouble from the point of view of temporal logic formula decomposition. However, all these methods are ineffective when applied automatically in isolation to tightly coupled systems made up from highly interdependent components [48]. Only a few theoretical works have been published in relation to the complexity and advantages of compositional model checking over these simple structures [39].

Secondly, we have methods that exploit the explosive availability of multicore (shared-memory) computers [96], fast interconnection networks [165] or MapReduce in clusters [22]. Generally, these operate by partitioning the system as defined by the Kripke structure and distributing the chunks among available computing units (both storage of the partial Kripke structure and computation of satisfiability of logic formulas [84, 30]). These would be applicable here, yet by themselves they are ineffective, as they address the size of the structure in number of states and not the complexity of each state, which is the other limiting factor in phylogenetic model checking. An approximation to this last group of methods is considered in this section, but adapting them to the special context of trees. We also give the notions of sliced model checking, an innovation that focuses on the complexity of the atomic propositions for the tree states and complements the partition and distribution of the phylogenetic tree in subtrees.

Alternatively, another strategy for phylogenetic model checking would consist of uncoupling atomic propositions from each state of the transition system and a) distribute them among several instances of a model checker tool (Section 7.3), or b) store the atomic propositions in an external database (Section 7.4).

Structural Partitioning of the Kripke Structure

Model checking based on distributed Kripke structures attempts to improve the performance by partitioning the graph into smaller subgraphs and disseminating the chunks among available computing units (both the storage of the partial Kripke structure and the computation of sets satisfying logic formulas [30]). These methods attack the size of the structure (number of states) and not the complexity of each state, which is the other limiting factor in phylogenetic model checking. The complexity of the states will be solved by sliced model checking, but the division of the Kripke structure in subgraphs is still valid here.

Recently, model checking over tree-like data structures is gaining interest. Some application papers motivate the benefits of CTL model checking in tree-like data structures such as XML [2]. In the context of distributed model checking, the advantages of model checking over trees is more evident: the verification and communication process between chunks is simplified due to the inherent acyclicness of trees with respect to more generic data structures. However, only a few theoretical works have been published in relation to the complexity and advantages of compositional model checking over these simple structures [39].

Here, our contribution consists of showing how the division in subtrees (or clades) can guide the distribution and parallel computation of verifications in tree-like Kripke structures. For each division of a balanced binary tree, the state space is reduced by half and only an interface node is added for the communication with each subtree. Given the tree root, s_0, we verify the property for each of the direct subtrees and operate with the boolean results according to the logical quantifiers. In the case of a generic formula ϕ written in CTL, the equivalence is supported by the following recursive expansion law of the path operators (**EX**, **EG**, **E_U_**) [13]:

$$M, s_0 \models \mathbf{EX}\phi \equiv \bigvee_{s_i \in successors(s_0)} M_i, s_i \models \phi$$

$$M, s_0 \models \mathbf{EG}\phi \equiv M_0, s_0 \models \phi \wedge \left(\bigvee_{s_i \in successors(s_0)} M_i, s_i \models \mathbf{EG}\phi \right)$$

$$\begin{aligned} M, s_0 \models \mathbf{E}(\phi_1 \mathbf{U} \phi_2) \equiv M_0, s_0 \quad &\models \quad \phi_2 \vee \\ (M_0, s_0 \quad &\models \quad \phi_1 \wedge \left(\bigvee_{s_i \in successors(s_0)} M_i, s_i \models \mathbf{E}(\phi_1 \mathbf{U} \phi_2) \right)) \end{aligned}$$

where M is the original tree with root s_0. M_i is the subtree with root $s_i \in successors(s_0)$. The degenerated tree M_0 only needs the node s_0. The formula

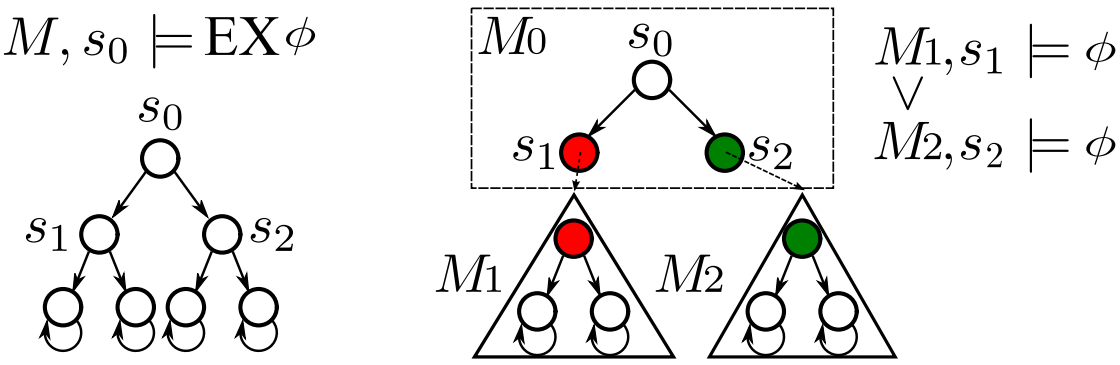

Figure 7.1: Distributed verification of **EX**ϕ through the parallel execution of ϕ in the direct subtrees.

can be unfolded indefinitely by means of **EX**. In this case, the set of successors s_i defines a border at a certain depth of the original tree: we must ensure that ϕ holds in s_0, in the subtree rooted by s_i and in a path between s_0 and s_i. The root s_i acts as an interface node for the communication process during the composition of the partial results. Figure 7.1 illustrates this technique.

The number and size of the subtrees, and the appearance of short circuits during the composition of results will determine the performance of this approach. The detection of clades with conserved regions or characteristic SNP's are the kind of properties that benefit of this approach.

The implementation of the model checker is also important. For example, Figure 7.2 represents the time needed for the initialization of the NuSMV model checker given a phylogenetic tree labeled with simple identifiers (GenBank accession numbers [23]) that act as pointers to the records of a potential external database. If the tree is balanced, the initialization and verification of the subtrees is faster than the original one as the trend is quadratic with respect to the tree size. Furthermore, the memory consumption is linear with the number of nodes as depicted in Figure 7.3.

The decomposition of the phylogenetic formula drives the division of the tree in autonomous modules that can be evaluated concurrently in parallel machines or sequentially in a single computer, as long as they store the partial results for synthesizing the answer to the initial hypothesis. The quadratic trend in the initialization of NuSMV makes also profitable the sequential evaluation of a property in two subtrees (for balanced binary trees) rather than in the complete phylogeny, which shows connections with compositional model checking [47]. Due to the or-like logic for the evaluation of the property in the subtrees, all the counterexamples generated are needed as partial solutions for the integration of the overall explanation.

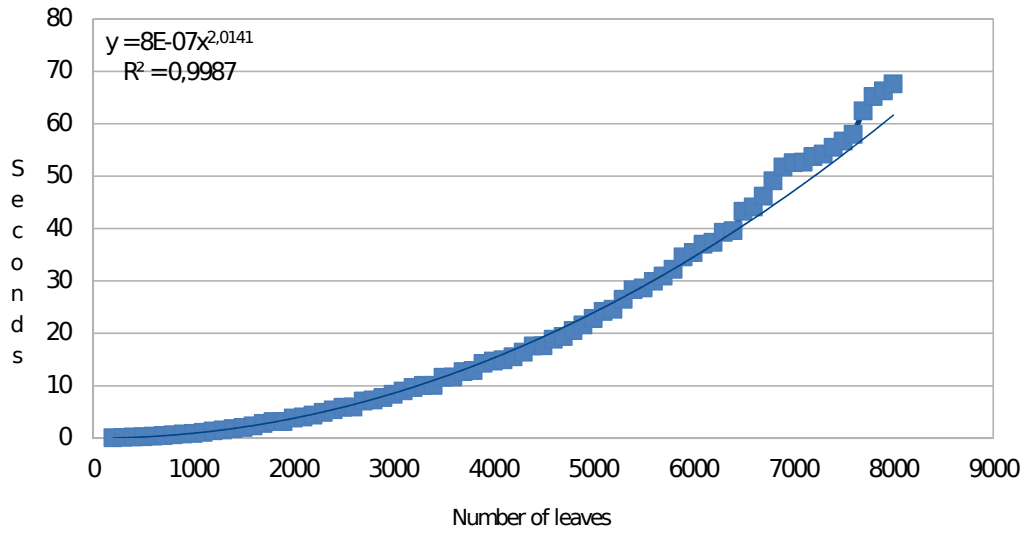

Figure 7.2: Time required in NuSMV for the initialization of a phylogenetic tree with GenBank identifiers in the nodes.

Thus, the distribution of the original state space in subgraphs is relevant for optimizing the analysis of phylogenies when the limiting factor is the number of leaves. The structure of the tree and a simple manipulation of the temporal logic formula grants the extraction of a valuable speed up with minor modifications of the model checking process. The technique proposed here is supplementary to the application of the state slicing of the Kripke structure introduced in the following paragraphs.

Slicing the States of the Kripke Structure

Phylogenetic model checking usually associates a complete DNA sequence per node (specie) of the tree. Sometimes the storage in local memory of the phylogenetic model together with the atomic propositions leads to a high memory consumption in the model checker tool. In order to solve it, the state slicing (also called sliced model checking) focuses on the state complexity of the Kripke structure by creating several copies of the original phylogenetic tree and verifying the subproperties in parallel, each slice labeled with a partial substring of the DNA. The slices may correspond to genes, codons or any entity with significance in the genome that motivates the local storage of near characters.

In the best case, the state slicing presented in this section can be efficiently

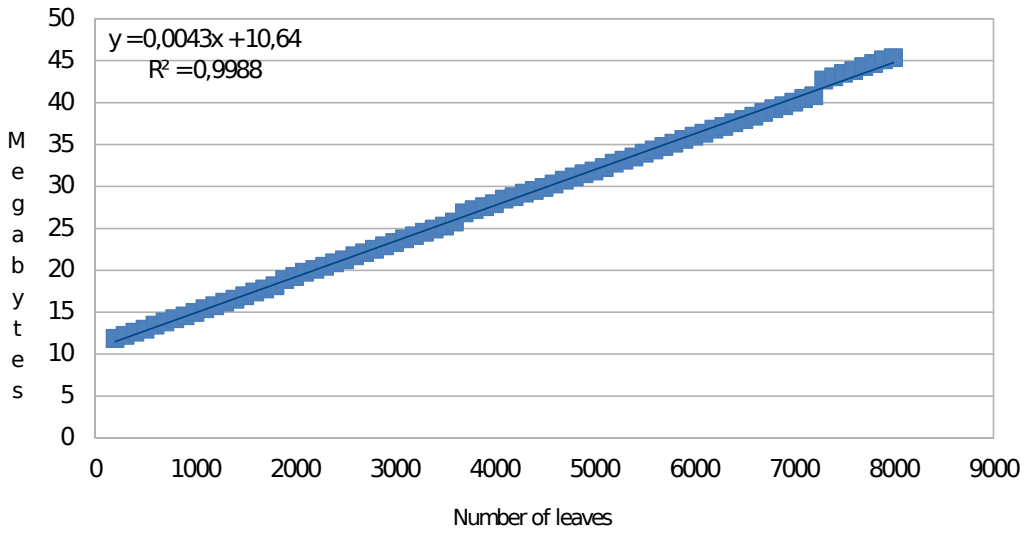

Figure 7.3: Memory required in NuSMV for the initialization of a phylogenetic tree with GenBank identifiers in the nodes.

applied to the verification of completely autonomous properties such as the detection of back mutations (Equation 4.20). Each slice verifies a subformula referred to a part of the DNA sequence, that is the only information stored in each state of the slice. These verifications can be done independently and so a computational speed up proportional to the number of slices can be obtained. That is, given an alignment of length l, each sequence is distributed in l independent slices where $\bigvee_{col\in\{1\ldots l\}} M, s_0 \models BM\,(col, \sigma)$ are verified asynchronously.

But the state slicing method can be applied into more intricate properties. Take into consideration the techniques of amino acid coevolution (Section 4.3). They have been recently used for the prediction of the three-dimensional structure and interrelations in the chain of proteins of the mitochondria respiratory pathway, which is essential for the cellular respiration and photosynthesis [173]. An intuitive cut of the mtDNA for that example matches with the genes coding those proteins. The inter-dependence correlation of amino acids from separate proteins has provided crucial indirect evidences about the connections among molecules. In fact, these signs may detect potential binding regions that work as interface zones between proteins and that are susceptible to any minimal change. Hence, the study of these regions requires the access to the gene information stored in disperse slices. Apart from the analysis of the character correlation in the alignment, the entanglement of the evolutionary drift for different inferred gene trees

should also show strong coevolution signs. The combination of topological (tree) and sequence properties gives rise to Section 4.4.

Coming back to the definition of the Kripke structure, it is common for the AP set of a system to encode a heterogeneous collection of properties, whose correlations can be arbitrary in both strength and complexity. Traditionally we focus on DNA, but the AP set is originally unstructured in the sense that different types of labels may be considered. Thus, the "normalization" of properties and variables along the graph turns out to be a critical issue.

Definition 5 (Homogeneous Kripke Structure)**.** *Let Σ be an alphabet. An homogeneous Kripke structure $M = (S, S_0, R, L, AP)$ over AP is a Kripke structure where,*

- $AP = \bigcup_{i=1}^{l} AP_i$ *is the set of atomic propositions,*
- $AP_i = \{seq[i] = \sigma \mid \sigma \in \Sigma\}$ *with seq an array of variables with type Σ,*
- $L : S \to AP_1 \times \ldots \times AP_l$*, where* $L(s) = (seq[1] = \sigma_1, \ldots, seq[l] = \sigma_l)$.

In Definition 3, the labeling function L tags each state of the Kripke structure with a unique tuple composed of several atomic propositions. Additionally, an extension to multiple labels per state ($L : S \to 2^{AP}$) can be considered. Note that the Cartesian product in the labeling function involves an implicit "and" operator; that is, the label states that a temporal formula is valid on the current state iff it asserts $[(seq[1] = \sigma_1) \wedge \ldots \wedge (seq[l] = \sigma_l)]$. Multilabels would be or-like though.

In opposition to traditional distributed model checking algorithms, which divide the system into disjoint Kripke substructures, we present a new slicing criterion. The notion of *AP normalization* in a homogeneous Kripke structure motivates the definition of a projection (reading) function that takes (cuts) a subset of the variables from the AP structure of state labels.

Definition 6 (Projection Function)**.** *A projection function with subindex $i \in \{1, \ldots, l\}$ over AP selects a subset of elements such that:*

- $proj_i : AP \to AP_i$ *with* $i \geq 1$ *such that* $proj_i(seq) = (seq[i] = \sigma)$.
- $proj_I : AP \to AP_{i_1} \times \ldots \times AP_{i_m}$ *such that* $proj_I(seq) = (seq[i_1] = \sigma_{i_1}, \ldots, seq[i_m] = \sigma_{i_m})$ *is a projection function with a set of indices* $I = \{i_1, i_2, \ldots, i_m\}$, $m \leq l$

The projection operator allows to slice the nodes of the Kripke structure instead of splitting the graph into regions. We obtain several lighter copies of the original one, where each slice of AP represents some characteristics of the initial system. Figure 7.4 describes graphically the concept of slicing. In conjunction, the set of slices works as a distributed repository of sequences.

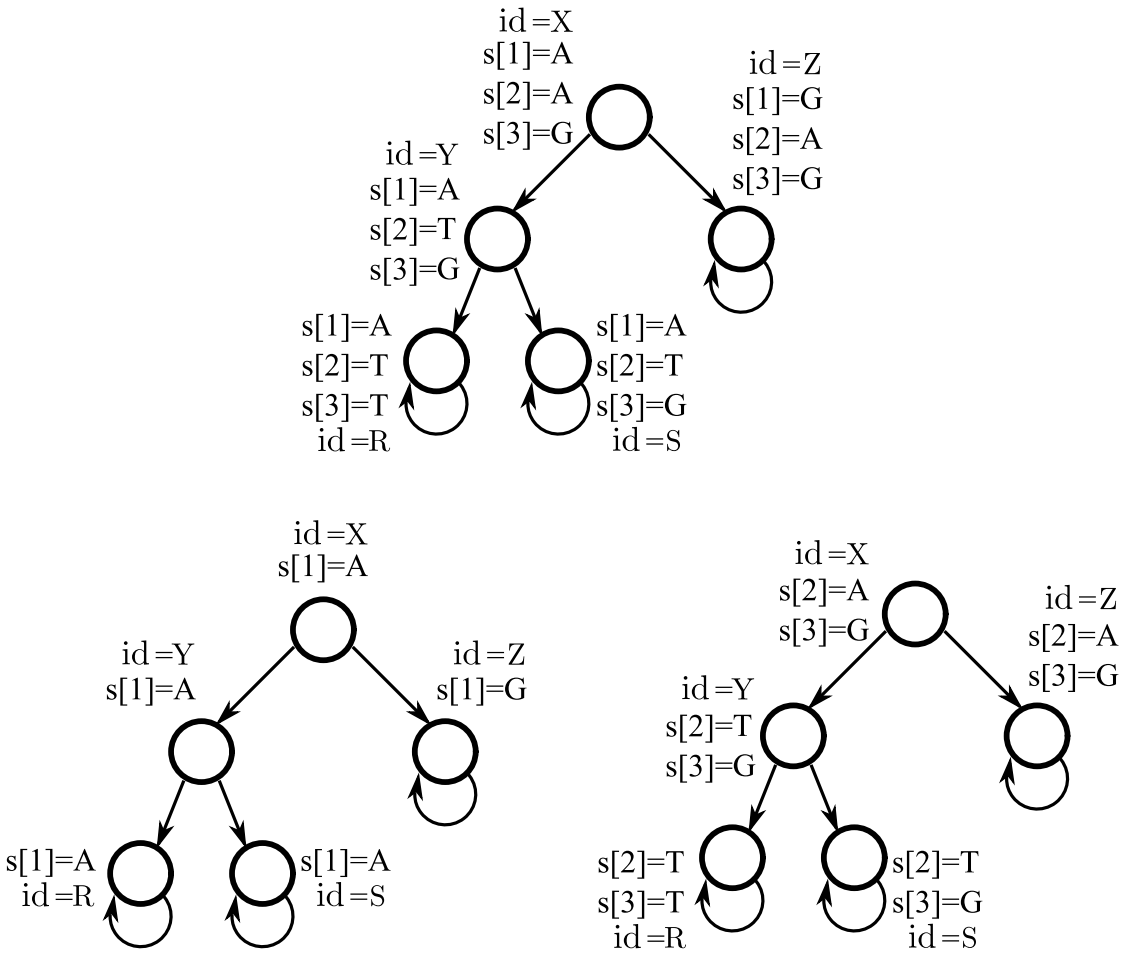

Figure 7.4: Division of the original Kripke structure into two slices.

Definition 7 (Sliced Kripke Structure)**.** *Let* $\{I_i \mid 1 \leq i \leq n, I_i \subseteq I\}$ *be a partition of the set of indices* $I = \{1, \ldots, l\}$*. We say that* $M = (S, S_0, R, L, AP)$ *can be obtained from the composition of* n *Kripke structures, named slices of* M*, each one defined as:*

- $M_i = (S, S_0, R, L_i, T_i)$, $\forall i \in \{1, \ldots, n\}$, *with* $T_i = \bigcup_{j \in I_i} AP_j$ *and*
- $L_i(s) = proj_{I_i}(seq)$, $\forall s \in S$ *with* $L(s) = seq$*, where*
- $L(s) = L_1(s) \cdot L_2(s) \cdot \ldots \cdot L_n(s)$, $\forall s \in S$*, with "*$\cdot$*" the operator function that concatenates tuples.*

Consider a generic phylogenetic property $\phi(p_1, p_2, \ldots, p_l)$ as input to the model checking tool, with $p_i \in AP_i = \{seq[i] = \sigma \,|\, \sigma \in \Sigma\}$ an atomic proposition that asks

for the tree nodes having the nucleotide σ in the i^{th} position of the DNA sequence. The classic model checking algorithm parses the formula and starts verifying the inner subformulas first. In the derivation tree of the CTL grammar for the given formula, we reach a propositional operator at some point of the recursion (boxed lines in Algorithm 3).

The verification of formulas with propositional operators, such as $\psi_1 \vee \psi_2$, begins with the computation of the satisfiability sets $Sat(M, \psi_j)$, $j = \{1, 2\}$. These sets allow us to distribute the computation in parallel, as long as we compose the partial results with a synchronized union. The support of ψ_j is $\|\psi_j\| = \{p_i \in AP_i \mid p_i$ or $\neg p_i$ appears in $\psi_j\}$. The verification of ψ_j is mapped to a remote model checking tool that has a copy of the phylogenetic tree labeled with the slice $T_I = \bigcup_{i \in I} AP_i$, with $I \subseteq \{1 \ldots l\}$ a set of index, $l = length(DNA)$ and $AP = \bigcup_{i=1}^{l} AP_i$ in this case. The remote model checker also executes Algorithm 3 but with the set of atomic propositions belonging to $\|\psi_j\| \subseteq T_{I_j}$. In order to obtain a perfect distribution, $\bigcap_{j=\{1,2\}} \|\psi_j\| = \emptyset$. The notation $par(T_{I_j}, Sat(M, \psi_j))$ means that $Sat(M, \psi_j)$ is computed in parallel in the remote model checker T_{I_j} associated to the slice containing $\|\psi_j\|$. L_{T_I} is the labeling function of the states of that slice. The computation of the CTL paths is denoted with the fixed point algorithms $\mu Z / \upsilon Z$. The number of parallel instances of the model checking algorithm is determined by the granularity in the slice of the AP set, but only one of those (i.e., the "coordinator") receives the full phylogenetic property as input.

Algorithm 3 Algorithm $Sat\,(M, \phi)$

Require: $M = (S, S_0, R, L, AP)$ is a Kripke structure
Require: $\phi(p_1, p_2, \ldots, p_l)$ is a CTL formula
Ensure: A subset of states of S that satisfies ϕ
if $\phi \equiv \top$ **return** S {Set of states from the Kripke structure M}
else if $\boxed{\phi \equiv p_i \in AP}$ **return** $par(T_I, \{s : p_i \in L_{T_I}(s)\})$ with $i \in I$
else if $\boxed{\phi \equiv \neg\psi}$ **return** $S \setminus par(T_I, Sat(M, \psi))$
else if $\boxed{\phi \equiv \psi_1 \vee \psi_2}$ **return** $par(T_{I_1}, Sat(M, \psi_1)) \cup par(R_{I_2}, Sat(M, \psi_2))$
else if $\phi \equiv \mathbf{EX}\,(\psi)$ **return** $\{s : (s, s') \in R, s' \in Sat\,(M, \psi)\}$
else if $\phi \equiv \mathbf{EG}\,(\psi)$ **return** $\upsilon Z.\,(Sat\,(M, \psi) \cap \mathbf{EX}\,(Z))$
else if $\phi \equiv \mathbf{E}\,[\psi_1 \mathbf{U} \psi_2]$ **return**
$\mu Z.(\,\boxed{par(T_{I_1}, Sat(M, \psi_1)) \cup par(T_{I_2}, Sat(M, \psi_2))} \cap \mathbf{EX}\,(Z))$
end if

By now, we consider the access to the atomic propositions transparent to the underlying technology. However, we must advance that this is usually the most time-consuming part during the experimentations in Chapter 8. This fact moti-

vates the introduction of information systems optimized for the management of huge amounts of phylogenetic data in the next section.

Finally, the size of the slices depends on the target DNA regions we desire to analyze (i.e., single nucleotides, genes, chromosomes, ...), the kind of properties we want to verify and the hardware requirements we have. A high number of slices will provide of a better level of parallelism and low hardware requirements (CPU, memory), but it will be limited by the potential appearance of bottlenecks during the composition of results. The detection of back mutations is a degenerated example of this case as it only needs a tree labeled with a single nucleotide. More intricate properties such as the detection of base correlations in a respiratory pathway advices the introduction of sliced model checking for analyzing each protein gene in isolation.

7.4. Model Checking Using Databases

The second step of our approach consists of uncoupling the atomic propositions from the model checker. The use of external databases as a repository of biological sequences alleviates the memory explosion problem when the local storage of trees with partial DNA is not enough. Moreover, the database manager simplifies the interface to the DNA data because it usually allows concurrent queries and it hides the internal synchronization and data structures. In a general sense, trees are labeled with pointers to DNA sequences stored in an external server.

As the AP set is now structured as consequence of Section 7.3, the atomic propositions can be purely mapped into a relational database. For example, a generic sequence $seq = \sigma_1\sigma_2 \ldots \sigma_l$ is composed of a set of elements $seq[i] = \sigma_i$ (in terms of AP notation) whose σ_i value is univocally assigned to the associated column $seq[i]$ in a relational table. The DNA alignment is stored in a single relational table with a row per sequence. Each row has two important fields: an identifier (i.e., the GenBank accession [23]) plus the plain string of nucleotides.

In fact, the tables of relational databases can be seen as matrices that we can slice by row (number of taxa) or by column (dividing the DNA in substrings) in case of the atomic propositions were stored in separated columns. These (sub)tables would be stored or replicated in separated hard discs, servers or cluster of nodes so as to allow parallel access to the DNA data and to improve the communication bandwidth between the database and the model checker. In addition, recent versions of SQL servers support multicore CPU's, which improves the time response when attacking the server with several queries. This approach allows to scale in memory and speeds up the system. A centralized database is source of bottlenecks during the concurrent access of multiple model checker clients: dis-

tributed database managers can be used in order to optimize the performance of the transactions.

Then, an external database manager is required for the storage of the AP set. In this way, the model checker algorithm will reduce the memory consumption by only saving a set of references (pointers) to the atomic propositions of each state plus a boolean variable with the truth value of the current proposition. In a hybrid system that combines slicing techniques and databases, the evaluation of a CTL formula consists of translating the last recursion call (the computation of the result set $\{s \in S \mid p \in L(s)\}$ in Algorithm 3) into a database transaction that gets the array $(s[1] = \sigma_1, s[2] = \sigma_2, \ldots, s[l] = \sigma_l)$ from the appropriate record of the relational table. Each character σ_i of a sequence can be stored in up to a independent column of the relational table: atoms of the type $(s[i] = \sigma_i)$ are potentially read in concurrency and the tuple is rebuilt as an "intersection" set (logical conjunction) of partial results. The projection function (Definition 6) is identifiable to the selection of a column subset. The slicing of the alignment is relegated to the vertical or horizontal partition of the database relational table.

Algorithm 4 is an adaptation of Algorithm 3 where $par(database, \cdot)$ is a parallel query to the database asking for the set of id's satisfying the atomic propositions (in case of the database front-end accepts multiple queries simultaneously). It substitutes the message $par(T_I, \cdot)$ requesting the data to the rest of slices in Algorithm 3. Thanks to the centralization of the sequences in an external repository, the parallel evaluations $par(T_I, \cdot)$ are replaced by $par(\cdot)$ for the rest of situations, which can be executed in a multicore CPU using threads. Therefore, Algorithm 4 requires a shared memory machine for its execution and not a pool of sliced model checking workers.

Algorithm 4 Algorithm $Sat\,(M, \phi)$

Require: $M = (S, S_0, R, L, AP)$ is a Kripke structure
Require: $\phi(p_1, p_2, \ldots, p_l)$ is a CTL formula
Ensure: A subset of states of S that satisfies ϕ
if $\phi \equiv \top$ **return** S {Set of states from the Kripke structure M}
else if $\boxed{\phi \equiv p_i \in AP}$ **return** $par(database, \{s : p_i \in L_{R_I}(s)\})$ with $i \in I$
else if $\boxed{\phi \equiv \neg\psi}$ **return** $S \setminus par(Sat(M, \psi))$
else if $\boxed{\phi \equiv \psi_1 \vee \psi_2}$ **return** $par(Sat(M, \psi_1)) \cup par(Sat(M, \psi_2))$
else if $\phi \equiv \mathbf{EX}\,(\psi)$ **return** $\{s : (s, s') \in R, s' \in Sat\,(M, \psi)\}$
else if $\phi \equiv \mathbf{EG}\,(\psi)$ **return** $\upsilon Z.\,(Sat\,(M, \psi) \cap \mathbf{EX}\,(Z))$
else if $\phi \equiv \mathbf{E}\,[\psi_1 \mathbf{U} \psi_2]$ **return**
$\mu Z.(\boxed{par(Sat(M, \psi_1)) \cup par(Sat(M, \psi_2))} \cap \mathbf{EX}\,(Z))$
end if

Finally, relational databases can execute internally the model checking algorithms for the evaluation of CTL formulas. Procedural language extensions for SQL such as PL-SQL support this possibility [151]. The main advantage is that they avoid the exportation of DNA data from the database to a model checking tool and the associated bandwidth bottleneck. The lack of symbolic data structures that are habitual in model checking tools such as ORBDDs [33] is resolved by the manipulation of the data with the inherent adaptations for databases. An evaluation of SQL model checkers for phylogenetic data will comprise our future work. As far as we know, this is a novel approach because model checking was mainly used for verifying database correctness [43] and querying temporal databases [60, 44], but databases has never been used as an external repository. Furthermore, relational databases cannot only store phylogenetic data, but also partial verification results or counterexamples.

Chapter 8

Workflow

The White Rabbit put on his spectacles.
"Where shall I begin, please your Majesty?" he asked.
"Begin at the beginning," the King said gravely, "and go on till you come to the end: then stop."

— Lewis Carroll, *Alice in Wonderland*

8.1. Introduction

In this chapter, we describe the implementation of the different methods presented previously. We compose them in a workflow by integrating the modules encapsulating each technique. Here, we detail the internal characteristics of our framework as well as the performance results with large phylogenetic data. The combination of all these optimizations facilitate the efficient management of big phylogenies by current model checking tools. In particular, we develop an hybrid system: we code a multi-threading version of Algorithm 4 and use an external database for saving the sequences. To this end, we modify the model checker NuSMV and send queries to a MySQL server.

This part is thus divided in three points. Firstly, Section 8.2 draws a panoramic overview of the pipeline, details every component and how they are connected through its interface. Next, Section 8.3 gives brief notions about the experimental results and performance of model checking with these new approximations. We search for the presence of back mutations over ZARAMIT [27] (a real phylogenetic tree) and evaluate the temporal costs. Finally, Chapter 9 summarizes the conclusions of this part and suggests the future work.

8.2. Description

Figure 8.1 represents a graphical description of our workflow. The central core of our approach is the model checker package NuSMV v2.5.4 [45], which is a well-known public software. It is surrounded by a set of modules and tools that accommodate the phylogenetic data for each step of the verification. The system also counts with an external MySQL server v5.5.24-7 for saving the atomic propositions (i.e., the DNA alignment).

The use of databases simplifies the coding of the sliced model checking algorithms. Here, we present an hybrid system where the new sliced model checking algorithms are highly coupled with the database (Algorithm 4). The sliced model checking algorithms are implemented in two steps. First of all, an external script analyzes the specification of the phylogenetic properties that the system must evaluate, and subsequently accesses in parallel to the atomic propositions (DNA sequences) stored in the database. After the extraction of the GenBank identifiers satisfying those AP, these sets are converted into the internal data representation of NuSMV (i.e., ORBDDs). Finally, a multi-threading version of Algorithm 4 collects the results and compounds the CTL paths for evaluating the truth of the property.

In sum, our framework is thus divided in three important modules. First of all, the *loader* consists of a BioPython script that creates and initializes the relational database with the DNA sequences. It is executed only once during the initialization phase for loading the sequences into the database. The database habitually remains constant for the rest of the verification process, but it can be updated periodically with the addition of new aligned strings. By default, we use BioSQL, a shared database schema for storing sequences in SQL servers that is supported by several script languages [121].

Secondly, the *property transformer* is a BioPython script that pre-processes the phylogenetic formulas and rewrite them in terms of GenBank identifiers. Every time the user asks for the states satisfying a pattern $seq[i] = \sigma$ in the DNA, the script retrieves the set of identifiers of those species that satisfy the formula in the database. This script translates the phylogenetic properties expressed in terms of sequences and nucleotides into phylogenetic properties written in terms of taxon identifiers. For example, the following formula reinterprets the back mutation property (Equation 4.20):

$$BM \equiv (id \in seq_{i\sigma}) \wedge \mathbf{EF}\left[id \notin seq_{i\sigma} \wedge \mathbf{EF}\left(id \in seq_{i\sigma}\right)\right] \tag{8.1}$$

where id is the identifier of the state reached by each temporal operator **EF** (at the beginning, id corresponds to the identifier of the initial state). The set $seq_{i\sigma} = \{id_1, \ldots, id_n\}$ contains the identifiers of the set of states satisfying the

property $seq[i] = \sigma$ in the external database. The states of the phylogenetic tree are labeled with these identifiers.

The script is optimized for sending concurrent queries to the database in order to take advantage of current multicore servers, each question demanding the set of identifiers satisfying $seq[i] = \sigma$. As result, it outputs an specification file with the original phylogenetic properties expressed in terms of GenBank identifiers. In addition, it generates a set of auxiliary files corresponding to the projections of the formula in the sliced model checking methodology. These auxiliary files, one per $seq[i] = \sigma$, store the set of identifiers $seq_{i\sigma}$ satisfying an atomic proposition of the original formula.

Next, the *NuSMV transformer* is a BioPerl script that translates the phylogenetic tree into the NuSMV syntax (Section 5.3) and labels the states with the associated identifiers of the database. The script is easily extended in order to divide automatically the original tree in multiple subtrees as a part of the implementation of partitioned model checking, but the transformation of the equations together with the composition of counterexamples and results has to be realized manually at the moment. The subtrees can be stored either in separated NuSMV model files in order to evaluate the specifications in concurrency; or in individual modules inside the same file for a sequential evaluation and composition of the results. The cost in time/memory increases fast with the size of the tree and therefore a sequential evaluation of the specifications in the subtrees also becomes profitable in terms of improving the efficiency. The CTL parser in NuSMV may assist to the unfolding of the temporal logic formulas according to the explanation in Section 7.3.

Once the phylogenetic tree is translated into the model checker syntax and the properties are translated and projected according to the slicing criterion, the workflow starts the verification process. To this end, a set of independent instances of NuSMV are launched. Each one works with a copy of the Kripke structure and a DNA projection, that is, the auxiliary files generated by the *property transformer*. Their mission consists of converting the set of identifiers of $seq[i] = \sigma$ stored in the auxiliary files into the internal notation of NuSMV (an ORBDD). At the end of the process, a multi-threading version of the model checker collects the ORBDDs for the atomic propositions and reconstructs the CTL paths. The final speed up depends on the distribution and the number of formulas and slices. The results of the verification process offer an important feedback for the refinement of the original tree and the biological assumptions.

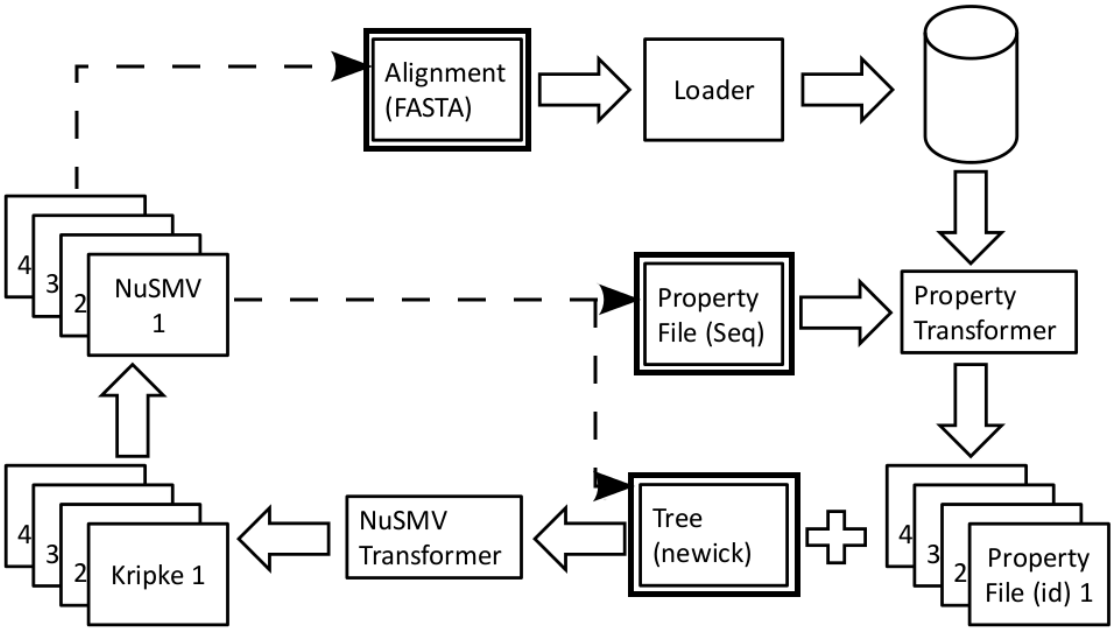

Figure 8.1: Workflow diagram with the alignment, phylogenetic tree and properties as input.

8.3. Experimental Results

The data set of our experimentation comprises the ZARAMIT tree [27], which has been reconstructed from 7390 aligned sequences of Human mitochondrial DNA with 16,569 base pairs. In total, the number of internal plus terminal nodes is 14512. Thanks to the integration with databases, the memory consumption of the model checker is reduced: the representation of the ZARAMIT tree consumes around 50 MB. Besides a light consumption of memory requirements, the use of references to an external database reduces the initialization time of NuSMV. Figure 7.2 shows that our architecture outperforms previous works in [138]: we spend less time and memory resources for the initialization of the model checking tool with the ZARAMIT tree than for the initialization of a protein tree with 2000 tips and sequences of 500 aminoacids. Taking advantage of these optimizations, we have employed the ZARAMIT tree for the evaluation of 25 back mutations formulas using our framework.

From the point of view of performance, the database is the most important point of our workflow because it connects the model checker with the DNA sequences. The database server runs in a desktop workstation (AMD Opteron @3GHz, 4GB RAM, Debian Linux). Due to the alignment size (around 350 MB for 7390 leafs plus ancestors) and the hardware available, we work with a single

instance of MySQL server. The database loader spends less than 2 minutes for inserting the alignment in the database at the initialization point, but it remains constant for all the verifications. The retrieval of atomic propositions from the database lasts 7.2s for 25 concurrent SQL queries (44.6s if we serialize them). Each SQL query asks for the set of identifiers associated to the DNA in each one of the 25 back mutation properties.

A second desktop workstation (Intel Core 2 Duo E6750 @ 2.66GHz, 8GB RAM, Debian Linux) is devoted to the execution of the rest of the workflow. This comprises the translation of the phylogeny into the NuSMV syntax (*NuSMV transformer*), the conversion of the GenBank identifiers to the internal data structure of NuSMV (auxiliary files of the *property transformer*) and the execution of the multi-threading algorithm that accomplishes the evaluation. By now, the back mutation properties are evaluated over the whole phylogeny. The application of partitioned Kripke structures (i.e., the verification of the back mutation property independently for each direct subtree of the phylogeny) will divide the time by more than a factor of 2 for binary trees.

However, we have detected that the internal representation of the Kripke structure in NuSMV penalizes the access and manipulation of big sets during the model checking process. Given a set of GenBank identifiers whose DNA sequences satisfy the atomic propositions $seq[i] = \sigma$, the time required for the extraction of the tree states with those id and their exportation to a temporal ORBDD file raises up to 1m20s for each auxiliary file of the *property transformer*. Nevertheless, this latency can be alleviated because the exportation of every set of GenBank identifiers to a ORBDD file is executed in parallel. Conversely, the integration of partial results (read of a ORBDD file) and the computation of CTL paths for the verification of a back mutation property only lasts 1m 2s for the multi-threading algorithm. All these temporal costs embeds the initialization time of NuSMV for the ZARAMIT tree (around 57s).

In sum, the initialization costs of NuSMV raises up to 55s-1m for phylogenetic trees with thousands of species and nodes labeled with references to the database. Besides, the framework offers a competent performance for evaluating phylogenetic specifications, such as the detection of back mutations. The first step, the extraction of big sets of states satisfying a simple atomic proposition, lasts around 1m27s (SQL query + exportation to a ORBDD). The extraction of multiple atomic proposition can be effectuated in parallel, with the number of cores the only limitation to this part. Later, the integration of the ORBDD results and the computation of the CTL paths takes 1m2s, leading to 2m30s for a complete verification of a single back mutation property. The parallel evaluation of the back mutation property in the direct subtrees of ZARAMIT will reduce the verification time by more than a half. Longer unexplored phylogenetic properties can take advantage of multi-threading

as the CTL verification cost depends of the formula length [13].

Chapter 9

Conclusions

The only place success comes before work is in the dictionary.

— Vince Lombardi

The use of monolithic model checking for the verification of huge phylogenetic data is unfeasible because of the high requirements in time and memory. The aim of this part has been to illustrate two different group of approaches for the memory scalability problem in model checking with phylogenetic data. Firstly, this part presented a couple of fully distributed approaches to model checking based on "state slicing" and "graph partition" techniques.

Assuming that the biological data (atomic propositions) can be fragmented into minimal meaningful blocks that fit into the model checker local memory, each thread (or remote task) executes the verification process over one slice (projection) from the original data while sharing a common Kripke structure skeleton. Then, the final result will be composed from the partial results. The computation of sets of states satisfying a certain atomic proposition (owning a portion of DNA), is one of the most time consuming parts. Hence, the global performance of our system benefits of this approach. Due to the emerging abundance of multicore computers, we introduced algorithms for shared-memory architectures, although they can be intuitively adapted for computer networks as well.

Additionally, the inherent acyclicness of trees facilitates the implementation, synchronization and composition of the verification results using graph partition techniques. Besides sliced model checking, we show how to distribute subtrees and take advantage of this data structure for the graph partition. Thanks to the logical expansion laws, we can verify the same phylogenetic property in each subtree and compose the results using first order logics. As the initialization time of the model checker usually depends of the number of nodes (our experimentations

with NuSMV shows that this dependence is quadratic), the number and size of the subtrees of the distribution process has an important impact in the total speed up.

Our second proposal, which is oriented to database techniques, takes advantage of the separation of the data from the execution flow. The use of an external and specialized database management system enables scalability in data storage and the introduction of optimizations during the retrieval of huge amounts of biological information. The final speed up will depend on the architecture and efficiency of the specific database management system. We remark that strategies for the dynamic update of the atomic propositions set should be considered. Furthermore, relational databases can emulate the model checking algorithms using PL-SQL.

Also, hybrid systems (distributed execution flow plus an external database system) combines the advantages of both previous proposals and it offers a promising framework. The viability (implementation and experimental results) of each one of these approaches over a real platform was showed in previous chapters.

Finally, the integration of all this approaches helped us to execute the verification of properties in a real phylogenetic tree. The execution of model checking in the ZARAMIT tree has outperformed the results of monolithic model checking in Section 5.4 [138]. For instance, the initialization time of the model checker raises up to 1m. The verification of a back mutation property lasts less than 2m30s if we discount the inevitable initialization cost of NuSMV. Although we use a specific architecture and technology, it should be noted that we can consider alternative open source model checkers like SPIN, standard interprocess communication languages such as MPI, or clusters of databases. It is also feasible to develop a specialized model checker to take advantage of the particular properties of phylogenetic model checking.

Part IV

Quantitative Extensions in the Analysis of Phylogenies: Generalities and Introduction of Time

Chapter 10

Introducing Time, Probabilities and Quantification on Phylogenetic Properties

The Doctor: [about the nature of time] People don't understand time. It's not what you think it is.
Sally: Then what is it?
The Doctor: Complicated.
Sally: Tell me.
The Doctor: Very complicated.
Sally: I'm clever, and I'm listening, and don't patronize me because people have died and I'm not happy. Tell me.
The Doctor: People assume that time is a strict progression of cause to effect, but, actually, from a non-linear, non-subjective viewpoint, it's more like a big ball of wibbly-wobbly ... timey-wimey ... stuff.

— Doctor Who, *Blink (Episode 10, Season 3)*

10.1. Introduction

In the preceding chapters we have introduced a formal framework for the analysis of qualitative properties concerning the discrete process of evolution. That work has bridged a conceptual link between the model checking framework and the process of verification of phylogenetic hypothesis. In other words, the phylogenetic trees are interpreted as transition systems, the biological specifications are expressed as formulas of temporal logic, and both of them are introduced as input in a model checking tool for their validation.

The boolean properties considered until now are only refereed to the *basic* information contained in a phylogenetic tree: a) the main structure of the tree, and b) the DNA information associated to the nodes (states) of the phylogeny. These properties, or a combination of both, can be expressed with more or less effort in a conventional temporal logic such as LTL or CTL, and, thus, we call these properties qualitative. In Chapter 4 of this thesis, an extensive (although not exhaustive) table of qualitative phylogenetic properties has been presented and defined in the context of phylogenetic analysis.

Nonetheless, those standard qualitative logics become insufficient for some usual practices of phylogenetic analysis since they don't allow the inclusion of quantitative information in the specifications and models. The labeling of the phylogenetic tree sometimes includes extra information beyond the original propositional information of the states [181, 111]. These numerical annotations of the tree depends on each particular study, but most of them share common roots like the inclusion of time and probabilities in the specifications. Take for example the properties defined in Table 10.1 of Section 10.4, where we have organized the biological hypothesis according to this criterion.

For instance, one of the more frequent topics is the inspection of the divergence dates between species or populations [76, 36, 101, 166] (first line, second column of Table 10.1). In this toy sample, with $distance(pongo, homo) > 10$ we want to test if the pongos and homos diverged more than 5 million years ago, i.e., the common ancestor of both families lived in that epoch (if the branches Hominidae-Homo and Hominidae-Pongo are symmetric, the aggregated distance in both paths is 10). Taking the ancestral node Hominidae as the initial state for the evaluation of the following formula, the distance between pongos and homos can be expressed with CTL-like path operators embedding the explicit time:

$$\mathbf{EF}_{\leq 5}(Pongo) \wedge \mathbf{EF}_{\leq 5}(Homo) \tag{10.1}$$

Obviously, these kind of questions cannot be resolved with standard CTL queries because of the lack of explicit temporal information in the phylogenies considered until now. Figure 10.1 clearly illustrates this situation: the internal branches of the tree are not proportional to the flow of time. The phylogeny and the logical operators should be updated for considering explicit time (either discrete or continuous).

Thus, this property requires the addition of time to the branches. The inclusion of temporal distances is also necessary for the conjecture and discovery of dates marking the migratory movements or their consequences in other hypothesis: the separation of the genome in lineages and ethnic groups (Tibetan people [19, 20, 25, 177]), the apparition of chronic and endemic diseases [57, 51, 129] or phenotypic adaptations (lactose [160, 163, 97, 92] and alcohol tolerance [72, 178]).

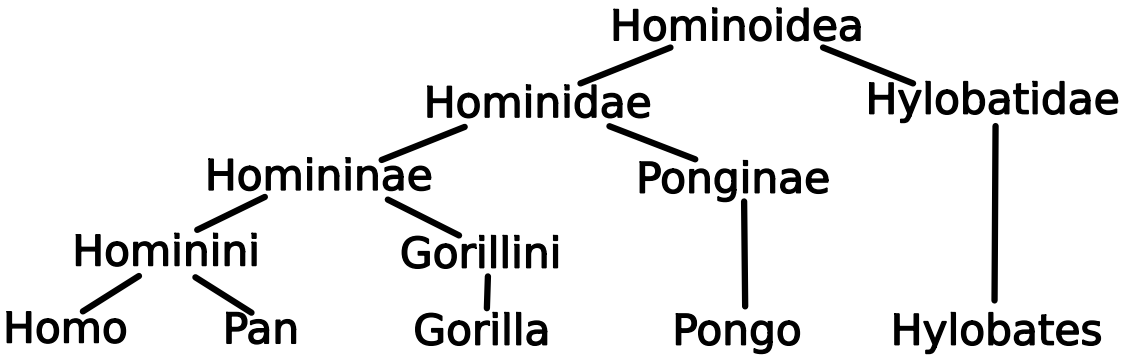

Figure 10.1: Phylogenetic tree for the Hominoidea.

The incorporation of CTL path operators with temporal bounds allows the localization of ancestors whose descendants are placed at a certain distance and are labeled with one of the mentioned tags (see Section 11.2 for a more detailed example and notation). Traditionally, the phylogenetists solve the problem of computing distances with an ad hoc script that adds the weights of a path in a enriched tree with time. The insertion of explicit time in the CTL path operators avoids the codification of these specific scripts and maintains all the advantages of model checking exposed in this thesis. The new model checking algorithms will embed the extraction and addition of distances through the traversing of the states in the phylogeny.

Besides, the algorithms of qualitative model checking are unable to manipulate quantitative information in the edges of a transition system or generate an integer as output result. This is of remarkable interest when we try to extract temporal information from the tree (e.g., $distance(pongo, homo)$) instead of testing with particular values. In general, the new model checking algorithms should focus on a) the evaluation of properties whose truth value depends on these quantitative relations; or b) the evaluation of properties whose result is a numerical value computed using these annotations.

For the first case, the descriptive aptitudes of the logics are extended to handle with the numerical information in the equations. The existing temporal logics are recycled for this field in order to cover the maximum number of phylogenetic properties querying about quantitative data. For the second case, the model checking process is modified with respect to the typical verification of a temporal logic formula: now, it will return a number instead of a boolean value. The model checking algorithms are extended with the capability for evaluating arithmetic functions over the set of states characterized by a formula of temporal logic. As explained in Section 2.4, the set of tree states are selected according to the relations imposed by a temporal logic formula, and they can be considered as *objects* or *entities* in

a phylogenetic tree. The studies can be enlarged with the introduction of parameters in the equations: there exist automatic tools that infer the combination of propositions and values (quantitatives or not) that make the equations true.

The aim of this chapter is to review the group of non-qualitative phylogenetic properties and classify them according to a) the kind of information that is tagged in the tree/specifications for the verification process, and b) the kind of answer the verification will return (boolean or numerical) as well as the procedure it uses for its computation. The introduction of quantitative information in phylogenetic specifications will be motivated by real examples. Looking at these examples, we will identify the requisites that the future logics should have. The design of such logics will also pay attention in the main aspects of decidability: many properties expressed in linear or branching-time logics based on Presburger formulas are undecidable.

Finally, the last important factor to take into account is the availability of predefined logics and the associated verification tools covering the proposals presented here. We will present the current developments of quantitative extensions in the field of model checking, apply them to the definition of new properties over phylogenies and provide to the phylogenetists a complementary framework where they could inspect numerical relations. The enlargement of the semantics of Kripke structures and the expressiveness of the temporal logics contribute to the definition and verification of such properties. The expression and evaluation of quantitative properties involving phylogenetic hypothesis is the main advantage of this chapter.

10.2. Towards a Classification of Quantitative Properties Arising in Phylogeny

Evolutionary pressure of natural selection affects differently to each taxon because they are fixed to incomparable environments and ecosystems. Even, genes concerning disparate traits of the same species mutate with apparently independent speed, generating different topologies of gene trees. Moreover, current phylogenetic inference methods working with the same input not necessarily obtain equal output phylogenies [71, 176]. Consequently, evolution is not an homogeneous process, which is reflected in the variability of regions in genomic alignments and, indirectly, the trees built from these data [131, 132, 119]. Phylogenies express these biases through the tree diversity and unbalanced trees (binary or not).

Therefore, the branches that are supported by the majority of the inferred phylogenies determine the main tree skeleton and the core of the evolution. The inclusion of probabilistic information in the branches of the tree appears naturally during the statistical analysis of the phylogeny: probabilities help to the detection

of the *consensus* tree [71], i.e., expressing the rate of trees confirming the topology. The computation of these probabilities in the branches can be carried out by the model checking tools using the equations of maximum likelihood estimations (MLE) [69]. To this end, we need to extend the capabilities of the software tools for evaluating quantitative functions over the set of states selected by the traditional model checking algorithms: now, they will return a number as output result instead of a boolean value.

Otherwise, likelihoods also work for inspecting the probability of reaching a given state of the phylogeny, for example, the probability of reaching a population with a chronic disease from the root (i.e., the lactose intolerance [160, 92]). For this case, we introduce explicitly the probability inside the specification of the property that we are going to evaluate: the result will be a boolean telling whether the number of states satisfying the temporal logic formula is over a certain ratio or not. Commonly, it is assumed that every descendant is reached with equal probability from the root (i.e., every transition is equiprobable).

Additionally, the edges are occasionally enriched with time labels or lengths for marking the differences in the speciation speed that appear in the nature. A tree with integer weights in the branches is generated as output of a distance-based phylogenetic reconstruction method [148], which builds the tree measuring the number of mismatches between each pair of genomes. Therefore, the inclusion of temporal labels in a phylogenetic tree inherits the concept of character distances from the tree inference methods. In fact, the integer tags representing the time in the edges have a dual meaning: they can be interpreted as the rate of nucleotide variation between two consecutive genomes of the tree, or the chronological time of the evolution. Besides, the knowledge of time is useful for learning complex properties about the evolution and speciation speed, for instance, the estimation of the temporal point of divergence between species [17], the diaspora of human populations [76, 36, 101, 166] or army invasions [179].

Hence, a phylogenetic tree can be labeled with many types of tags, quantitative or not: phenotypes such as chronic diseases or endemic features, geolocation of populations and migrations, and so on. These data are specific for each phylogenetic study, but we search for a generic criteria for classifying the addition of new information. Time and probability are the most common quantitative information and they are shared by almost every phylogenetic study because of the aforementioned arguments. Therefore, the modifications of the logic, data structures and algorithms will be driven by these types of data. In sum, we can organize the tags as integers (chronological or character distances) or non-integers (probabilities) for this first approach. Apart from the new annotations of the phylogeny, the quantitative information can also be placed in the result of the model checking evaluation: instead of returning a boolean, the output is an integer or real number.

For compatibility with the annotations of the tree, the quantitative output of the model checking algorithms can follow the classification of integers (chronological or character distances) or non-integers (probabilities) as well.

10.3. Extending the Labels of the Phylogenetic Tree and the Results of Model Checking

So, the inclusion of quantitative information in the phylogenetic analysis can be boarded at two different levels: either in the definition of properties and the annotations of the tree, or in the result returned by the model checking process. If we focus on the first level, the numerical data inserted into the tree and specifications can be divided in a) *timed* or distance information, b) *probabilistic* information and c) *raw quantitative* information. The first two ones are mainly related to the labeling of the tree branches, while the last one is more general and involves numerical values and comparisons in the atomic propositions [137].

Take the following example. A phylogenetic tree with advanced annotations is illustrated by Figure 11.1. It represents a phylogeny whose branches are scaled proportionally to the time. The subtrees are labeled with probabilities representing either the statistical support for that clade, or the likelihood of taking that edge during the speciation. For instance, imagine that the tips represent individuals from different ethnic groups and the leaves are labeled with their DNA, an identifier and a boolean indicating if they suffer from a specific illness. Suppose that X and Y suffer from lactose intolerance, an endemic disease that causes the inability to digest milk. The rest of members are healthy.

Timed information. The branch lengths are proportional to the mutation clock pointed by the segment in the legend. Each transition implicates a change of 0.05 nucleotides (5%) per site and per unit length with respect to the direct ancestor, and thus, it provides a visual mode for studying the total rate of variation of the genomic sequences along a segment. In this example, we can inspect the number of changes between the ancestor of X and Y, and the rest of healthy individuals: it indirectly tells the estimated date of appearance of the (in)tolerance. A phylogenetic tree whose branches are proportional to the number of nucleotide changes is called *phylogram*, and *chronogram* if their edges are directly proportional to the expected evolutionary time. For distinction, the basic phylogenetic trees whose branches are not scaled with respect to neither time nor character changes are simply called *cladograms*. ■

Probabilistic information. The red values attached to the internal nodes of Figure 11.1 indicate the statistical consistency or support of a clade. In brief, those numbers express the percentage of inferred or bootstrapped phylogenies

whose leaves are arranged forming the same subtree. The ordering of the tips is irrelevant for the identification of subtrees [18]. An internal node labeled with 1.0 means that the descending taxa are clustered together in every inferred phylogeny constructed using the relationships imposed by the sequences. It is possible to compare phylogenies through the computation of maximum likelihoods estimations [69]. Not only do the probabilities play an important role for the comparison of trees through the evaluation of the tree coherence, but also they are useful for showing the prevalence of chronic diseases inside genetic populations. The blue values indicate the probability of moving from one node to another following that branch: by default, we assume that every node is equiprobable and the summation of probabilities for each descendant must be 1.0. Using the blue values in this example, the probability of reaching an individual with lactose intolerance from the root is 0.25 (the likelihood of the X-Y subtree is 0.5×0.5). ■

Raw quantitative information. The incorporation of raw quantitative information to the atomic propositions of the tree states increases the expressive power of the system. This quantitative information is essential for describing questions about biochemical properties related to the sequences, for example, asking the polarity, hydrophobia and size of the amino acids in the synthesized protein. The necessity of including these data was already pointed in Section 4.3 during the investigation of the conservation and covariation of sequences and their attributes. The addition, manipulation and comparison of numerical labels is already supported in the verification process of some current model checking tools, but it is important to take in mind the complexity of introducing Presburger and Peano arithmetics. We classify the states with raw quantitative information in a separate category in order to emphasize the distinction of the labels (genomic characters vs numerical atomic propositions). ■

Apart from the previous concerns that classify the numerical labels of the tree in three families (timed, probabilistic or raw quantitative information), the need for the extraction of quantitative results in some phylogenetic analysis is clear: recall the calculation of distances between populations (pongos and homo) instead of testing particular values. If we focus on the answer returned by the verification process rather than the type of tags, the second big classification of phylogenetic properties is separated in:

- Boolean properties, where the result is *true* when the property is satisfied, or counterexamples when the property is false;

- Numerical properties, where the result is the computation of a number according to a function applied over a group of elements, and whose most prominent exponents are the calculation of (continuous or discrete) time distances or probabilities; and, finally,

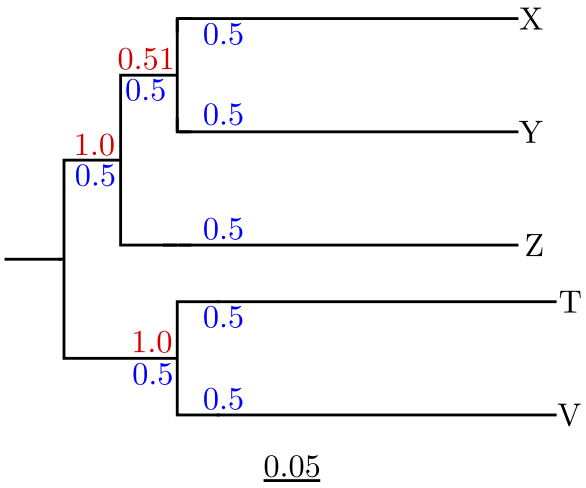

Figure 10.2: Phylogenetic tree labeled with quantitative information.

- Parametric properties, where the result is an algebraic equation, occasionally enriched with the set of integers or characters whose values make true the temporal logic formula. This equation may involve quantitative relations or genomic sequences.

The study of boolean properties has been already detailed in Chapter 4. Given an initial state, they compute a set of reachable nodes fulfilling a certain feature. The application of quantitative functions to inspect the numerical relations over that set is also important. This quantification is crucial for the computation of distances (timed or not) between sets of taxa or clades. The distances between elements are mainly employed for the calculation of topological measures (e.g., tree depth and balance [130]) and phylogenetic comparative methods (e.g., Robinson-Foulds [146]). The inclusion of distances in the topological properties presented in Section 4.2 helps to study, compare and refine the tree structure [158]. These supplementary metrics are of particular interest in the case of several phylogenies sharing identical scores under the same inference method [159]. Conversely, the evaluation of numerical properties may return probabilities too; for instance, during the calculation of the statistical support of a tree [69, 71]. We can analyze the fitness of a set of trees according to all these informations.

Finally, parametric model checking allows the placement of parameters either in the specifications or in the transitions of the model. The use of parameters in the specifications helps to the identification of potential valuations that satisfy a property instead of the manual inspection of the whole state space [38]. The evaluation of a parametric formula will return a set of instances that confirm the equation. Some properties such as the detection of back mutations, or the appear-

ance of conserved and correlated regions of the DNA, can be expressed in terms of parameters instead of particular instances of nucleotide characters. Properties involving the verification of numerical restrictions over time, probabilities or raw quantitative data are also susceptible to the definition of upper and lower parametric bounds in the CTL path operators [32]. In case of the placement of numeric parameters in the model, the result of the evaluation is a mathematical equation representing the relations that must be fulfilled by the values.

10.4. Quantitative Properties in Phylogenetics

One of the main results of this chapter is the sorting of the phylogenetic properties with quantitative requirements. Following the catalog proposed in this section, the phylogenetic properties are organized in a Cartesian product presented in Table 10.1. In the first column, we are centered in properties related with the genome and its arrangement in the tree. If we focus on properties with a boolean result, we obtain Table 4.1. In Chapter 4 we detailed the methodology for defining and evaluating properties of this genre. The use of parameters allows the extraction of the columns (i,j) and nucleotides (σ_i, σ_j) making true the equations. By definition, the output result of a qualitative formula cannot be a number.

Secondly, the inclusion of explicit time admits the specification of restrictions such as $distance(ape, homo) > 10$, where we test if the accumulated weight for the transitions between these two taxa is over or below 10. As explained previously, the verification or computation of distances between sets of populations can be carried out by extended CTL-like formulas that are evaluated over a enriched phylogeny with temporal information. These new CTL path operators collect the weight of each branch and test if the accumulated length is over the indicated value or not (Chapter 11). Conversely, the algorithms of quantitative model checking are updated for calculating integers (distances) and float numbers as output. The calculation of distances between symbolic objects is detailed in Chapter 12. Either the distance or the symbolic identifier of species in the specifications can be replaced by a parameter.

Thirdly, we test if the number of states validating a property in the phylogeny is over or below a certain probabilistic bound. Several types of phylogenetic properties accept the inclusion of probabilities in their definition. For instance, we can evaluate sequence properties (e.g., checking the ratio of DNA mutations that activate the lactose tolerance in the phylogeny) or tree properties (e.g., checking the likelihood of generating a particular topology with a DNA substitution model). Similarly to the calculation of distances, the CTL path operators that are adapted for probabilities collect the likelihood of each branch and test if the accumulated probability is over the indicated value or not. Although probabilities are essentials

in both types of properties, the calculation of likelihoods differs in each situation because they use different models: the evaluation of probabilities over the alignments requires a phylogeny (model) labeled with likelihoods (Chapter 15), while the evaluation of the tree support for a certain topology requires a DNA substitution model (Chapter 14). Some model checking tools such as PRISM [124] have algorithms that return these float numbers as output of the evaluation.

The parameters introduced in the genome for a sequence property are identical to the parameters introduced in the first column. The DNA mutation model can be also left undetermined: in this way, the model checking tool will search for the best parameter of the DNA model that generates the topology of the tree, but this field is still unexplored and needs a more detailed study. The model checking algorithms can be updated for returning the maximum or minimum probability of satisfying a property.

Finally, the properties with raw quantitative information include arithmetic comparisons in the atomic propositions. The last column is equal to the first column in terms of restrictions and expressiveness, with the addition of numbers as the only novelty in the inequalities and parameters. The type of properties explained in Section 4.3 (conservation or covariation of biochemical patterns) falls within this category. The incorporation of quantitative information may increase the complexity of the associated logics and model checking algorithms.

10.5. Conclusions

The main contribution of this chapter is the identification of the limitations of phylogenetic model checking for inspecting more complex properties. To this end, we have detected new quantitative requirements for the analysis of phylogenies. The phylogenetic properties are classified and organized according to the quantitative information they use: principally the time (distance) between states and probabilities. There exists many more quantitative tags, but these two are the most representative. The evaluation of hypothesis with this kind of data requires the extension of the phylogenetic tree with temporal and probabilistic annotations, a logic with a renovated syntax, and an algorithm considering this knowledge. The inclusion of quantitative information is not exclusively restricted to the definition and verification of specifications with explicit time or probabilities: now, the model checking process may return numerical results as output as well. This last approximation demands the update of the traditional verification algorithms for managing and computing specific functions over a sets of states characterized by a temporal logic formula.

In sum, the next chapters explain the incorporation of time and probabilities to model checking with intuitive examples. Our goal is the application of existing

	Qualitative	Time	Probability	Raw
B	Table 4.1	Phylogenetic properties enriched with distances: $distance(pongo, homo) > 10$?	Probabilities over alignments: $Pr(conservation(2, 3, AA)) > 0.8$? Likelihood of a tree topology given a DNA mutation model: $Pr(Jukes\_Cantor, topology) \leq 0.4$?	Biochemical properties: $\mathbf{EG}((size > 110) \wedge (size < 130))$?
N	—	▪ (Im)balance metrics ▪ Max/min distances between clades, roots and leaves ▪ Length of right and left paths ▪ Symbolic distances (Robinson-Foulds)	▪ Max/min probabilities over alignments ▪ Max/min likelihood of a tree topology	—
P	$i, j \in \mathbb{N}$, $\sigma_i, \sigma_j \in \Sigma$, $seq \in \Sigma^{(j-i)}$: $covariation(i, j, \sigma_i, \sigma_j)$ $conservation(i, j, seq)$ $hasBM(i, \sigma_i)$	Phylogenetic properties enriched with distances: $X \in \mathbb{N}$: $distance(pongo, homo) > X$ $A, B \in species$: $distance(A, B) > 10$	Probabilities over alignments: $i, j \in \mathbb{N}$, $seq \in \Sigma^{(j-i)}$: $Pr(conservation(i, j, seq)) > 0.8$ Inferring the parameters of a DNA mutation model for a tree topology: $Pr(model, topology) \leq 0.4$	Biochemical properties $X, Y \in \mathbb{N}$: $\mathbf{EG}((size > X) \wedge (size < Y))$

Table 10.1: Summary of the most important phylogenetic properties (Type B: boolean; N: numeric; P: parametric).

developments in quantitative model checking to the verification of phylogenetic questions. The structure is arranged as indicated. First of all, Chapter 11 introduces real (continuous) time in the specifications and transition systems. It presents the syntax and semantics of Timed CTL (TCTL), with an application to compute temporal distances between objects characterized symbolically in Chapter 12. We continue with the introduction of probabilities that depend either on discrete (Chapter 14) or continuous time (Chapter 15). To this end, we use previous investigations in Markov chains and logics such as Probatilisic CTL (PCTL) of Continuous Stochastic Logic ($\mathrm{CSL}^{\mathrm{TA}}$). Finally the application of parametric model checking is presented as a future work and we finish with the conclusions of this thesis.

Chapter 11

Timed Transition Systems and Logics

Marty McFly: Wait a minute, Doc. Ah... Are you telling me that you built a time machine... out of a DeLorean?
Dr. Emmett Brown: The way I see it, if you're gonna build a time machine into a car, why not do it with some style?
Back to the Future

11.1. Introduction

The importance of temporal constraints and distances transcends multiple aspects of phylogenetic analysis. Phylogenetic reconstruction methods such as Neighbor-Joining establish their theoretical background in the concept of distance between each pair of taxa or individuals [148]. Hence, the edges of the inferred tree are implicitly labeled with distances. Although they are based on nucleotide distances, which measure the number of differences among sequences of an alignment, the relation between chronological time and nucleotide distances is known if only we can approximate the mutation clock (i.e., estimating the number of changes in the genome per year).

Indeed, a mapping between time and mutations in the transitions of the phylogenetic tree returns meaningful information about the number of changes during the elapsed time of a branch: nucleotide distances can be reformulated in terms of time intervals. The inclusion of temporal distances in the branches is relevant for centering the verification of a phylogenetic property to paths with a specific length in the phylogeny.

The presence of time in the branches of the phylogenetic tree is not unusual, particularly in the studies of migrations in human populations [76, 36, 101, 166] or army invasions [179]. Although the distances in population trees are expressed in multiple ways (via the number of mutations between states or with a chronological scale), the need of collecting these informations inside our framework is essential. One of our main interests is the verification that the distance between the root and the rise of a particular mutation falls within a certain threshold.

Moreover, the logical formulas that investigate sequence properties with deleterious or compensated mutations are usually focused in the vicinity of a state or the tree surroundings. A deleterious mutation requires to terminate without offspring in a small finite number of steps (Equation 4.18). The extreme conditions of low atmospheric pressure at high altitude in the Tibetan Plateau and Andean Altiplano mark the genome and penalize disadvantageous characteristics [19, 25]. Tibetan women with genetic adaptations that improve the supply and transport of oxygen has more surviving children [20]. Recent studies approximate the hypothetical dates of colonization of those areas using an estimated mutation clock, counting the number of expected mutations per kiloyear (1000 years) with respect to the most recent common ancestor of the individuals sharing the beneficial genotype [177].

Another example is portrayed by the woolly mammoths [34]. Woolly mammoths include a mutated gene that evolved from a previous version found in the common ancestor of elephantidae, and that remains almost unaltered in modern elephants. This mutated gene changes the polarity and structure of the hemoglobin and modifies how the oxygen binds in that place. The adaptation prevents freezing the blood and awards the mammoths with an improved oxygen delivery around the body in cold climates in opposition to their closest warm-climate relatives, the African and Asian elephants. Knowing the length of this branch of the phylogeny, it is possible to infer the date of the last glaciation because it supposed a critical climate event in mammals settled in northern latitudes.

Internally, the acquisition of new functionalities arises by the accommodation of correlated changes that, in a stable way, regulates the expression of a protein or updates its morphology and biochemical properties. The detection of correlated and conserved positions in the genome is influenced by the proximity of the tree states. Short paths whose characters evolve in covariation are less prone to appear by hazardous mutations but emerge caused by structural or biochemical restrictions and selection pressure that guide those compensatory changes [41, 77]. Hence, the inclusion of timed bounds in the verification of a covariation hypothesis (Table 4.1) helps to separate compensated mutations from spontaneous changes. A similar reasoning can be applied to the study of time and space locality in SNP's, conserved sequences, and reversions or back mutations.

In sum, the phylogenetists are interested in evaluating phylogenetic properties that returns distances as result of the analysis; or properties with the option of limiting the validity of a formula to a period of time. The use of explicit time in the branches delimits more precisely the interval of time and the cone of influence (tree paths) where we aim to check a phylogenetic property. Enriching the states with ecological information, biochemical properties and three-dimensional images of the proteins, it is possible to determine the effects of a string of mutations caused by compensatory and selection-pressure events in the branch of Tibetans or mammoths. The scrutiny and comparison of the genomical sequences in closely related species allows the biologist to obtain the principal genetic differences between them. Finally, the inclusion of time in the branches allows the computation of distances for the detection of topological properties such as parallel and convergent evolution (Section 4.2). The previous examples reaffirm the obligation of introducing temporal logics with real time.

Therefore, the questions solved by timed logics and structures in the field of phylogeny are mainly related to the verification of common phylogenetic properties enlarged with explicit time distances. The accretion of temporal logics with time constrains enhances most of the properties defined in temporal logic in Table 4.1. In brief, the kind of questions that we try to solve in this section can be summarized as: when (dis)appeared a certain mutation or phenotype? Later, this information can be associated with cultural, migratory or environmental events.

Thus, this chapter is devoted to the presentation of an existing temporal logic that includes explicit time in the specifications. The objective is to provide an extended framework for the definition and evaluation of phylogenetic properties with quantitative restrictions; in this case, distances. To this end, we enrich the phylogenetic tree with clock restrictions in the branches and adapt the existing TCTL (Timed CTL) to the domain of phylogenetics. After this introduction, Section 11.2 introduces the syntax, data structures and semantics required for the logic. We exemplify the specification of a phylogenetic property with this new notation. Next, Section 11.3 shows the algorithms for verifying and managing properties with explicit time and distances in the specifications.

Finally, Chapter 12 gathers all the concepts presented here and applies TCTL to solve specific problems, for instance, the inspection of the balance and asymmetries of trees. More in detail, Chapter 12 works with an extended version of TCTL that returns temporal distances as result of the evaluation of a formula. Our main contribution is to show how we apply the notions presented here for the modeling of phylogenetic properties that need the incorporation of time in the specification or the computation of distances as output result.

11.2. Timed Logic and Structure

The first step for introducing time is the choice of its domain. A discrete time is conceptually simple and the operations needed for its manipulation become easier, while a continuous time is more powerful and expressive. For that reason, we select a continuous time domain in order to cover the definition of a large number of phylogenetic properties. Real time logics explicitly introduces the notion of time in model checking [8]. The inclusion of time constraints is addressed through heterogeneous ways: from bounded temporal operators syntax in the temporal logic till references to explicit clock variables in the model. The first notation, inherited of Timed CTL (TCTL) [6, 7], suffices for our purposes. Such logic allows bounded temporal operators of the form $\mathbf{AF}_{\leq 5}p$ meaning that inevitably event p will occur within five time steps, with p the mutation characterizing the Tibetan trait for example. The evaluation of real time logics needs a refined data structure that provides timed semantics to the temporal formulas. Many definitions have already been proposed [90, 13], but an adaptation of the transition system with timed paths is enough for our objectives.

A Kripke structure (Definition 2) must be modified for managing timed environments. Originally, it models a system capable of an infinite number of behaviors or *paths*, infinite sequences of successive states $\pi = s_0 s_1 s_2 \ldots$ such that $s_0 \in S_0$, $(s_i, s_{i+1}) \in R$, $i \in \mathbb{N}$. The set of possible executions (paths) in that structure can be unfolded into its *computation tree*. Now, the paths should be extended with temporal information. In a discrete time system, the representation of paths keeps the explicit succession of consecutive states because the distinction of states in function of their time stamp is clear. However, in dense time, there is always an infinite number of intermediate states between two temporal points. Thus, a new description of the path is required in which we map the time to the states. A *timed path* is a path π that evolves through the time. The function $\rho(\pi, t)$ identifies the state s_i of the path in which the system is found after $t \in \mathbb{R}$ time units since the initial state s_0, i.e., $\rho(\pi, 0) = s_0$.

A complete grammar and semantics of TCTL formulas can be defined from a minimal subset of logical operators. Note that TCTL has no *next* ($\mathbf{X}$) operator: if the time is dense, then there isn't a successor by definition of real numbers. In fact, there are infinite real numbers between two time stamps, and probably an infinite number of states as well.

Definition 8 (Timed Computational Tree Logic)**.** *A temporal logic formula ϕ is defined by the following grammar where $c \in \mathbb{R}$ is a time stamp and $\sim$ stands for one of the binary relations $\{<, \leq, =, \geq, >\}$:*

$$\phi ::= true \mid p \mid \neg\phi \mid \phi \vee \phi \mid \mathbf{E}\left[\phi \mathbf{U}_{\sim c} \phi\right] \mid \mathbf{A}\left[\phi \mathbf{U}_{\sim c} \phi\right] \tag{11.1}$$

The formulas are checked against a transition system modeled with a Kripke structure $M = (S, S_0, R, L, AP)$ *over the set of atomic propositions* AP*. The verification considers all timed paths* π *from a certain initial state* $s_0 \in S$*. Notice that* $M, s_0 \vDash \phi$ *means that a path* π *starting from* s_0 *satisfies* ϕ *in* c *time units. The semantics of well-formed formulas is as follows (let* $\pi = s_0 s_1 s_2 \ldots$ *with* $\rho(\pi, t) = s_j$ *the state of the Kripke structure at time* t*):*

- $M, s_0 \vDash p \Leftrightarrow p \in L(s_0)$,
- $M, s_0 \vDash \neg\phi \Leftrightarrow M, s_0 \nvDash \phi$,
- $M, s_0 \vDash \phi \vee \psi \Leftrightarrow M, s_0 \vDash \phi$ *or* $M, s_0 \vDash \psi$,
- $M, s_0 \vDash \mathbf{E}[\phi \mathbf{U}_{\sim c} \psi] \Leftrightarrow$ *for some* $t \sim c$, $\exists \pi : M, \rho(\pi, t) \vDash \psi$, *and* $M, \rho(\pi, t') \vDash \phi$ *for all* $0 \leq t' < t$.
- $M, s_0 \vDash \mathbf{A}[\phi \mathbf{U}_{\sim c} \psi] \Leftrightarrow$ *for some* $t \sim c$, $\forall \pi : M, \rho(\pi, t) \vDash \psi$, *and* $M, \rho(\pi, t') \vDash \phi$ *for all* $0 \leq t' < t$.

Timed variants of the modal operators $\mathbf{F}$ and $\mathbf{G}$ are obtained via $\mathbf{U}$ as $\mathbf{F}_{\sim c}\phi = true\,\mathbf{U}_{\sim c}\phi$ and $\mathbf{G}_{\sim c}\phi = \neg\mathbf{F}_{\sim c}\neg\phi$. The representation of $\mathbf{F}_{\sim c}$ is often a shorthand of intervals. For instance, $\mathbf{F}_{\leq c}$ denotes $\mathbf{F}_{[0,c]}$ and $\mathbf{F}_{>c}$ denotes $\mathbf{F}_{(c,\infty)}$. The inspection of properties with time constraints requires a deep knowledge of the phylogeny so as to approximately estimate the distances between sets of nodes represented with a temporal formula.

A simplified discrete time logic called Real Time CTL (RTCTL) assumes by default that each transition of a Kripke structure consumes an unit time tick [66]. Then, it is possible to verify RTCTL formulas in a Kripke structure without explicit timed extensions because the logic implicitly adds an unitary weight to each transition. Nevertheless, TCTL is more flexible and works with a powerful syntax that requires the definition of a background timed structure. A timed automaton is the basic data structure that provides the semantics to the verification process.

Definition 9 (Timed Automaton)**.** *A timed automaton is represented by a tuple* $TA = (S, S_0, R, L, AP, C, \lambda, \tau)$ *where:*

- S *is a finite set of states,*
- $S_0 \subseteq S$ *is the set of initial states,*
- $R \subseteq S \times S$ *is a total transition relation between states, i.e., for every state* $s_i \in S$*, there exists* $s_j \in S$ *such that* $(s_i, s_j) \in R$,
- $L : S \to 2^{AP}$ *is the labeling function that associates each state with the subset of atomic propositions that are true of it,*

- *AP is the set of atomic propositions,*
- *C is a finite set of clocks,*
- $\lambda : R \to 2^C$ *indicates the transitions whose clocks are reset, and*
- $\tau : R \to 2^{CC(C)}$ *is a function labeling each transition with a set of clock constraints* $CC = \{x \sim c \mid x \in C \wedge c \in \mathbb{N}\}$ *that restricts the conditions for firing that edge. The value c is a time stamp of the clock x and* $\sim$ *stands for one of the binary relations* $\{<, \leq, =, \geq, >\}$.

The decidability of the model checking problem is not affected if c is a rational: it can be scaled in order to convert it to a natural. The transitions between states are instantaneous and several transitions can be fired at the same time instant. The set of clock restrictions τ indirectly represents the delay spent in a current node during the execution of the system until the branch conditions are fulfilled. All the clocks run regularly at the same speed and are initialized to zero. Let $Eval(C)$ be the set of all the clock assignments to the clocks of C. The notation $\nu(x)$ accesses to the current value of the clock $x \in C$, with $\nu(x) \in \mathbb{R}$. In other words, ν is a function that maps clocks with dense time such that $\nu : C \to \mathbb{R}$. More generally, ν without parameters represents the valuation of the whole clocks C.

The transition system associated to a timed automaton corresponds to the Kripke structure needed for handling the infinite timed paths that the specifications in TCTL require. The verification of a TCTL formula in a timed automaton is equivalent to the verification of the same TCTL formula in the transition system [7, 13]. The timed transition system demanded for the evaluation of a TCTL formula is defined as:

Definition 10 (Transition System)**.** *A transition system* $TS(TA)$ *associated to a timed automaton* $TA = (S, S_0, R, L, AP, C, \lambda, \tau)$ *is represented by a Kripke structure defined by the tuple* $M = TS(TA) = (S', S'_0, R', L', AP')$ *where:*

- $S' = S \times Eval(C)$ *is the set of states,*
- $S'_0 = \{\langle s_0, \nu_0 \rangle \mid s_0 \in S_0 \wedge \nu_0(x) = 0 \text{ for all } x \in C\}$ *is the set of initial states,*
- $R' \subseteq S' \times S'$ *is a total transition relation between states defined as:*
  - *A* discrete transition $(\langle s_i, \nu_i \rangle, \langle s_j, \nu_{i+1} \rangle)$ *such that* $r_i = (s_i, s_j) \in R$. *The clocks* $\lambda(r_i)$ *are reset, i.e.,* $\nu_{i+1}(x) = 0$, $\forall x \in \lambda(r_i)$. *Besides,* $\nu_{i+1}(x) = \nu_i(x) + t_{i+1} - t_i$, $\forall x \in C \setminus \lambda(r_i)$ *with* $t_{i+1} > t_i$, *and* $t_i, t_{i+1} \in \mathbb{R}$. *The clock valuation* ν_{i+1} *satisfies a firing condition* $\tau(r_i)$.

- *A* delay transition $(\langle s_i, \nu_i\rangle, \langle s_i, \nu_{i+1}\rangle)$. *Only the clocks are updated with* $\nu_{i+1}(x) = \nu_i(x) + t_{i+1} - t_i$, *with* $t_{i+1} > t_i$, *and* $t_i, t_{i+1} \in \mathbb{R}$. *The clock valuation* ν_{i+1} *may not satisfy any firing condition* $\tau(r_i)$ *yet.*

- $L'(\langle s, \nu\rangle) = L(s) \cup \{g \in CC(C) \mid \nu \vDash g\}$ *is the labeling function, and*
- $AP' = AP \cup CC(C)$ *the set of atomic propositions.*

Now, the notation of the infinite paths π is enlarged for considering the potential infinite set of states S' (which is caused by the Cartesian product with the uncountable number of valuations for the real time clocks $Eval(C)$). Hence, following the discrete transitions of the transition system, π evolves with the cadence $\pi = \langle s_0, \nu_0\rangle\langle s_1, \nu_1\rangle\langle s_2, \nu_2\rangle \ldots$. In this case, the state $\langle s_i, \nu_i\rangle$ changes to $\langle s_j, \nu_{i+1}\rangle$ when ν_i satisfies the enabling condition τ, which indirectly determines a *discretization* of the dense-time paths in the timed automaton. Otherwise, the path π remains unaltered in the same state s_i while it executes a delay transition, i.e., $\pi = \langle s_0, \nu_0\rangle\langle s_0, \nu_0 + t_1\rangle\langle s_0, \nu_0 + t_2\rangle \ldots$ with $t_2 > t_1$. The function $\rho(\pi, t)$ is updated for considering the new type of π, i.e., $\rho(\pi, 0) = \langle s_0, \nu_0\rangle$.

The translation of a branching-time phylogeny to that transition system is addressed by the extension of Definition 3 with the inclusion of clock restrictions τ in the set of edges E of the tree. In a phylogeny, these constraints define upper or lower temporal bounds in the connections between species, probably marking the interval of time when those taxa diverged. The self-loops in terminal nodes of the transition relation R' lead to perpetual siphons (by default, these self-loops consume 0 ticks of time). Every tree node can be also enlarged with local clock variables for monitoring the time spent until the next state. Normally, a single clock representing the chronological time through all the tree will be enough for our purposes in phylogeny, but we keep the notation of a set of clocks C for generality.

Finally, let's remember the examples presented in Section 11.1 that motivated the introduction of temporal logics with explicit time. For instance, we want to discover the date of divergence of the Tibetan ethnic group with respect to its closest relative population. In particular, we aim to detect when the genetic adaptation that favors the habitability in high altitude zones appeared (it marks the tribe genome as well). To this end, we define a phylogenetic property that searches for the most recent common ancestor of the Tibetans. The following TCTL Equation 11.2 with time extensions introduces the notation that we have presented here. This formula investigates, in the phylogenetic tree of Figure 11.1, if there exists an internal node (**EF**) whose terminal leaves are reachable in a distance less than 3 kiloyears ($\mathbf{AF}_{\leq 3ky}$) and the individuals they represent possess the mutations affecting to this new phenotype:

$$\mathbf{EF}\ \mathbf{AF}_{\leq 3ky}(seq[i] = j) \tag{11.2}$$

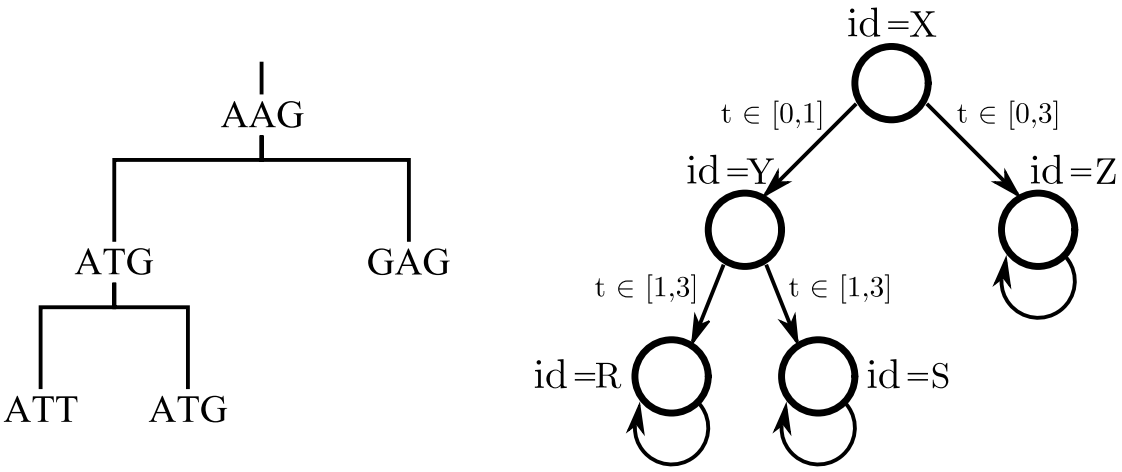

Figure 11.1: Phylogenetic tree and its transition system labeled with time intervals in the branches and taxon identifiers in the nodes (the DNA sequences are omitted for readability).

This equation can be exported to other examples such as the study of time in the woolly mammoth lineage. Hence, TCTL is suitable for the definition of phylogenetic properties enriched with explicit time distance. The existence of software tools and model checking algorithms for managing TCTL queries provides a complete framework for the verification of biological hypothesis. The model checking algorithms needed for handling TCTL formulas are presented in the next section.

11.3. Algorithm for Timed Model Checking

The algorithms of model checking are enlarged for taking into consideration the new temporal information of the transition system. Due to the infinite computational tree, the model checking process has to do a few simplifications in order to cope with this complexity. These simplifications are founded on the fact that certain states of the transition system are indistinguishable by this logic if we focus exclusively on the time, for example, when the time elapsed in two different states is the same. The underlying idea is the identification of equivalence classes according to compatible temporal constraints. To this end, we define an equivalence of clocks [7].

Definition 11 (Equivalence of Clock Assignments)**.** *Two clock valuations* $\nu, \nu' \in Eval(C)$ *are equivalent* ($\nu \equiv \nu'$) *if they satisfy the following conditions.*

- *For each* $x \in C$, $\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor$ *or both* ν *and* ν' *are greater than* c_x, *with* c_x *the largest constant of* x *found in the firing conditions* τ *and* $\lfloor \cdot \rfloor$ *the integer part of the real number.*

- *For each pair of clocks* $x, y \in C$, *such that* $\nu(x) \leq c_x$ *and* $\nu(y) \leq c_y$, *the fractional parts* $(fract)$ *of the clocks should:*

  - $fract(\nu(x)) \leq fract(\nu(y))$ *iff* $fract(\nu'(x)) \leq fract(\nu'(y))$, *and*
  - $fract(\nu(x)) = 0$ *iff* $fract(\nu'(x)) = 0$.

The application of this equivalency defines a finite number of clock regions in the original transition state space. In other words, it groups the infinite set of states S' into a finite set of clusters, each one fulfilling compatible time restrictions: the clock valuations $\nu \in Eval(C)$ are organized in equivalent regions (denoted as $Eval(C)/\equiv$). The equivalence class of ν is denoted by $[\nu]$, i.e., the set of ν' that are equivalent to ν. As a particular case, an *end class* is the equivalence class satisfying $x > c_x$ for all clocks x. An equivalence class α is a *boundary class* if for any $\nu \in \alpha$ and $t \in \mathbb{R}_{>0}$, ν and $\nu + t$ are not equivalent. We must define a *successor* function over the clock equivalences in order to capture the transitions between clock regions [7].

Definition 12 (Successor Region)**.** *Given a set of clock regions defined over* $Eval(C)/\equiv$ *according to the equivalence of clock assignments (Definition 11), the successor of a clock region* α *is* β $(succ(\alpha) = \beta)$ *iff for each clock* $\nu \in \alpha$ *there exists* $t \in \mathbb{R}_{>0}$ *such that* $\nu + t \in \beta$ *and* $\nu + t' \in \alpha \cup \beta$ *for all* $t' < t$.

The state space is reordered in terms of regions defined over the transition system, classifying the states $\langle s, \nu \rangle$ in a two dimensional space by proposition ϕ and valuation ν. The state region of $s' = \langle s, \nu \rangle$, denoted $[s']$ with $s' \in S'$, is defined by $[s'] = \langle s, [\nu] \rangle = \{\langle s, \nu' \rangle \mid \nu' \in [\nu]\}$. Any movement among nodes inside this set doesn't change the validity of the state proposition, which is only affected by an inter-region transition. Hence, the clock valuations $Eval(C)$ and the state labels determine a finite number of equivalence classes that are the basis of the model checking process. The aim of the algorithm presented in this section is the execution of the classic CTL model checking algorithm over the regions of this last finite structure, called Region Transition System (RTS) and that we will define later. The objective of the following transformations is to translate the problem of solving a TCTL equation to the standard problem of solving a CTL formula.

The process of verifying a formula ϕ is divided into five big steps, which can be summarized in pseudo code as explained in Algorithm 5 [13]. In brief, it starts transforming a TCTL formula ϕ into a CTL equation ϕ' by deleting the timed

Algorithm 5 Algorithm $Sat\,(M, \phi)$

Require: $TA = (S, S_0, R, L, AP, C, \lambda, \tau)$ is a timed automaton
Require: ϕ is a TCTL formula
Ensure: $TA \vDash \phi$

- Transformation of ϕ into ϕ' by the elimination of timed restrictions $\sim c$ of the formula, for example, in $\psi_1 \mathbf{U}_{\sim c} \psi_2$
- Migration of the timed restrictions $\sim c$ to a state condition with z an external clock, for example, in $\phi' = \psi_1 \mathbf{U}(\psi_2 \wedge (z \sim c))$
- Determination of the equivalence classes under $\equiv^*$
- Construction of the region transition system M=RTS(TA,ϕ) equivalent to the timed automaton TA
- Application of the CTL model checking algorithm $Sat\,(M, \phi')$ to check $M, \langle s, [\nu] \rangle \vDash \phi'$

restrictions $\sim c$. The temporal information must be reincorporated to the verification process in the second step. To this end, we introduce an extra clock $z \notin C$ to the model and specifications for tracking the time elapsed in traversing the regions of the region transition system. The clock z is initialized to 0, it is never reset along the path and it is updated consistently with the other clocks. Next, the formulas are rewritten in terms of state conditions with z instead of temporal restrictions over the path operators. The states of the RTS are labeled with atomic propositions representing the clock constraints that are fulfilled in that state. Later, we calculate the equivalence classes according to $\equiv^*$. The equivalence class $\equiv^*$ is an extension of $\equiv$ that includes the set of clocks $C^* = C \cup \{z\}$. Finally, we construct the region transition system associated to the timed automaton TA and formula ϕ, and we apply the algorithms of CTL.

A region transition System RTS(TA,ϕ) is a transition system whose nodes are clustered by time and by property according to the restrictions they satisfy. It depends implicitly of the formula ϕ that we try to verify: its maximal constants $\sim c$ are of relevance to the clock equivalence only. In case the region transition system does not depend on (the maximal constants occurring in) ϕ, we simply write RTS(TA). The RTS is defined as:

Definition 13 (Region Transition System)**.** *Given a TCTL formula ϕ, the region transition system $RTS = (S^*, S_0^*, R^*, L^*, AP^*)$ is an enhanced transition system $TS = (S', S_0', R', L', AP')$ from a timed automaton $TA = (S, S_0, R, L, AP, C, \lambda, \tau)$*

where:

- $S^* = S'/\equiv$ *is a finite set of states obtained by the equivalence classes of* $Eval(C)$ *and* S*,*
- $S_0^* = \{[s'] \mid s' \in S_0'\}$ *is the set of initial states,*
- $R^* \subseteq S^* \times S^*$ *is a total transition relation between states defined as:*
  - *A* discrete transition $(\langle s_i, \alpha\rangle, \langle s_j, \alpha'\rangle)$ *such that* α *is an equivalence class and* $r_i = (s_i, s_j) \in R$*. The equivalence class* α *is not a boundary class. The equivalence class* α' *is the equivalence class* α *with some clock that are reset. The clocks* $\lambda(r_i)$ *are reset, i.e.,* $\nu(x) = 0$*,* $\forall x \in \lambda(r_i)$ *and* $\forall \nu \in \alpha$*. The clock valuation* ν *satisfies a firing condition* $\tau(r_i)$*.*
  - *A* delay transition $(\langle s_i, \alpha\rangle, \langle s_i, succ(\alpha)\rangle)$ *where* α *is not an end class.*
- $L^*(\langle s, [\nu]\rangle) = L(s) \cup \{g \in AP^* \backslash AP \mid [\nu] \vDash g\}$ *is the labeling function.*
- $AP^* = AP \cup CC(C) \cup CC(\phi)$ *with* $CC(C)$ *the clock constraints of the timed automaton TA and* $CC(\phi)$ *the clock constraints introduced by the formula* ϕ*.*

Therefore, the verification of a TCTL formula ϕ in a timed automaton TA is reduced to the verification of a CTL formula ϕ' in RTS(TA,ϕ). Once the region transition system is obtained, a CTL-like model checking algorithm is applied. Thus, the complexity of verifying a TCTL formula ϕ against a timed automaton TA is linear in $|\phi|$ and $(|S| + |R|)$, with $|\phi|$ the number of logical connectives and temporal operators of the formula, and $|S|$ and $|R|$ the number of states and transitions in the region transition system RTS respectively. More generally, the complexity is $\Theta((|S| + |R|) * |\phi|)$.

The commercial software UPPAAL is an example of a model checking tool that implements these algorithms and it is prepared for managing timed transition systems and logics. The open source NuSMV and PRISM also include variants of these logics and systems. In fact, NuSMV assumes that each transition of the transition system consumes a clock tick. Finally, PRISM implicitly embeds the notion of time in probabilistic timed automata and Markov chains.

11.4. Conclusions

The main contribution of this chapter has been the study of phylogenetic properties that need the use of explicit time in the specifications. Firstly, we have motivated the necessity of this extension with real examples; for instance, the inspection of the date of appearance of certain phenotypes in the branches of the

phylogenetic tree such as the peculiar traits of the Tibetan populations or woolly mammoths. Next, we have collected the requisites for augmenting the model checking framework and managing this new kind of restrictions in the specifications. To this end, we have introduced TCTL for solving these problems. TCTL is an existing temporal logic with real time capabilities: it provides a syntax, semantics, model checking algorithms and tools that fit with our requirements. After the presentation of the logic, we have shown how to translate a phylogenetic property to this particular notation. This approach shows the conceptual feasibility of our proposal for defining hypothesis with temporal restrictions and distances. Finally, we have presented the algorithms and software tools that can process this kind of formulas.

The estimated theoretical complexity of verifying this type of specifications is $\Theta((|S| + |R|) * |\phi|)$, with $|\phi|$ the number of logical connectives and temporal operators of the formula, and $|S|$ and $|R|$ the number of states and transitions in the region transition system RTS associated to the phylogenetic tree respectively. The experimentation with this framework is delayed to further chapters. In particular, Chapter 14 and Chapter 15 evaluate phylogenetic properties that mix probabilities with temporal constraints, which poses a more complex, versatile and realistic framework for the analysis of phylogenies. The time and memory consumption observed in the experiments over there places an upper bound for the properties studied here. Following to this chapter, Chapter 12 continues with the analysis of temporal distances between sets of tree states but from a complementary perspective: now, the output result of a phylogenetic property is an integer indicating a distance.

Chapter 12

Computing Distances Between Symbolic Objects

Time is the longest distance between two places.
— Tennessee Williams, *The Glass Menagerie*

12.1. Introduction

The result of some phylogenetic properties is not necessarily constrained to a boolean result. As well as checking the validity of a formula with *qualitative* model checking, biologists often need *quantitative* information as output of an evaluation. For example, the increasing number of inferred phylogenies and procedures for computing them leads to the necessity of incorporating comparative methods that help to analyze the structural differences of the trees. These comparative methods are based on distances between sets of states, the number of movements for converting one tree in another, or more generic topological measures (tree balance and asymmetry) involving the calculation of numerical values. In fact, the use of topological measures allows the comparison of heuristics for building phylogenies as well as the discrimination of trees violating phylogenetic restrictions [130, 158, 159].

Thus, the calculation of quantitative information is a key question for comparing and discarding trees. The identification and selection of the elements over which we will evaluate an arithmetic function comprises the first step. Here, the aim of this chapter is twofold. First of all, we focus on the symbolic manipulation of *phylogenetic objects*: we try to characterize a particular set of states of the tree satisfying certain restrictions imposed by a formula of temporal logic.

Later, we show that the model checking framework supports the manipulation of these objects and the computation of distances among them. Previously in

this thesis, the analysis of phylogenies with model checking was centered primarily on the specification and evaluation of boolean hypothesis over the whole tree. Now, we focus on the characterization of a particular set of tree states satisfying certain restrictions imposed by a formula of temporal logic. For instance, the set of nodes of the phylogeny belonging to a clade represents one of these objects. The computation of the distance between two clades is important for discerning if the tree is coherent or aberrant with respect to a hypothesis of the phylogenetists.

Obviously, the inclusion of numerical restrictions in the computations requires the incorporation of quantitative capabilities in current temporal logics and model checking algorithms. The presentation of the TCTL logic (Chapter 11) provides a syntax and semantics for the definition and inquiry of properties with explicit integer values in the specifications. But the distances introduced in the specifications can also be discovered by the model checking algorithms in some cases. Following the same classification introduced for the labeling of the branches with quantitative information, the numerical output of model checking can be grouped in distance (or temporal) values and probabilities.

In this chapter, we focus exclusively on returning temporal distances between phylogenetic objects defined symbolically as result of the model checking process. We show how some common phylogenetic metrics can be formulated with temporal logics. Alternatively to all the measures involving time and distances, we center our attention for probabilities in Chapters 14–15. There, we initially study the truth of a boolean property over a phylogenetic tree with a particular instance of the probability threshold. Most of the usual model checking tools are also capable of automatically infer the upper (maximum) and lower (minimum) probability bounds.

12.2. Returning Time Distances as Output of the Model Checking Procedure

Firsts extensions of classic model checking algorithms cater for the computation of quantitative bounds. Normally, they count the maximum or minimum number of (timed) steps between groups of elements, considered as sets of states characterized by a logical formula [35]. These modifications in the logic result in the addition of MAX and MIN operators to the repertory of instructions of common model checkers such as NuSMV, which returns numerical values as output of their evaluation. Those operators belong to the syntax and semantics of the RTCTL logic, one of the firsts time extensions to CTL that assumes a weight of one time step in each branch of the tree.

As root, common ancestors, leaves and clades are easily characterized sym-

bolically with a proper labeling of the states and querying logic (see Chapter 4, Section 4.2), this feature provides a precise way for the computation of branch lengths and distances. A clear example of application corresponds to the calculation of the chronological date of divergence between populations in the Tibetan Plateau, where each one is characterized by its particular polymorphisms.

Moreover, most tree (im)balance and asymmetry metrics for testing the quality of a phylogeny use topological distances between the root and subtrees or leaves. Sometimes, they discriminate the path lengths of left and right descendants (whose direction is marked when labeling the states of the tree) [136, Table 2]. For instance, the computation of the mean topological distance M from the root to the leaves is:

$$M = \frac{1}{N}\Sigma_{i \in L} M_i$$

where N=|S| is the number of states in the tree (Kripke structure), L is the set of leaves and M_i is the topological distance, i.e., the number of intermediate nodes between the i^{th} leaf and the root. In the particular case of acyclic directed graphs such as trees, a pair of states are connected with a single path and thus the MAX/MIN distance between nodes is identical. Any of the next two expressions in NuSMV will return the same value for the length between the i^{th} leaf and the root:

```
COMPUTE NAME Mi := MAX [id=root, id=i]
COMPUTE NAME Mi := MIN [id=root, id=i]
```

Storing the partial results of each `Mi` and applying the correspondent addition to all the elements, we obtain the aforementioned metric. The mean topological distance is used for the calculation of the statistical variance or in the next metric:

$$\Sigma_{i \in L} M_i / 2^{M_i}$$

As the range of output values of `Mi` is well-determined by the number of nodes of the tree, the balance of the tree is known by the examination of this result. Additional metrics are defined in terms of maximum distances between the root and one of the leaves:

$$\Sigma_{i \in I} Z_i^{-1}$$

where I is the set of internal nodes. The Z_i values are computed through:

```
COMPUTE NAME Zi := MAX [(id=root) & !terminal, terminal]
```

The tree states are enriched with boolean variables in order to identify internal and terminal nodes. Hence, it is patent that distance-based phylogenetic inference methods can take advantage of this particularities and metrics for the construction of a tree [176].

The previous approach is motivated by systems modeled with timed automaton and discrete time logics where the branches are labeled with unit time stamps. Further extensions should focus on the computation of MAX and MIN paths in branches tagged with more flexible time stamps or lengths, i.e., continuous time with $t \geq 1$.

In addition, new and more complex quantitative functions can be applied over the result set characterized by a symbolic function. Powerful quantitative model checking tools such as PRISM allow us to count the number of states satisfying a property or relation, the average of a numerical label in those states or the maximum/minimum value. Supplementary balance metrics may take advantage of these functions. For example, the metrics involving the count of leaves in left (l_j) and right (r_j) subtrees descendant of the ancestor j:

$$\frac{2}{(N-1)(N-2)} \Sigma_{j \in I} |r_j - l_j|$$

PRISM can evaluate the equation using the subsequent steps. Figure 12.1 represents the phylogenetic tree depicted in Figure 2.1 expressed in the notation of the PRISM model checker. Each state is labeled with an identifier and two boolean values: the first one tells if it is a leave or internal node, and the second one tells if the node is the left or right descendant of his parent in case of a binary tree. The phylogenetic tree is defined backwards: from the leaves to the top, plus a self-loop in the root. The name of the root is changed to `X1` in order to avoid collisions with the reserved word **X** (next operator).

The counting of the terminal leaves belonging to the left subtree of the root is achieved executing the following expressions. The first step returns the identifier of the left son of the root, and the second step calculates the number of terminal leaves that has the previous node as its root; i.e, the leaves belonging to the left subtree. The computation of the number of leaves in the right subtree is symmetric.

```
"direct_left_son": filter(print, left & P>=1 [X id=X1])
filter(count, P>=1 [F "direct_left_son"], terminal)
```

Additionally to these metrics, the direct comparison of phylogenetic trees is a key question. Robinson-Foulds [146] is one of the most common and popular metrics in comparisons of (un)rooted trees, but there are many more [102]. Another phylogenetic comparative method called SPR, that calculates the distance of two trees as the minimal number of moves that transforms one tree into the other, is

```
// Markov decision process
mdp

const int X1 = 1;
const int Y = 2;
const int Z = 3;
const int R = 4;
const int S = 5;
const int ext = 6;

module BACKWARD_TREE
// Initial state
id: [X1..ext] init ext;
terminal : bool init false;
left : bool init false;


// Root
[trans] id=X1 -> (id'=X1) & (left'=false) & (terminal'=false);

// Internal nodes
[trans] id=Y -> (id'=X1) & (left'=false) & (terminal'=false);
[trans] id=Z -> (id'=X1) & (left'=false) & (terminal'=false);

[trans] id=R -> (id'=Y) & (left'=true) & (terminal'=false);
[trans] id=S -> (id'=Y) & (left'=true) & (terminal'=false);

// Terminal leaves
[trans] id=ext -> (id'=R) & (left'=true) & (terminal'=true);
[trans] id=ext -> (id'=S) & (left'=false) & (terminal'=true);
[trans] id=ext -> (id'=Z) & (left'=false) & (terminal'=true);
endmodule
```

Figure 12.1: Description of a phylogenetic tree in PRISM syntax.

already implemented using logics in SAT [28]. This fact encourages the feasibility of computing distances using temporal logics.

The complexity of model checking with quantitative outputs is directly related to the complexity of CTL model checking because the procedure uses the

satisfiability sets obtained by the original model checking algorithms and applies quantitative functions over them. The complexity of the functions evaluated over the result set determines the final complexity of the system, while the generation of that set is proportional to the size of the formula.

12.3. Conclusions

In this chapter, we have emphasized the hidden features of the model checking framework for selecting and manipulating sets of states symbolically. In particular, the model checking algorithms are capable of computing numerical values (either integers or real numbers) as output result: they apply arithmetic functions or count the accumulated weight in the branches of a path separating sets of states. Originally, we have focused on the calculation of temporal distances between sets of states or phylogenetic objects defined in this way, but in future chapters we will inspect the estimation of probabilities and ratios as well. Our main contribution has been the application of these techniques for the extraction of phylogenetic measures, mainly the balance and asymmetry of trees. We have illustrated the description of the formulas using the notation of temporal logics. The examples presented here open the possibility of implementing a wide range of phylogenetic computations.

In sum, model checking is a complete and heterogeneous framework for modeling systems, specifying and evaluating phylogenetic hypothesis, and manipulating symbolically sets of states according to the properties they fulfill.

Chapter 13

Conclusions

A good decision is based on knowledge and not on numbers.

— Plato

In this part, we have shown the heterogeneity in the specification of hypothesis and the limitations of boolean temporal logics for expressing properties involving complex phylogenetic relations or labels. In fact, we have motivated the extension of model checking to quantitative domains using real examples, mainly for the case of time and probabilities. After presenting the necessity of this quantification, in Chapter 10 we have cataloged the properties according to the kind of quantitative data they will handle. To this end, we have proposed the inclusion of temporal and probabilistic labels in the branches because of its natural interpretation in the phylogeny. The complexity of the new temporal logics managing time and probabilities also constrains the decidability and computational costs.

Next, Chapter 11 develops the concepts and theory for evaluating properties with dense (real) time in the specifications. There, we have recycled and adapted an existing temporal logic, called Timed CTL (TCTL), as well as the data structures and model checking algorithms for manipulating this kind of formulas. More in detail, we have shown the description of a phylogenetic property using this logic, the association of a weighted phylogenetic tree to a timed transition system, and the resolution of these specifications by a model checking algorithm. The study of probabilities is relegated to Part V.

Finally, the introduction of time and probabilities is not limited to the addition of labels in the phylogenetic tree or numeric variables in the specifications. The computation of distances between states of the tree and the calculation of the statistical support of a phylogeny need the modification of the model checking algorithms for returning numerical values. The capability of our formal framework for the symbolic manipulation of states using temporal logic formulas has been an important factor. For example, Chapter 12 resolves the estimation of topological

measures (e.g., tree balance and asymmetry) by applying arithmetic functions over sets characterized with a particular equation. The evaluation of maximum likelihood estimations is relegated to Chapter 15.

Part V

Quantitative Extensions of Kripke Structures and Logics for the Analysis of Phylogenies: Approaching to Probabilistic Properties

Probabilities play an important role for the study of phylogenetic properties. They are applied in two complementary domains. From the point of view of the topological coherence, the additional labels of the tree mark the statistical consistency of the branches. Probabilities tell the number of bootstrapped or inferred trees that support that specific shape. There exists a range of statistical methods that compute these support values, which are usually expressed with percentages between $[0, 1]$. One of the most common processes for scoring and comparing trees numerically is based on the computation of maximum likelihoods estimations [69].

Not only do they play an important role for the comparison of trees through the evaluation of the tree coherence, but also probabilities are useful for showing the prevalence of genetic patterns inside populations. Omitting the concrete support values in the branches of the phylogeny, we can center entirely on the study of the labels in the tree states. Even assuming that all the branches are generated with equal probability, the likelihood of finding a particular label (mutation or phenotype) in a state is not homogeneously distributed in the tree and it will variate with respect to each section of the phylogeny. Hence, the calculation of percentages of satisfaction of a property inside a clade or along a path of the phylogeny enlarges the expressiveness during the verification.

In any case, the necessity of extending the formal framework with probabilities for both circumstances is evident. Although they share common roots, the operations needed for the computation of the topological support slightly differ from the operations needed for the computation of probabilities in reachable states. The first problem usually evaluates continuous time equations for the elaboration of maximum likelihood estimations, while the second problem is simplified with the use of discrete time distances in the paths of the phylogeny. The application domain changes with the type of time considered for the computation of probabilities. That is the main reason why the content of probabilistic model checking is divided in two sections.

The aim of Chapter 14 is to present the main concepts of probabilistic model checking with a probability function that depends on the discrete time associated to the branches and, later, increment the complexity of probabilistic model checking with continuous time in Chapter 15. In particular, the first chapter is devoted to analyze the variation of a state property in several zones of the tree and compute the probabilities of satisfaction, while the formalism introduced in the second chapter is applicable to the estimation of maximum likelihoods.

Chapter 14

Discrete Time Probabilistic Transition Systems and Logics

The probability of life originating on Earth is no greater than the chance that a hurricane, sweeping through a scrapyard, would have the luck to assemble a Boeing 747.
— Richard Dawkins, *The God Delusion*

14.1. Introduction

Previously, we have seen that the phylogenies are occasionally enriched with time labels or weights in the edges. This knowledge is useful for learning complex properties about the evolution, for instance, the estimation of the temporal point of divergence between species [17] or the diaspora of human populations [36]. The extension of the phylogenetic properties in Table 4.1 with time and probabilities increases the expressivity of biological hypothesis.

Take the following disease as a clarifying example. The lactose intolerance in adults is a chronic disease caused by the inhibition of the lactase gene after the breastfeeding and childhood. The inability for processing the milk and its derivations is not homogeneously distributed in the human population. While in some African pastoralist groups in North/East Africa and the northern cultures of Europe their stock breeding tradition and diet motivated an evolutionary adaptation to digest the milk ($> 70\%$ of tolerance), the percentage of acceptation decreases in the rest of areas and ethnic groups [160, 92]. In addition, the phenotype in Europe and Africa appeared at a different epoch and the point mutations that regulate the activation of the lactase persistence are disparate [163, 97]. Some illustrative questions that we desire to ask to the phylogeny, and that are expressed below,

require the addition of time to the branches of a population tree. The time allows the estimation of the divergence points between individuals or mutations, while the probability of the lactose persistence in different zones is calculated through the study of the distribution of the point mutations that regulate the phenotype. The questions are:

1. What is the rate of lactase persistence in a population? i.e., do their members define a characteristic haplogroup? and in that case,

2. Which polymorphism, among the multiple activators and inhibitors of the lactase gene, is the most frequent over there? and finally,

3. When did this phenotype approximately start to be predominant? i.e., does this date mark an important event in the diet, culture or migration of that population?

These questions ask about the time (dates) and probabilities (frequencies/rates) stored in the branches of the tree. Besides, the deductive process that answers the queries also needs the manipulation of quantitative information. Thus, we must introduce a logic, a transition system and a model checking algorithm capable of expressing and managing these kind of questions. The notion of time introduced here matches with the concept of evolutionary or chronological clock.

This chapter is organized as follows. After this introduction, Section 14.2 presents the syntax of the discrete time probabilistic logic. Secondly, Section 14.3 introduces the model checking algorithm for PCTL. Next, Section 14.4 focuses on the existing tools that handle this kind of logic and data structures. Finally, Section 14.5 draws the conclusions of this chapter.

14.2. Discrete Time Probabilistic Logic and Structure

In this section we are considering a phylogenetic tree enriched with numerical information that tells the probability of selecting a branch descending from an internal node. Therefore, we can answer questions like: what is the probability of reaching a set of states of the tree from the root? The logic and data structure defined here settle the basis for future updates and extensions for continuous time systems in Chapter 15. Stochastic systems generally use Markov chains as the underlying data structure that provides semantics to the verification process [106]. Discrete time Markov chains capture the essentials of probabilities between states of the tree and implicitly associates an unit time step to every transition of the system.

Definition 14 (Discrete time Markov Chain)**.** *A discrete time Markov chain is a finite transition system represented by a tuple* $M = (S, S_0, \mathbf{P}, L)$*, where:*

- S *is a finite set of states,*
- $S_0 \subseteq S$ *is the set of initial states,*
- $\mathbf{P} : S \times S \rightarrow [0, 1]$ *is the transition probability matrix that indicates the probability of outgoing from state* s_i *to a certain state* s_j *satisfying* $\Sigma_{s_j \in S} P(s_i, s_j) = 1$*, and*
- $L : S \rightarrow 2^{AP}$ *is the labeling function that associates each state with the subset of atomic propositions that are true of it.*

A phylogenetic tree is assimilated to a discrete time Markov chain making the corresponding associations of states to the definition of phylogeny (Definition 3). The leaves are labeled with the genome information of the population or specie they represent, plus additional data when necessary. Each branch of the phylogeny is mapped to an element $\mathbf{P}(s_i, s_{i+1}) > 0$ of the transition probability matrix. This value gives the probability of making a transition from state s_i to state s_{i+1} in one time step. By default, we assume that all the descendants s_j of a state s_i are equiprobable ($\mathbf{P}(s_i, s_j) = 1/n$ with n the number of successors), but this value can be adjusted. The terminal leaves modeled with self-loops in the Kripke structure of a branching-time phylogeny are represented in the transition probability matrix by a single transition going back to the same state with probability 1.

For any set of infinite paths Π starting in the initial state s_0, the subset $\Pi(\pi_n)$ selects the paths $\pi \in \Pi$ whose prefix equals to the finite sequence $\pi_n = s_0 s_1 s_2 \ldots s_n$ of length $n + 1$ states. The set of infinite sequences sharing the prefix π_n has probability $Pr(\Pi(\pi_n)) = \mathbf{P}_\Pi(\pi_n)$. The probability $\mathbf{P}_\Pi(\pi_n)$ is calculated as the product of probabilities for each intermediate transition, except for paths with unitary length in which case $n = 0$, $\pi_0 = s_0$ and $\mathbf{P}_\Pi(\pi_0) = \mathbf{P}_\Pi(s_0) = 1$:

$$\mathbf{P}_\Pi(\pi_n) = \begin{cases} 1 & \text{if n = 0} \\ \mathbf{P}(s_0, s_1) \cdot \mathbf{P}(s_1, s_2) \cdot \ldots \cdot \mathbf{P}(s_{n-1}, s_n) & \text{otherwise} \end{cases}$$

Bayesian model checking methods allow for analyzing stochastic systems [100]. Probabilistic CTL (PCTL) [150, 88, 13] and Continuous Stochastic Logic (CSL^{TA}) [59] help to formulate conditions on a discrete or continuous time Markov chain, respectively. The properties are referred to state formulas (ϕ) or path formulas (Φ). Besides, they allow enriched queries of temporal formulas such as $\mathbb{P}_{\sim\lambda}(\Phi)$. Given an initial state s and a comparison $\sim \in \{<, \leq, =, \geq, >\}$, the operator $\mathbb{P}_{\sim\lambda}(\Phi)$ returns true if the probability for a set of paths satisfying Φ is $\sim \lambda$, with $\lambda \in [0, 1]$.

Definition 15 (Probabilistic Computation Tree Logic)**.** *A temporal logic formula ϕ is defined by the following grammar with the minimal set of operators, where $p \in AP$ and $k \in \mathbb{N} \cup \{\infty\}$:*

$$\begin{aligned} \phi &::= true \mid p \mid \neg\phi \mid \phi \vee \phi \mid \mathbb{P}_{\sim\lambda}[\Phi] \\ \Phi &::= \mathbf{X}\phi \mid [\phi \mathbf{U}_{\leq k} \phi] \end{aligned} \tag{14.1}$$

The formulas are checked against a structure M considering all infinite paths $\pi \in \Pi$ from a certain state s_0. Notice that $M, s_0 \vDash \phi$ means that s_0 satisfies ϕ. The semantics of well-formed formulas is as follows (let $\pi = s_0 s_1 s_2 \ldots$):

- $M, s_0 \vDash p \Leftrightarrow p \in L(s_0)$,
- $M, s_0 \vDash \neg\phi \Leftrightarrow M, s_0 \nvDash \phi$,
- $M, s_0 \vDash \phi \vee \psi \Leftrightarrow M, s_0 \vDash \phi$ *or* $M, s_0 \vDash \psi$,
- $M, s_0 \vDash \mathbb{P}_{\sim\lambda}[\Phi] \Leftrightarrow Prob(M, s_0, \Phi) \sim \lambda$,

The calculation of the probability $Prob(M, s_0, \Phi)$ requires the identification of the infinite paths π satisfying the path formula $M, \pi \vDash \Phi$:

- $M, \pi \vDash \mathbf{X}\phi \Leftrightarrow M, s_1 \vDash \phi$
- $M, \pi \vDash [\phi \mathbf{U}_{\leq k} \psi] \Leftrightarrow \exists 0 \leq i \leq k,\ \forall 0 \leq j \leq i : (M, s_i \vDash \psi) \wedge (M, s_j \vDash \phi)$

This set, $\{\pi \in \Pi \mid M, \pi \vDash \Phi\}$, can be obtained by the union of finitely many pairwise disjoint subsets $\Pi(\pi_n)$ by [106, Definition 3], each one characterized by the finite prefix π_n of all infinite sequences of the set. Therefore, $Prob(M, s_0, \Phi) = Pr\{\pi \in \Pi \mid M, \pi \vDash \Phi\} = \Sigma_{\pi_n} Pr(\Pi(\pi_n))$ computes the probability as the summation of probabilities in all possible prefixes π_n by [106, Theorem 1].

These logics usually support timed transitions in the **U** operator. The notion of time in a Markov chain falls within the concept of state distances. Each state transition of the discrete time Markov chain involves an unit time step. A mapping between the chronological time and state distances allows the inference of the evolutionary speed in the branches of the phylogenetic tree. The computation of time and probabilities are embedded in the model checking algorithm. Timed variants of the modal operators **F** and **G** are obtained via **U** as $\mathbf{F}_{\sim c}\phi = true\, \mathbf{U}_{\sim c}\phi$ and $\mathbf{G}_{\sim c}\phi = \neg\mathbf{F}_{\sim c}\neg\phi$. Instead of writing intervals explicitly, sometimes they are abbreviated with inequalities. For example, $\mathbb{P}_{\leq 0.5}[\Phi]$ denotes $\mathbb{P}_{[0,0.5]}[\Phi]$.

Far beyond the use of discrete or continuous time, the main difference between PCTL and CSL^{TA} syntax is the substitution of the long-run operator $\mathbb{L}$ in PCTL

by a steady-state operator $\mathbb{S}$ of CSL^{TA}. These advanced operators are unnecessary for the phylogenetic problems we are trying to solve, but they could be included in the future if necessary [106, 13].

By now, we can translate the questions presented in the motivation example of lactose into the PCTL syntax. In a phylogenetic tree, the tips correspond to individuals of disjoint populations whose states are tagged with their DNA and a boolean indicating if they are lactose (in)tolerant. The internal nodes of the inferred ancestors are labeled with their estimated DNA sequence and lactose phenotype as well. The following equation asks if there exists an ancestor ($\mathbb{P}_{>0}$) at distance 3 or above from the initial state ($\mathbf{F}_{\geq 3}$) that is the root of a population with lactase persistence over 70% ($\mathbb{P}_{\geq 0.7}\,[\mathbf{F}_{\geq 0}\, lactose\_tolerant]$). The members of a population, including the leaves and internal nodes, are reached through $\mathbf{F}_{\geq 0}$.

$$\mathbb{P}_{>0}\,[\mathbf{F}_{\geq 3}\ (\mathbb{P}_{\geq 0.7}\,[\mathbf{F}_{\geq 0}\, lactose\_tolerant]\,)] \tag{14.2}$$

The outer restriction $\mathbb{P}_{>0}\,[\mathbf{F}_{\geq 3}]$ corresponds to the question 3 of the motivation. It searches for an internal node from which the phenotype starts to be predominant after a certain date since the phylogenetic root. The inner formula $\mathbb{P}_{\geq 0.7}\,[\mathbf{F}_{\geq 0}\, lactose\_tolerant]$ answers the question 1 about the rate of lactase persistence in a population. Finally, the addition of a genetic marker in this place inside the $\mathbb{P}_{\geq 0.7}$ equation helps to investigate the relation between a polymorphism and phenotype (question 2). The evaluation of the formulas needs the algorithm introduced in the next section.

14.3. Algorithm for PCTL Model Checking

The evaluation of formulas written in PCTL over discrete time Markov chains drastically differs from the evaluation of formulas written in CSL^{TA} over continuous time Markov chains due to the divergent interpretation of time. This point clearly arises during the verification process, which adds semantics to the specified formulas. The model checking algorithms for managing and solving PCTL or CSL^{TA} formulas in stochastic systems are mainly identical to those of classic model checking except for the resolution of $\mathbb{P}_{\sim\lambda}[\phi]$, i.e., the next and until operators with probability thresholds. In short, the recursive algorithm of model checking incorporates the new sentence:

$$Sat(\mathbb{P}_{\sim\lambda}\,[\Phi]) = \{s \in S \mid Prob(M, s, \Phi) \sim \lambda\}$$

$\mathbb{P}_{\sim\lambda}[\mathbf{X}\phi]$ formula. In PCTL, the probability of satisfying the next operator requires the probabilities of the immediate transitions from s. It is resolved by:

$$Prob(M, s, \mathbf{X}\phi) = \Sigma_{s' \in Sat(\phi)} \mathbf{P}(s, s')$$

$\mathbb{P}_{\sim\lambda}[\psi \mathbf{U}_{\leq k}\phi]$ **formula.** The computation of the probability for the until operator depends on the value of k. For the case of $k \in \mathbb{N}$, then $Prob(M, s, \psi \mathbf{U}_{\leq k}\phi)$ is equal to:

$$\begin{cases} 1 & \text{if } s \in Sat(\psi) \\ 0 & \text{if } k = 0 \text{ or } s \in Sat(\neg\phi \wedge \neg\psi) \\ \Sigma_{s' \in S} \mathbf{P}(s, s') \cdot Prob(M, s', \psi \mathbf{U}_{\leq k-1}\phi) & \text{otherwise} \end{cases}$$

When $k = \infty$, the until operator is analogous to the original until operator of CTL with semantics of infinite paths. That is, $Prob(M, s, \psi \mathbf{U}_{\leq \infty}\phi)$ can be rewritten as $Prob(M, s, \psi \mathbf{U}\phi)$ and it equals to:

$$\begin{cases} 1 & \text{if } s \in Sat(\psi) \\ 0 & \text{if } k = 0 \text{ or } s \in Sat(\neg\phi \wedge \neg\psi) \\ \Sigma_{s' \in S} \mathbf{P}(s, s') \cdot Prob(M, s', \psi \mathbf{U}\phi) & \text{otherwise} \end{cases}$$

The time complexity of verifying a PCTL formula ϕ against a discrete time Markov chain is linear in $|\phi|$ and polynomial in size of S, with $|\phi|$ the number of logical connectives and temporal operators of the formula. More generally, the complexity is

$$\Theta(poly(size(S)) * t_{max} * |\phi|)$$

where t_{max} is the maximal step bound of a path subformula $\psi_1 \mathbf{U}_{\leq t}\psi_2$ of ϕ, with $t_{max} = 1$ if it doesn't contain any $\mathbf{U}$ subformula.

The evaluation of CSL$^{\text{TA}}$ formulas demands the upgrade of Markov chains structures for providing continuous time semantics. The computation of probabilities is assisted by a partial transformation of the continuous time problem to a discrete time problem. Hence, the operations realized by the model checking algorithms for the calculation of probabilities in PCTL are the basis for future extensions. The time complexity of verifying a PCTL formula also poses a lower bound to the complexity of analyzing a CSL$^{\text{TA}}$ formula. The introduction of these semantics and complexity are presented in Chapter 15.

14.4. Model Checking Tools and Experimentation

PRISM [124] is a generic model checking tool capable of handling probabilistic and timed specifications over Markov chains. Among its basic conceptions, PRISM checks if the probability of reaching a set of satisfiable states is up or below a

predefined threshold. There exists a considerable diversity of model checking tools with different performances and qualities [98]. Although the real performance depends on the particular structure of the model and specifications, PRISM offers Java portability, a powerful syntax for handling time and probabilities in models and specifications, and a good scientific community support. Besides, it is open source, which allows the modification and optimization of its code.

The model checking tool requires two input files for the verification process: a first file with the description of the model, and a second file with the specification of the properties. A description of the phylogenetic tree (Kripke structure with atomic propositions) in PRISM syntax musts be provided by the user as first input for the model checker. To this end, we precompute a sequence alignment and a phylogenetic tree. The data set used for this experimentation is synthetic. With this data set, we try to cover the spectrum of small phylogenies and analyze the cost of the evaluation of the lactose property over there. We have created random phylogenetic trees of up to 1000 tips using a Yule backward model [156]. For each tree size (number of tips), we have generated ten random trees and calculated the harmonic mean time. The DNA sequences have 50 bases with an homogeneous distribution of nucleotides.

The PRISM codification of the phylogeny follows the same idea presented for NuSMV in Section 5.3. Figure 14.1 shows the implementation of the branching-time phylogenetic tree of Figure 2.1 in PRISM code. The main module describes the topology of the evolutionary tree, where the names of the tree nodes (taxa) are defined numerically $(1, \ldots, 5)$. PRISM distinguishes between the current state and the next state using the quotation mark (’). The second part of the description consists of a function returning the DNA string associated to each node, which evolves in synchronization with the tree skeleton ([id1] tags in PRISM). The translation of the phylogenetic tree to the PRISM syntax has been performed automatically by a BioPerl script [153]. The script can be upgraded in order to include extra features such as the generation of multiple instances of phylogenetic trees, bootstrapping and so on.

We have evaluated the lactose formulas introduced in the motivation but enriched for the detection of polymorphisms in the DNA. The underlying objective consists of the identification of a correlated evolution between the lactose tolerance and patterns in the genome. Other studies such as [92] use cultural information for discovering this coevolution and the influence of a milk-based diet. The utilization of phylogenetic comparative methods and regression techniques establishes the essentials of this approach.

The probability threshold of the internal $\mathbb{P}_{\geq x}\left[\mathbf{F}_{\geq 0}\, seq[i] = j\right]$ ranges from $x \in [0.1, 0.9]$, with $i \in [1, 50]$ the position where we search for the polymorphism and j a certain nucleotide. Figure 14.2 plots the time required for the computation of

50×9 formulas corresponding to the expansion of i and x for all the columns of the alignment and probability bounds. All tests have been run on a Intel Core 2 Duo E6750 @ 2.66 GHz with 8 GB RAM and Linux.

PRISM performs well for the verification of the lactose formulas in small phylogenies because it follows a polynomial trend in time with respect to the number of tips. However, it requires the integration of new technologies and solutions to scale for larger phylogenetic trees and specifications. In fact, we desire to find the value of x, i and j parameters for which the verification of the equation returns true. The definition of patterns is a common procedure, which intuitively leads to parametric model checking [32]. Nonetheless, mining for knowledge without prior information requires a more or less thorough exploration of the structure, which can be combinatorial in some or all of its dimensions. Although inherently parallel, the exhaustive inspection of potential solutions involves the test of large sets of formulas and an intensive use of the topology and information of each state. The application of parametric model checking for model exploration is a future extension that will increase the potential of our framework.

Additionally, the phenotype is regulated by several polymorphisms and the likelihood of finding a certain nucleotide depends on the selected path. For those reasons, the model checking process demands the introduction of operators that automatically compute the intolerance rate and the probability of reaching a certain nucleotide. The calculation of parameters representing probabilistic bounds in the specifications are already managed by current model checking tools. The computational capabilities of PRISM grants the inference of maximum (minimum) probabilities of satisfying a property.

14.5. Conclusions

In this chapter we have motivated the extension for the analysis of phylogenies via model checking using quantitative information. We have enriched the phylogenetic tree with respect to the tree in Chapter 11, where we only use explicit time. Here, we have also proposed the inclusion of probabilities in the branches of the tree because of its natural interpretation in the phylogeny as the main novelty. In particular, we have presented a phylogenetic example based on the lactose (in)tolerance that needs these kind of quantitative information. To this end, we have introduced an extended logic and data structure adapted for probabilities and time together with the algorithms and computations for managing them. Our first goal has been the increase of the logical capabilities for querying about the date of appearance and degree of distribution of mutations and phenotypes.

Next, we have experimented with synthetic data in order to prove the feasibility of our approach with existing probabilistic model checking tools. PRISM is a

generic model checking tool that performs well for small phylogenies in polynomial time with respect to the number of tips. The tool is independent of the application domain: it automatically verifies any proposition expressed with temporal logic over a model of the system. However, it requires the integration of new technologies and solutions to scale for larger phylogenies and specifications due to the particularities of the phylogenetic analysis. The distribution of the Markov chain structure, the parallelization of the formula verification with the computation of probabilities, and the integration of PRISM with the atomic propositions stored in an external database constitutes a further step. Some of these ideas has been already applied for standard CTL model checking in Chapter 7.

This work opens the door for the review of bigger phylogenies with properties similar to the lactose persistence. The modularity of our framework allows the evaluation of hypothesis and the comparison of results for a set of phylogenetic trees by only changing the tree file (the specification of the property remains constant).

```
// Markow decision process
mdp

// Nucleotids
const int a = 1;
const int c = 2;
const int g = 3;
const int t = 4;

module TREE
    // Initial state
    id: [1..5] init 1;

    // Root successors
    [id2] id=1 -> (id'=2);
    [id3] id=1 -> (id'=3);

    // Left clade successors
    [id4] id=2 -> (id'=4);
    [id5] id=2 -> (id'=5);

    // Self loops
    [] id=3 -> (id'=3);
    [] id=4 -> (id'=4);
    [] id=5 -> (id'=5);

    // In case of continious/discrete time markov chains (ctmc/dtmc)
    // or probabilistic timed automata (pta), 'x' and 'y' will
    // indicate probabilities or branch lengths
    // [] id=1 -> x:(id'=2)+y:(id'=3);
endmodule

module SEQUENCES
    // Root sequence
    x1 : [a..t] init a;
    x2 : [a..t] init a;
    x3 : [a..t] init g;

    // Sequences
    [id2] true -> (x1'= a)&(x2'= t)&(x3'= g);
    [id3] true -> (x1'= g)&(x2'= a)&(x3'= g);
    [id4] true -> (x1'= a)&(x2'= t)&(x3'= t);
    [id5] true -> (x1'= a)&(x2'= t)&(x3'= g);
endmodule
```

Figure 14.1: Mapping of the phylogenetic tree of Figure 2.1 in PRISM.

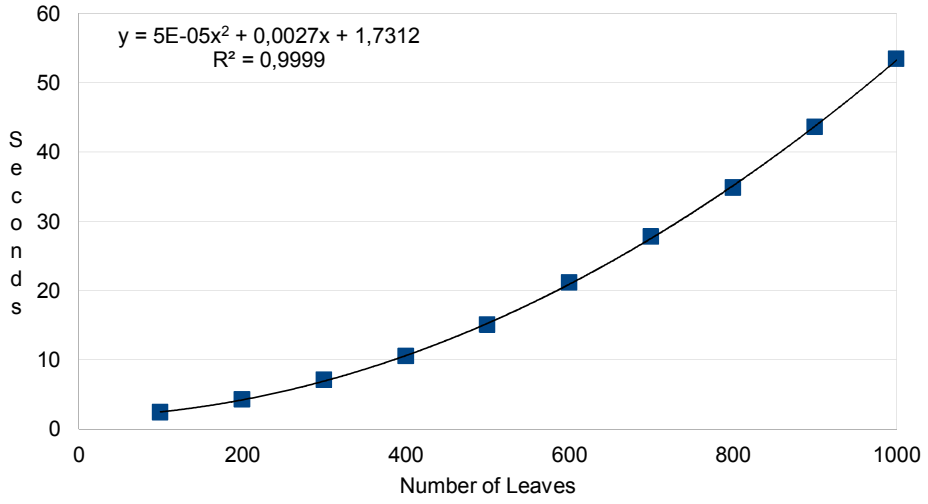

Figure 14.2: Time required for the verification of a set of probabilistic formulas with respect to the number of tips in the phylogeny.

Chapter 15

Continuous Time Probabilistic Transition Systems and Logics

Those are my principles, and if you don't like them... well, I have others.

— Groucho Marx

15.1. Introduction

In the preceding Chapter 10 we have introduced some of the common extensions over the branches of the phylogenetic trees, that is, the labeling of the phylogeny with time and probability tags. Besides, we have classified the phylogenetic properties according to these new criteria and the output results that produce their analysis. Now in this chapter, we dive into the problem of analyzing the statistical consistency of a phylogeny. More in detail, we study the computation of the support values that annotate the internal nodes and clades of the tree. This approach is similar to that presented in Chapter 12, where we computed temporal distances between sets of elements, but focusing on probabilities.

Here, the objective is to calculate the probability of generating a phylogenetic structure as the result of a certain model of speciation/extinction [152, 156]. That is, we evaluate a scoring function over the set of states of the tree. The value of this probability is essential for guiding the search for the best phylogeny in tree inference methods. Since no one knows the exact evolutionary process because of its complexity, the inferred phylogenetic tree with maximum likelihood is selected as the most representative and reliable evolution tree [69].

The maximum likelihood method is a statistical procedure for scoring the tree and finding the best topology that describes the phylogenetic relationships of an

alignment. It is one of the most important methods for assuring the reliability of a tree. It takes a phylogeny, an alignment and an explicit model of DNA substitution as input for the calculation of the maximum likelihood estimations (MLE). This method uses continuous time functions for the estimation of mutation probabilities, which motivates the extension of the probabilistic model checking presented in Chapter 14 with real numbers.

Thus, this chapter revolves around the work with models of DNA evolution for scoring the trees using maximum likelihood estimations. We show how the model checking framework can be used for computing probabilities. The first part of this chapter introduces the concepts and details about the models of DNA evolution and its use for calculating the likelihood score. Secondly, we detail the equations for computing the likelihood over phylogenies and comment the main limitations, drawbacks and simplifications. The third point defines the logic and algorithms for managing continuous time probabilities in model checking. Finally, the last section talks about the experimentation phase: it explains the implementation of the equations in a probabilistic temporal logic, their execution under a stochastic model checking tool and the analysis of the performance results.

15.2. Models of DNA Evolution

Birth-death Markovian models describe the macroevolution process of speciation [152, 156]. They are usually employed as basis for the generation of initial random trees fulfilling the particular topologies and properties that are supposed to appear in correct phylogenies. These trees, or those constructed by maximum parsimony methods, are later utilized as initial seed for the tree exploration phase by the evaluation of the maximum likelihood estimations [155].

Otherwise, the genome constitutes nowadays one of the most important traits of a specie for its study. The models of DNA substitution provide finer details about the genetic changes operating behind the macroevolution models. A DNA substitution model describes the process from which a sequence of characters switches into another one. In short, the models of DNA substitution capture how the genomic information is placed in each state of a tree inferred from a speciation model.

Often, the phylogenetists identify these DNA substitution models as the main source of the evolutionary process and then associate the notion of evolution history to the concept of mutation trace. Hence, the DNA substitution models are commonly referred to as *phylogenetic models* in the bio community: the phylogenies arise as the result of a spanning simulation of these mutation models embedding the complex process of speciation[1]. At this point, it is not surprising that biol-

[1]The consideration of DNA substitution models as phylogenetic models sometimes causes

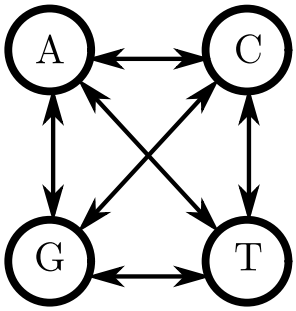

Figure 15.1: Model of DNA substitution.

ogists use an instance of a DNA substitution model to validate, using MLE, the tree of life that they obtained from a particular reconstruction method and input alignment.

In the context of DNA evolution, the phylogenetic tree would indicate the order of occurrence of the mutations along the history in the speciation process. Therefore, a given phylogenetic tree will reflect how the substitution model should act, showing the trace of changes in a long-term run of the model. In other words, the phylogeny *specifies* the behavior of the system, understood as the hypothetical drift of the DNA during the time (Figure 15.2). Thus, a phylogenetic tree, implicitly formulated as a succession of transitions and transversions of the nucleotides, is checked over a DNA substitution model. The validation process returns the likelihood of emerging that tree from that particular model[2].

The evolution drift leading to the substitution of nucleotides in the genome is usually *modeled* with a Markov chain (Definition 16, Figure 15.1). Different models of DNA evolution have been proposed in the literature, though the simplest ones consider a four state Markov chain with variants in the probabilities of transitions and transversions among nucleotides [174]. They are usually defined as templates, indicating with parameters the ratios and relations between bases. By now, there are highly optimized models for specific purposes (particular genes or organisms) including the special biochemical features of the nucleotides such as stability or expressiveness.

---

confusion with our terminology. Initially in previous chapters, we have identified the concept of phylogenetic models with the phylogenetic trees over which we can ask questions about the evolution.

[2]Transitions involve an interchange of bases of similar shape (purines A $\leftrightarrow$ G, or pyrimidines C $\leftrightarrow$ T). Transversions are interchanges of purine for pyrimidine bases.

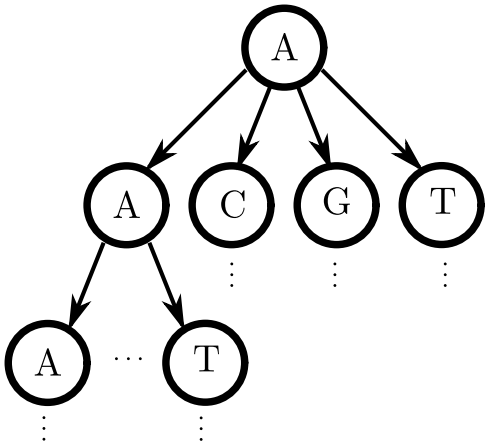

Figure 15.2: Unfolding of a model of DNA substitution.

15.3. Maximum Likelihood Estimation

Maximum likelihood is an optimal criterion that searches for computing the maximum probability L_i (Equation 15.2) of obtaining an specific configuration of phylogeny for the current model of DNA evolution. This is one of the most popular methods for guiding the tree reconstruction. The value L_i represents the confidence of a phylogeny with respect to the particular mutation model under consideration [69]. Due to the uncertainty of the *true* evolution drift generating the tree of life, a highest value for this score is supposed to increase the degree of confidence over the estimated tree: the likelihood values guide the refinement process. The maximum likelihood estimation works as a metric for comparing the topology of two (or more) dendrograms built over the same data set.

Given an inferred phylogenetic tree and an input alignment, the computation of maximum likelihoods returns the probability that a model of DNA substitution follows the trace imposed by the topology and sequences. Therefore, the selection of other phylogenies, alignments or alternative models of DNA substitution modifies the probabilities of the arrangements and returns different scores [117].

The computation of the likelihood score is carried out by the following formulas. At position l of a sequence with length n, the probability of changing from nucleotide i (in state X) to nucleotide j (in state Y) is denoted by the probability function $P_{ij}(d_{XY})$. The variable d_{XY} indicates the branch length between the states X and Y, which is a positive value for each pair of connected states. The Equation 15.1 corresponds to the overall likelihood for the tree in Figure 2.1. The probability function $P_{ij}(\cdot)$ depends on the selected DNA mutation model [69, 42, 170]. This probability function is determined by the model checking tool by analyzing the transitions of the DNA substitution model that it receives as

input. The expression $Z(l)$ denotes the base of the DNA in taxon Z at position l. The constant π_i is the expected frequency of the base i in the alignment.

$$L^l = \Sigma_{i=\{A,C,G,T\}}\Sigma_{j=\{A,C,G,T\}} \quad \pi_i P_{ij}(d_{XY})P_{iZ(l)}(d_{XZ}) \quad P_{jR(l)}(d_{YR})P_{jS(l)}(d_{YS}) \tag{15.1}$$

The Equation 15.1 leads to a summation of 4^N terms for alphabets with four nucleotides, and N the number of internal nodes of the tree. It can be factorized in order to avoid the recalculation of the same subformula multiple times. The Equation 15.2 rewrites the MLE equation in terms of the likelihood functions of the left and right subtrees for each internal node:

$$L^l_X = \Sigma_{i=\{A,C,G,T\}} L^l_{X,i} \tag{15.2}$$

$$L^l_{X,i} = \Sigma_j P_{ij}(d_{XY}) L^l_{Y,j} \cdot \Sigma_k P_{ik}(d_{XZ}) L^l_{Z,k} \tag{15.3}$$

Normally, the DNA of present-time taxa in the leaves of the tree has maximum likelihood (i.e., $L^1_{Y,A} = 1$). In opposition, the nucleic bases of the internal nodes are unknown and they shall be inferred with the mentioned formulas. Due to the uncertainty in the genome of the ancestral states, the estimation of the likelihood in X at position l implicates the summation of $L^l_{X,i}$ for all the possible nucleotides (Equation 15.2). The evaluation of the likelihood equation at the root node returns the score L_{Root} for the whole tree. The likelihood value of a complete genomic sequence corresponds to the composition of the likelihood values of every position. Due to the small likelihood values, logarithms over L_{Root} are regularly used.

$$L^l_{Root} = \Sigma_{i=\{A,C,G,T\}} \pi_i L^l_{Root,i} \tag{15.4}$$

$$L_{Root} = \Pi^{l=n}_{l=1} L^l_{Root,i} \tag{15.5}$$

Nevertheless, the search for the tree with maximum likelihood requires the evaluation of a considerable number of equations over a great tree space[3], which converts the searching process using scores in an NP-hard problem [71]. Despite the introduction of $L^l_{X,i}$, the method requires the computation of 4 partial results for each node. So, heuristics must be introduced in order to gain feasible solutions. Instead of exploring all the tree space, bootstrapping [91] looks for a local maximum by injecting small perturbations to a phylogenetic tree in order to examine the direct surroundings. Another trivial simplification consists of the

[3]Total number of rooted trees for $n \geq 2$ leaves: $(2n-3)!! = \frac{(2n-3)!}{2^{n-2}(n-2)!}$

propagation of the local maximum $L^l_{X,i}$ instead of the four internal values $L^l_{X,i}$ during the calculation of MLE. Additionally, some of the current inference tools for MLE estimate an initial phylogeny using a maximum parsimony method [155] and, later, they may assign a preliminary nucleotide to the internal ancestors of the phylogenetic tree. The main drawback of these approaches is the extraction of a local maximum instead of the real solution. The selection of the initial seed and the intensive sampling method of the tree space determines the quality of the approximation to the real MLE value for the phylogeny [164].

Finally, the user must select explicitly the DNA mutation model to carry out the computation of the MLE. Normally, the DNA mutation models only specify a template for the probability transitions among nucleotides and the particular valuation of these parameters must be provided. In any case, the parametric model checking approach presented in Chapter 18 allows the symbolic evaluation of parameters in temporal logic formulas or models, and even infer the numerical values that satisfy the resulting relations. In Section 15.6 we show how a model checking tool can test and compute the fidelity of a given tree according to the defined substitution model.

15.4. Continuous Time Probabilistic Logic and Structure

The existence of adapted tools, data structures and probabilistic temporal logics that capture the underlying randomness of the systems allows for inspecting stochastic properties. Thus, our framework based on formal methods techniques will also accept the characterization of DNA substitution models as input. The analysis of these models opens a new perspective that extends the original applications presented in Chapter 4 to other phylogenetic fields.

Probabilistic model checking combines the expressivity for representing paths of the tree with temporal logics and the computation of likelihoods associated to the nucleotide changes. The mathematical equations of MLE for evaluating the phylogeny are rewritten using probabilistic temporal logics. Later, they are executed over a continuous time Markov chain corresponding to the DNA mutation model. Finally, the model checking procedure queries whether the probability of appearance of a particular mutation (or a set of) in the tree is over or not a predefined threshold.

Occasionally, it is also possible to let the model checker tool discover this probability bound. In that case, the output returns a confidence value between $[0, 1]$, which represents the probability of getting that peculiar arrangement of nucleotides in the states of the tree for that model of DNA substitution. The score provided

by the result of the verification process is useful for guiding the refinement of the phylogenetic tree. For example, our framework can play the role of *worker* for computing the maximum likelihood or evaluating the tree topology in current inference algorithms and tools [154, Figure 2]. The script presented in Section 5.3 accepts extensions for generating multiple bootstraps that feed the iterative refinement process.

The definition of DNA substitution models requires the extension of Markov chains with continuous time. Instead of describing explicitly the probability of a nucleic change from one state to another, the transitions are labeled with rates delimiting the time spent in that branch.

Definition 16 (Continuous time Markov Chain)**.** *A continuous time Markov chain is a finite transition system represented by a tuple* $M = (S, S_0, \mathbf{R}, L)$*, where:*

- S *is a finite set of states,*
- $S_0 \subseteq S$ *is the set of initial states,*
- $\mathbf{R} \subseteq S \times S \to \mathbb{R}_{\geq 0}$ *is the transition rate matrix between states, i.e., for every pair of states* $s, s' \in S$*, a transition occurs only if* $\mathbf{R}(s, s') > 0$*, and the probability of this transition being triggered in* t *time units equals* $1-e^{-\mathbf{R}(s,s')\cdot t}$*, and*
- $L : S \to 2^{AP}$ *is the labeling function that associates each state with the subset of atomic propositions that are true of it.*

As there exist multiple pair candidates with $\mathbf{R}(s, s') > 0$ outgoing from state s, a race condition appears and the selection of the next state determines the history of events. The transition rate matrix $\mathbf{R}$ implicitly defines a transition probability matrix $\mathbf{P}$ that corresponds to the transition probability matrix of an embedded discrete time Markov chain ($emb(M)$), and whose values are:

$$\mathbf{P}_{emb(M)}(s, s') = \begin{cases} \frac{\mathbf{R}(s,s')}{E(s)} & \text{if } E(s) \neq 0 \\ 1 & \text{if } E(s) = 0 \text{ and } s = s' \\ 0 & \text{otherwise} \end{cases}$$

$E(s)$ is known as the *exit rate* of state s. It is defined as the summation of every output transition rate $\mathbf{R}(s, s')$. The value $E(s) = 0$ means that s is an *absorbing state* or *siphon*:

$$E(s) = \Sigma_{s' \in S} \mathbf{R}(s, s')$$

The computation of probabilities is influenced by the paths starting in the initial state. Following the same notation presented for timed transition automata,

$\pi(t)$ returns the state s_i of the path in which the system is found after $t \in \mathbb{R}_{\geq 0}$ time units since the initial state s_0, i.e., $\pi(0) = s_0$. For any set of infinite paths Π starting in the initial state s_0, the subset $\Pi(\pi_n)$ selects the paths $\pi \in \Pi$ whose prefix equals to the finite sequence $\pi_n = s_0 s_1 s_2 \dots s_n$ of length $n+1$ states. Specially, the set $\Pi(s_0, I_0, \dots, I_{n-1}, s_n)$ is defined as all the paths from $\Pi(\pi_n)$ such that $\pi_n(t) = s_i$ for all $i < n$ with $t \in I_i$, and I_i the time interval corresponding to that state.

The likelihood value of reaching a state s_n is determined by the probability measure $Pr(\Pi(\pi_n)) = Pr(\Pi(s_0, I_0, \dots, I_{n-1}, s_n))$ over all the paths sharing the prefix π_n with $\pi_n(0) = s_0$. For the trivial case over $\Pi(s_0)$, $Pr(\Pi(s_0)) = 1$. In general for $\Pi(s_0, I_0, \dots, I_{n-1}, s_n, I', s')$, $Pr(\Pi(s_0, I_0, \dots, I_{n-1}, s_n, I', s'))$ is equal to:

$$Pr(\Pi(s_0, I_0, \dots, I_{n-1}, s_n)) \cdot \mathbf{P}_{emb(M)}(s_n, s') \cdot \left(e^{-E(s_n) \cdot \inf I'} - e^{-E(s_n) \cdot \sup I'} \right)$$

Starting in s, the probability of arriving to a state s' of the model at a particular time instant t is represented by $\mathbf{P}_t(s, s') = Pr\{\pi \in \Pi \mid \pi(t) = s'\}$, with $\mathbf{P}_t$ an exponential matrix indicating the transient probabilities for time t:

$$\mathbf{P}_t = e^{\mathbf{Q} \cdot t} = \Sigma_{i=0}^{\infty} \frac{(\mathbf{Q} \cdot t)^i}{i!}$$

and $\mathbf{Q}$ the infinitesimal generator matrix of M. The infinitesimal generator matrix $\mathbf{Q}$ is used for the calculation of transient states and it is defined as:

$$\mathbf{Q}(s, s') = \begin{cases} \mathbf{R}(s, s') & \text{if } s \neq s' \\ -\Sigma_{s'' \neq s} \mathbf{R}(s, s'') & \text{otherwise} \end{cases}$$

Unfortunately, this method for computing $\mathbf{P}_t$ tends to be unstable. The Jensen's method (also called uniformisation or randomisation) is an alternative and more stable standard technique for calculating the probability matrix $\mathbf{P}_t(s, s')$ [157].

Once the data structure and probabilistic semantic are defined, the next step involves the presentation of the syntax for the new temporal logic. The logic proposed here for working with continuous time Markov chains is CSL^{TA} [59]. In a non-probabilistic temporal logic, it is already possible to query about the arrangement of the DNA information in the states of the tree (see Section 4.3 and Section 4.4). However, what we look for here is the addition of probability information to the specifications. To this end, we will be able to investigate the probability of reaching a certain state of the tree owning a particular mutation.

Definition 17 (Continuous Stochastic Logic)**.** *A temporal logic formula ϕ is defined by the following minimal grammar, where $p \in AP$:*

$$\begin{aligned} \phi &::= true \mid p \mid \neg\phi \mid \phi \vee \phi \mid \mathbb{P}_{\sim\lambda}[\Phi] \\ \Phi &::= \mathbf{X}\phi \mid [\phi \mathbf{U}_I \phi] \end{aligned} \tag{15.6}$$

The formulas are checked against a structure M considering all paths π from a certain state s_0. Notice that $M, s_0 \vDash \phi$ means that s_0 satisfies ϕ. The semantics of well-formed formulas is as follows (let $\pi = s_0 s_1 s_2 \ldots$):

- $M, s_0 \vDash p \Leftrightarrow p \in L(s_0)$,
- $M, s_0 \vDash \neg\phi \Leftrightarrow M, s_0 \nvDash \phi$,
- $M, s_0 \vDash \phi \vee \psi \Leftrightarrow M, s_0 \vDash \phi$ *or* $M, s_0 \vDash \psi$,
- $M, s_0 \vDash \mathbb{P}_{\sim\lambda}[\Phi] \Leftrightarrow Prob(M, s_0, \Phi) \sim \lambda$,

The calculation of the probability $Prob(M, s_0, \Phi)$ requires the identification of the infinite paths π satisfying the path formula $M, \pi \vDash \Phi$:

- $M, s_0 \vDash \mathbf{X}\phi \Leftrightarrow M, s_1 \vDash \phi$
- $M, s_0 \vDash [\phi \mathbf{U}_I \psi] \Leftrightarrow$ *for some* $t \in I$, $\exists \pi : M, \pi(t) \vDash \psi$, *and* $M, \pi(t') \vDash \phi$ *for all* $0 \leq t' < t$.

This set, $\{\pi \in \Pi \mid M, \pi \vDash \Phi\}$, can be obtained by the union of finitely many pairwise disjoint subsets $\Pi(\pi_n)$ by [106, Definition 3], each one characterized by the finite prefix π_n of all infinite sequences of the set. Therefore, $Prob(M, s_0, \Phi) = Pr\{\pi \in \Pi \,|\, M, \pi \vDash \Phi\} = \Sigma_{\pi_n} Pr(\Pi(\pi_n))$ computes the probability as the summation of probabilities in all possible prefixes π_n by [106, Theorem. 1].

Similarly to TCTL and PCTL, this logic supports timed transitions in the **U** operator. Timed variants of the modal operators **F** and **G** are obtained via **U** as $\mathbf{F}_I \phi = true\, \mathbf{U}_I \phi$ and $\mathbf{G}_I \phi = \neg \mathbf{F}_I \neg\phi$. Instead of writing probability and time intervals explicitly, sometimes they are abbreviated with inequalities. For example, $\mathbb{P}_{\leq 0.5}[\Phi]$ denotes $\mathbb{P}_{[0,0.5]}[\Phi]$.

By now, we can translate the mutation traces along a phylogenetic tree into the CSL$^{\text{TA}}$ syntax. As a toy example, suppose that the initial state s_0 (the root of the phylogenetic tree) has a sequence with $seq[i] = A$ and there is a direct successor s_1 having a mutation $seq[i] = T$, with i the position of the sequence. The equation $\mathbb{P}_{\geq 0.25}[\mathbf{X}\, seq[i] = T]$ tells whether, according to an specific DNA substitution model, the change of A by T is over the probability threshold of 25%. The initial state s_0 is embedded for the operator $\mathbb{P}$. For the operator **X**, the time elapsed between both nucleotides is implicit and it takes the value of the branch length.

Following this simple step, the computation of the overall probability of the phylogenetic tree can be obtained by the concatenation of $\mathbb{P}_{\sim\lambda}$ for each mutation happening along the transitions of a path. The evaluation of the CSL$^{\text{TA}}$ formulas needs the algorithm introduced in the next section. The new model checking algorithms deal with the computation of time and probabilities.

15.5. Algorithm for CSL$^{\text{TA}}$ Model Checking

In the previous section, we have defined CSL$^{\text{TA}}$. With that logic, we can express the succession of mutations appearing in the phylogenetic tree, that is, the specification of a trace for a DNA substitution model. The verification process now consists of returning a numerical value telling the degree of agreement of the phylogeny with respect to that mutation model.

The model checking algorithm for managing and solving CSL$^{\text{TA}}$ formulas in stochastic systems is mainly identical to those of classic model checking except for the resolution of $\mathbb{P}_{\sim\lambda}[\phi]$, i.e., the next and until operators with probability thresholds. In short, the recursive algorithm of model checking incorporates the new sentence:

$$Sat(\mathbb{P}_{\sim\lambda}[\Phi]) = \{s \in S \mid Prob(M, s_0, \Phi) \sim \lambda\}$$

$\mathbb{P}_{\sim\lambda}[\mathbf{X}\phi]$ formula. In CSL$^{\text{TA}}$, the next operator has no sense as in continuous time there is not an unique next real number. This operator is included for compatibility. It corresponds to the semantics and operations of the associated PCTL formula for the embedded discrete time Markov chain (Section 14.3).

$\mathbb{P}_{\sim\lambda}[\psi\mathbf{U}_I\phi]$ formula. The computation of the probability for the until operator depends on the value of the interval I. Generally, when $I = [0, \infty]$, then $Prob(M, s_0, \psi\mathbf{U}_{[0,\infty]}\phi) = Prob(emb(M), s_0, \psi\mathbf{U}_{\leq\infty}\phi)$. For the rest of cases, the interval I is classified as:

- $I = [0, t]$ with $t \in \mathbb{R}_{\geq 0}$;
- $I = [t, t']$ with $t, t' \in \mathbb{R}_{\geq 0}$ and $t \leq t'$;
- $I = [t, \infty]$ with $t \in \mathbb{R}_{\geq 0}$.

For the case of $I = [0, t]$, then $Prob(M, s, \psi\mathbf{U}_{[0,t]}\phi)$ is equal to:

$$\begin{cases} 1 & \text{if } s \in Sat(\psi) \\ 0 & \text{if } k = 0 \text{ or} \\ & s \in Sat(\neg\phi \wedge \neg\psi) \\ \int_0^t \Sigma_{s' \in S} \mathbf{P}_{emb(M)}(s, s') \cdot E(s) \cdot e^{-E(s)\cdot x} \cdot Prob(M, s', \psi & \mathbf{U}_{[0,t-x]}\phi)dx \\ & \text{otherwise} \end{cases}$$

with $E(s) \cdot e^{-E(s)\cdot x}$ the probability of leaving s at time x. Following a simpler alternative approach, the probability can be reformulated in terms of transient probabilities:

$$Prob(M, s, \psi\mathbf{U}_{[0,t]}\phi) = \Sigma_{\substack{s \in Sat(\neg\psi \vee \phi) \\ s' \in Sat(\phi)}} \mathbf{P}_t(s, s')$$

When $I = [t, t']$, the until operator must consider a) the time t spent in states satisfying ψ plus b) up to time $t' - t$ required for reaching ϕ. The first part can be compared to $\mathbf{F}_{[0,t]}\psi$ and the second part corresponds to $\psi \mathbf{U}_{[0,t'-t]}\phi$:

$$\begin{cases} 1 & \text{if } s \in Sat(\psi) \\ 0 & \text{if } k = 0 \text{ or} \\ & s \in Sat(\neg\phi \wedge \neg\psi) \\ \Sigma_{\substack{s \in Sat(\neg\psi) \\ s' \in Sat(\psi)}} \mathbf{P}_t(s, s') \cdot Prob(M, s', \psi \mathbf{U}_{[0,t'-t]}\phi) & \text{otherwise} \end{cases}$$

The last case $I = [t, \infty]$ is similar to the previous $I = [t, t']$ but changing $\psi \mathbf{U}_{[0,t'-t]}\phi$ by $\psi \mathbf{U}\phi$ due to the infinity. As the until operator is unbounded, it can be evaluated in the embedded discrete time Markov chain:

$$\begin{cases} 1 & \text{if } s \in Sat(\psi) \\ 0 & \text{if } k = 0 \text{ or} \\ & s \in Sat(\neg\phi \wedge \neg\psi) \\ \Sigma_{\substack{s \in Sat(\neg\psi) \\ s' \in Sat(\psi)}} \mathbf{P}_t(s, s') \cdot Prob(emb(M), s', \psi \mathbf{U}\phi) & \text{otherwise} \end{cases}$$

The time complexity of verifying a CSL$^{\text{TA}}$ formula ϕ against a continuous time Markov chain is linear in $|\phi|$ and polynomial in the size of S, with $|\phi|$ the number of logical connectives and temporal operators of the formula. More generally, the complexity is

$$\Theta(poly(size(S)) * q \cdot t_{max} * |\phi|)$$

where t_{max} is the maximal bound of a path subformula $\psi_1 \mathbf{U}_I \psi_2$ of ϕ, with $t_{max} = 1$ if it doesn't contain any $\mathbf{U}$ subformula. The parameter q is equal to $q = max_{s \in S}|\mathbf{Q}(s, s)|$.

The evaluation of CSL$^{\text{TA}}$ formulas demands the upgrade of Markov chains structures for providing continuous time semantics. The computation of probabilities is assisted by a partial transformation of the continuous time problem to a discrete time problem. Hence, the operations realized by the model checking algorithms for the calculation of probabilities in PCTL are the basis for future extensions. The time complexity of verifying a PCTL formula also poses a lower bound to the complexity of analyzing a CSL$^{\text{TA}}$ formula.

15.6. Model Checking Tools and Experimentation

PRISM [124] is a generic model checking tool capable of handling probabilistic and timed specifications over Markov chains. There exists a considerable diversity of model checking tools with different performances and qualities [98]. Although the real performance depends on the particular structure of the model and specifications, PRISM offers Java portability, a powerful syntax for handling time and probabilities in models and specifications, and a good scientific community support. Besides, it is open source, which allows the modification and optimization of its code.

The model checking tool requires two input files for the verification process: a first file with the description of the model, and a second file with the specification of the properties. Thus, the DNA mutation model and the MLE equations are stored in separated files. The modularity and independence of the model and specification allow the evaluation of various trees over the same model of DNA substitution. Vice versa, the study of the same tree under the scope of various DNA mutation models is possible by simply maintaining the MLE specification and changing the model file with the DNA substitution pattern.

Figure 15.3 corresponds to the first file: the model of DNA substitution over which the phylogenetic tree will be evaluated. By changing the relations in the ratio of transversions and transitions, the user switches from one mutation model to another. The computation of the probability is model-dependent, and thus the user has to select a model *a priori*. Some of the historic mutation models are Jukes-Cator (JC), Kimura, Felsenstein or the Generalized time-reversible (GTR) [53]. For simplicity in this example, we consider the JC model. The substitution rate *nu* equals to 1. It expresses the number of ticks needed for the activation of the transition rather than an explicit probability.

The translation of the MLE equations to the syntax of the stochastic logic supported by PRISM is exemplified in Figure 15.4. It corresponds to the MLE equations defining the tree of Figure 2.1. The conversion process is dealt by a BioPerl script [153]. The unfolding of the MLE equations depends on the structure of the phylogenetic tree and the DNA sequences placed in its tips. In an indirect way, these MLE equations represent the *specification* of the phylogenetic tree over a DNA substitution model because they depict the trace of mutations from the root to the leaves.

Among its basic conceptions, PRISM checks if the probability of reaching a set of satisfiable states is up or below a predefined threshold. Besides indicating specifications with an explicit threshold, it is also prepared for determining the MAX and MIN probabilities of satisfying a property in a set of states. Instead of

```
// Continuous time Markov chain
ctmc

const double e = 2.71828;
const double nu = 1;

// Nucleotids //
const int A = 1;
const int C = 2;
const int G = 3;
const int T = 4;

//JC model
formula pr_ii = nu;
formula pr_ij = nu;

module TREE
x : [A..T] init A;

[] x=A -> pr_ii:(x'=A)+pr_ij:(x'=C)+pr_ij:(x'=G)+pr_ij:(x'=T);
[] x=C -> pr_ij:(x'=A)+pr_ii:(x'=C)+pr_ij:(x'=G)+pr_ij:(x'=T);
[] x=G -> pr_ij:(x'=A)+pr_ij:(x'=C)+pr_ii:(x'=G)+pr_ij:(x'=T);
[] x=T -> pr_ij:(x'=A)+pr_ij:(x'=C)+pr_ij:(x'=G)+pr_ii:(x'=T);
endmodule
```

Figure 15.3: Description of the Jukes-Cantor model in PRISM syntax.

asking for a particular bound, PRISM includes iterative methods for numerically calculate probabilities along a path of the tree, which allows us to extend the output of quantitative model checking from time (Chapter 12) to probabilities.

The operator `P=?[F[0,5] x1 = A]` of the model checking tool returns the probability of reaching a nucleotide A in the position `x1` of the alignment within 5 time steps in the future. It is equivalent to $\mathbb{P}_{=?}[\mathbf{F}_{[0,5]} seq[1] = A]$ in CSL^{TA} syntax, but with ? the probability threshold calculated by the model checking tool instead of a value introduced by the user. Filters in PRISM allow to combine the aforementioned operator with the specification of the nucleotide value in the initial state. For example, `filter(min, P=? [F[0,5] x1 = A], x1 = C)` returns the minimum probability of `P=?` considering that the initial state has the sequence `x1 = C`. If we desire to ensure explicitly that `x1 = C` holds **U**ntil we reach `x1 = A`,

```
// Markow decision process
// Notation: L_<id_state>_<seq_position>_<nucleotide>
// p_<nucleotide>
// Root likelihood
"LX": "LX1" * "LX2" * "LX3";
"LX1": pA*"LX1A" + pC*"LX1C" + pG*"LX1G" + pT*"LX1T";
...
// Likelihood for the intermediate node Y
"LY1": "LY1A" + "LY1C" + "LY1G" + "LY1T";
"LY1A": (filter(min, P=? [F[1.5,1.5] x = A], x = A) * "LR1A"
+ ... + filter(min, P=? [F[1.5,1.5] x = T], x = A) * "LR1T")*
(filter(min, P=? [F[1.5,1.5] x = A], x = A) * "LS1A"
+ ... + filter(min, P=? [F[1.5,1.5] x = T], x = A) * "LS1T");
...
"LY1T": (filter(min, P=? [F[1.5,1.5] x = A], x = T) * "LR1A"
+ ... + filter(min, P=? [F[1.5,1.5] x = T], x = T) * "LR1T")*
(filter(min, P=? [F[1.5,1.5] x = A], x = T) * "LS1A"
+ ... + filter(min, P=? [F[1.5,1.5] x = T], x = T) * "LS1T");

// Terminal leaves whose nucleotide is present in the tree.
"LR1A": 1;
"LS1A": 1;
"LZ1G": 1;
...
// Terminal leaves whose nucleotide is not present in the tree.
"LR1C": 0;
"LS1C": 0;
"LZ1A": 0;
...
```

Figure 15.4: Representation in PRISM syntax of the MLE equations for Figure 2.1.

then we can use `filter(min, P=? [(x1 = C) U<=5 (x1 = A)], x1 = C)`. The properties can be annotated with names and we use this feature for defining the partial likelihoods $L^l_{X,i}$. In case of phylogenetic trees with an estimated nucleotide for the ancestors (such as in the case of maximum parsimony), the intermediate nodes can be initialized with the value $L^l_{X,i} = 1$ or $L^l_{X,i} = 0$ for reducing the firsts calculations. The header of the file describes the notation of the variable names.

The execution of Figure 15.4 directly in PRISM leads to an extensive repetition

of the calculations. In spite of the annotation of the properties with names, the model checking tool does not store the partial results $L^l_{X,i}$ in local memory, which implies a reevaluation of $L^l_{X,i}$ every time it is accessed. The inability for caching these values damages the potential optimization caused by the factorization of the probabilities $P_{ij}(\cdot)$ in Equations 15.2–15.4 and slows down the system. In a perfect binary tree, every node requires the inspection of 8 values $L^l_{X,i}$ (4 per nucleotide for each descendant). This fact implicates the access to 8^M partial likelihoods for the whole tree, with $M = log_2(2N)$ the maximum depth. In terms of the number of internal nodes N, PRISM evaluates the likelihoods $8N^3$ times, which is greater than the storage of $4(2N)$ partial likelihoods needed for the complete phylogeny. This peculiarity, together with the inherent delay introduced by the Java virtual machine, penalizes the performance in comparison to the specific systems for computing MLE that are capable of analyzing several trees in seconds within clusters [155].

Figure 15.5 presents the performance results of the experiments. The data set used for this experimentation is synthetic. With this data set, we try to cover the spectrum of small phylogenies and analyze the cost of the evaluation of the MLE equations over there. We have created random phylogenetic trees of up to 20 tips using a Yule backward model [156]. For each tree size (number of tips), we have generated ten random trees and calculated the harmonic mean time. The DNA sequences have a single base with an homogeneous distribution of nucleotides. Although subject to a perfect parallelization, the temporal cost for larger strings can be estimated linearly by multiplying the time by the number of bases. All tests have been run on a Intel Core 2 Duo E6750 @ 2.66 GHz with 8 GB RAM and Linux. The factorization of the MLE equations, its adaptation to the context of the new tool, and the cost of initialization in PRISM are questions that must be considered for the judgment of the impact.

We have modified PRISM for caching the partial results $L^l_{X,i}$ in order to avoid its recomputation. Besides, the likelihood of the terminal nodes is now stored in constants (i.e., `const LR1A: 1;`). Using these simple changes, the model checker increases the speed effectively but doesn't fade out the exponential cost yet. Figure 15.6 presents the performance results of the experiments with the new revision of PRISM. We have used the same data set and workstation for simplifying the comparison. The exponential trend increases more slowly. Memory doesn't pose a limiting factor because the experiments have been executed with only 4GB assigned to the java virtual machine.

The exponential trend observed in the graphics is probably motivated by the exploration method selected by PRISM for calculating the maximum value of the probability function $P_{ij}(d_{ij})$, i.e., `filter(max, P=?  [F=`d_{ij} `(x1 = j)], x1 = i)`. In this way, PRISM unfolds the DNA mutation model (Figure 15.1) and

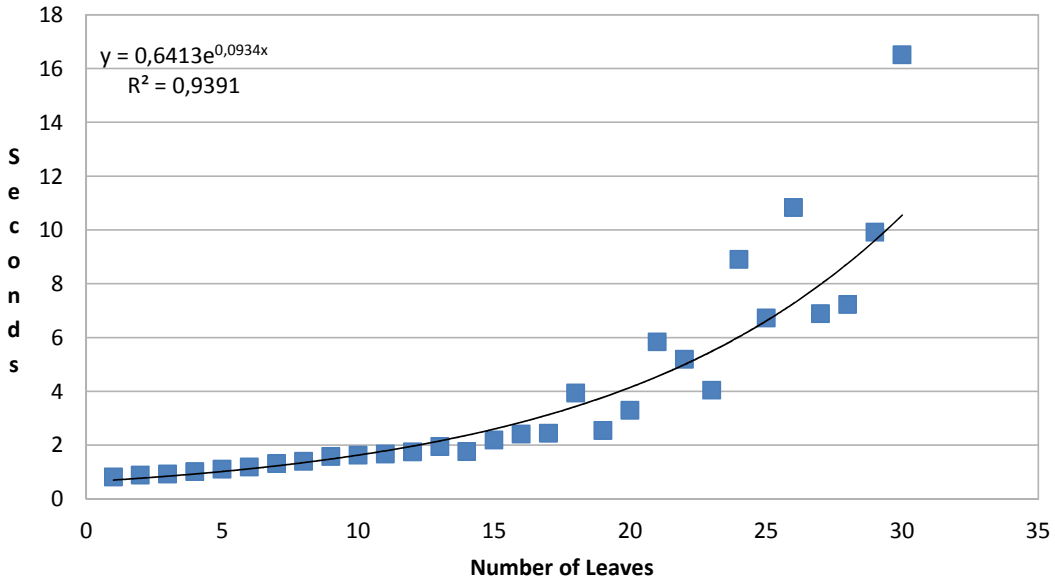

Figure 15.5: Time required in PRISM for the evaluation of the maximum likelihood equations in a binary phylogenetic tree.

generates all possible traces between the nucleotides i and j (Figure 15.2). Then, PRISM searches for the path with distance $\Sigma_{k=1\ldots n} d_k = d_{ij}$ that maximizes $P_{ij}(d_{ij})$, being i, j the initial and final states, and d_k the distance between each pair of connected states in a path with n nodes. The combinatorial examination of every path satisfying the previous restrictions of length, together with the computation of the path probabilities, leads to the exponential cost for calculating $P_{ij}(d_{ij})$.

In order to take advantage of the peculiar procedure in PRISM for managing probabilities, we propose an alternative reordering of the MLE equations. Figure 15.7 shows this rewriting. In this example, exclusively the phylogenetic tips are attached with an explicit definition of the nucleotides: the bases in the leaves are the only values that we know with certitude. Hence, the internal nodes are left undetermined in the specification. In fact, the MLE equations (Equation 15.2) test the four nucleotide values, while here we let PRISM to automatically discover the best choice. The constants `dxy` are the distance between states X and Y, as explained in Section 15.3. The summation of the distances `dxy+dyr` works for marking the length of the path between the root and the leaf R. The estimation resulting from the evaluation of the formulas in Figure 15.7 places an upper bound to the real value of MLE. The evaluation of each filter extracts independent paths with maximum probability, presumably giving rise to a set of disjoint routes whose only common ancestor is the root. In any case, this set of paths, although depict-

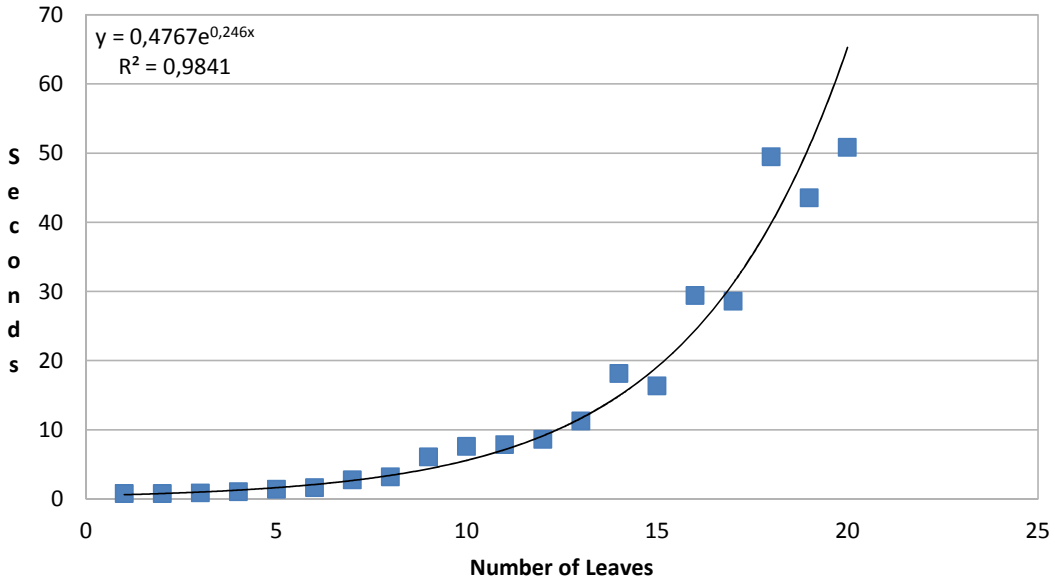

Figure 15.6: Time required in PRISMopt for the evaluation of the maximum likelihood equations in a binary phylogenetic tree.

ing a degenerated phylogenetic tree, results in a suitable heuristic for pointing out the maximum value of MLE.

Figure 15.8 presents the performance results of the experiments. We continue using the same methodology introduced in previous trials. We have created random phylogenetic trees of up to 1000 tips using a Yule backward model [156]. For each tree size (number of tips), we have generated ten random trees and calculated the harmonic mean time. The DNA sequences have a single base with an homogeneous distribution of nucleotides. Both versions of PRISM (with and without saving the partial results of named properties) are supposed to perform similarly. The first consequence of the reordering in the MLE equations is a linear trend in the temporal cost, plus the capacity of evaluating bigger phylogenies and larger sequences than before.

Although the tool is not extremely fast for the evaluation of the original MLE and the time results may be disappointing, the ultimate objective of this thesis has not been the score of phylogenetic trees using a new technique for computing the MLE. Our aim has been the definition of a framework for evaluating phylogenetic properties, in which the MLE is only a small portion of all the potential applications. The main advantages of our approach are still valid for this new situation. We emphasize that our main contributions comprise the generality for expressing any kind of property in a temporal logic, the independence of the model from the

```
// Markow decision process
// Notation: L_<id_state>_<seq_position>_<nucleotide>
// p_<nucleotide>
// Root likelihood
"LX": "LX1" * "LX2" * "LX3";
"LX1": pA*"LX1A" + pC*"LX1C" + pG*"LX1G" + pT*"LX1T";

// Likelihood of reaching a certain nucleotide in the tips
// starting with a particular nucleotide in the root.
"LX1A": filter(min, P=? [F=(dxy+dyr) x = A], x = A) *
filter(min, P=? [F=(dxy+dys) x = A], x = A) *
filter(min, P=? [F=dxz x = G], x = A);

"LX1C": filter(min, P=? [F=(dxy+dyr) x = A], x = C) *
filter(min, P=? [F=(dxy+dys) x = A], x = C) *
filter(min, P=? [F=dxz x = G], x = C);

"LX1G": filter(min, P=? [F=(dxy+dyr) x = A], x = G) *
filter(min, P=? [F=(dxy+dys) x = A], x = G) *
filter(min, P=? [F=dxz x = G], x = G);

"LX1T": filter(min, P=? [F=(dxy+dyr) x = A], x = T) *
filter(min, P=? [F=(dxy+dys) x = A], x = T) *
filter(min, P=? [F=dxz x = G], x = T);
```

Figure 15.7: Rewriting of the MLE equations for Figure 2.1.

specifications, the availability of powerful model checking tools adapted for different logics, and the encapsulation of the implementation in a model checking tool that hides and simplifies the access to the underlying technology. The application of model checking in the context of phylogenetics includes many functionalities such as the detection of back mutations, conserved sequences and correlated mutations, SNP's defining clades, population migrations, endemic diseases or phenotypes and so on. All these properties can be enriched with time and probabilities as well.

Future optimizations should focus on the implementation of more efficient libraries and dynamic programming techniques that solve, or at least ease, the aforementioned limitations and bottlenecks in PRISM. After enabling the capability of storing partial results, the next dilemma revolves around the portability of languages such as Java or the performance of high optimized codes in C.

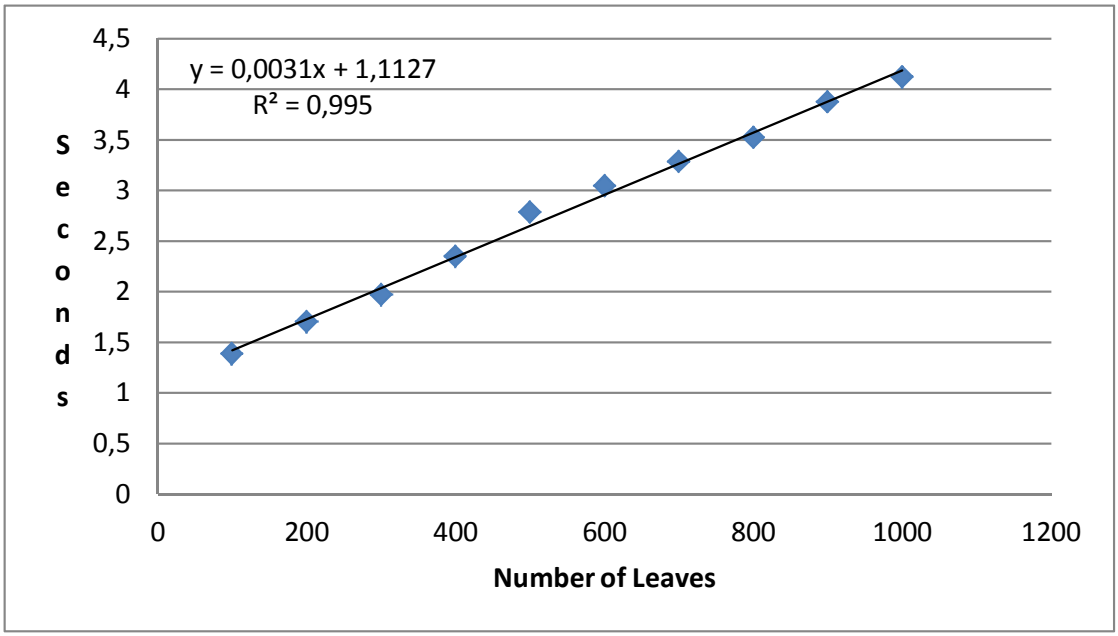

Figure 15.8: Time required in PRISM for the computation of the upper bound of the maximum likelihood value in a binary phylogenetic tree.

Finally, the support for constants in the specifications of PRISM, which can be later instantiated by numerical values during the experimentation, intuitively brings the notion of parameters. The symbolic manipulation of variables extracts algebraic expressions, which can be later evaluated over a domain of values for model exploration. The definition of models and specifications with parameters allows for an intuitive use of templates and patterns as introduced in Chapter 18.

15.7. Conclusions

In this chapter, we have motivated the extension of model checking to quantitative domains, in particular with continuous time. Our objective has been the extension of probabilities presented in Chapter 14 to continuous domains. Using continuous time Markov chains and continuous time logics, we can define and evaluate the same properties of Chapter 14 but with the dense time introduced in Chapter 11.

In particular, we have studied the models of DNA evolution and evaluated the equations of maximum likelihood estimations (MLE) over trees, one of the hot topics in phylogenetics. In this case, the verification process changes with respect to the methodology presented in previous chapters of this thesis. Now, the model

(described by the Markov chain) is the model of DNA evolution; and the specifications that we try to validate correspond to the equations of maximum likelihood. The equations of maximum likelihood depends on the structure and labeling of a phylogeny, and consequently, the evaluation of these formulas is equivalent to analyze the match between a hypothetical computation tree generated by the model of DNA evolution and the input tree that we investigate. The output result is a score representing the reliability and support of the input tree. This problem is similar to the computation of distances between symbolic objects (states) presented in Chapter 12, where the only difference is the domain of the numeric values of the functions we are working with: reals (probabilities) instead of integers (temporal distances).

After the presentation of the models of DNA evolution and the method for calculating maximum likelihood estimations over trees, we have introduced the temporal logic and algorithms for managing probabilities and continuous time in the specifications. Next, we have shown how to translate and evaluate the model and specifications with this particular notation. We have experimented with synthetic data in order to prove the feasibility of our approach with existing probabilistic model checking tools. PRISM is a generic model checking tool that performs exponentially with respect to the number of tips of the phylogeny. We have customized the tool in order to store the partial computations of the likelihoods and improve slightly the initial costs (by default, PRISM doesn't store the local results). Finally, we have illustrated an heuristic for the calculation of an upper bound of the likelihood score in linear time with respect to the number of leaves.

This work opens the door for the review of bigger phylogenies with properties similar to the computation of maximum likelihoods. The modularity of our framework allows the evaluation of hypothesis and the comparison of results for a set of phylogenetic trees by only changing the tree file (the specification of the property remains constant).

Chapter 16

Conclusions

I've seen things you people wouldn't believe. Attack ships on fire off the shoulder of Orion. I watched c-beams glitter in the dark near the Tannhäuser Gate. All those moments will be lost in time, like tears in rain. Time to die.

— Roy Batty, *Blade Runner*

In this part we have motivated the necessity of extending model checking techniques to probabilistic domains using real examples: we travel from explicit time logics in Chapter 11 to discrete and continuous time logics in Chapter 14–15. For instance, we have studied the ratio distribution of lactose (in)tolerance in a phylogenetic tree or the scoring of a phylogeny with respect to a DNA substitution model using maximum likelihood estimations. To this end, we have introduced extended logics and data structures adapted for probabilities and time together with the algorithms and computations for managing them. Our first goal has been the increase of the logical capabilities for querying about the date of appearance and degree of distribution of mutations and phenotypes.

The performance of the experimentations has been dissimilar. We have used the model checking tool PRISM and the same data set for both cases (the example of lactose intolerance and the score of likelihoods). Our data set comprises synthetic phylogenetic trees and sequences. While the evaluation of the lactose property using probabilities over discrete time Markov chains (phylogenies) scales polyniomally, the calculation of likelihoods over continuous time Markov chains (models of DNA substitution) scales exponentially. For the second situation, we needed the customization of the model checking tool and the inclusion of heuristics in order to obtain competitive results. More in detail, we introduced local variables for the storage of partial results to avoid the recalculation of values; and the extraction of upper bounds of likelihood scores as heuristics.

Additionally, the techniques introduced for the optimization of the performance in classic model checking environments also work for this context [140]. Most of the investigations related with quantitative information are a prolongation of the phylogenetic properties analyzed with boolean temporal logics, and consequently the inconveniences presented there would appear even more dramatically now [143] (for instance, the consideration of the decidability and computational costs). The division of the phylogeny in subtrees and the coordinated evaluation of properties in them potentially increases the overall speed. The use of the transition/rate probability matrix in stochastic systems simplifies the division of the phylogeny in subtrees or clusters of nodes. The partition in submatrices facilitates the distribution of the state space among several computers. The summations in the computation of probability paths, specially with the **U** operator, can be executed in parallel.

Furthermore, the incorporation of external data bases for storing the labeling of the states is compatible with any kind of atomic propositions, quantitative or not. The vertical and horizontal partitioning of the database table adds an extra dimension of parallelism. The slicing of the information associated to each state directly over the phylogeny may augment the number of nodes during the study, but it will help to parallel and scale the verification in model checking. In any case, the difficulty of implementing a completely new software, the existence of few (but good and powerful) model checking tools such as PRISM, and the availability of its source code favors the reuse of current tools instead of concerning with new implementations.

Part VI

Conclusions and Further Remarks

Chapter 17

Conclusions

> *On a long enough timeline, the survival rate for everyone drops to zero.*
>
> — Chuck Palahniuk, *Fight Club*

The aim of this thesis has been to build a bridge between two such apparently removed worlds as phylogenetics and formal verification using model checking techniques. A formal framework for describing, verifying and manipulating causal relationships of states irrespective of the structure (tree or network) has been proposed. This is founded on three pillars:

- The interpretation of a phylogenetic tree as a transition system, where the structure of the tree is transformed in order to assimilate it to the Kripke structure used by model checking techniques.
- The definition of a suitable temporal logic for the expression of phylogenetic properties, the semantics of which is based on Kripke structures derived from phylogenetic trees. The temporal properties inquire about logical or numerical relations in the tree.
- The use of standard model checking techniques and tools for the automated verification of phylogenetic properties expressed as temporal logic formulas in phylogenetic trees interpreted as Kripke structures. Nowadays, model checking is a mature field with well-founded theory supported by generic and extensively used software tools.

We must emphasize that the notion of the phylogenetic tree as an evolution process is captured by the states of a transition system, which reflect a specific step of the speciation process, and the transitions themselves, which describe mutations and reproduction events. Besides, the codification selected for the DNA sequences

as atomic propositions in the states of the Kripke structure allows for the evaluation of phylogenetic properties and the symbolic manipulation of sets using a suitable temporal logic.

The principal advantages stemming from the study of phylogenetic properties with this approach are that different phylogenies can be considered, complex properties can be specified as the logical composition of others, and the refinement of unfulfilled properties (as well as the discovery of new ones) can be undertaken by exploiting the verification results. The formal methods presented here offer an integrated framework for the verification of phylogenetic properties using a symbolic and abstract reasoning. In addition, the introduction of temporal logics adds independence and modularity to the definition of biological specifications with respect to the formalization of the model (phylogeny): the same property can be exported and evaluated in other phylogenies easily. Moreover, the specification with temporal logics is transparent to the underlying technology of a generic model checking tool.

In the proposed strategy, the use of existing tools for model checking has been an initial requirement. This is motivated by the fact that model checking techniques are a consolidated verification technique in these last 20 years, making available today tools that have been proved, debugged and optimized. In this sense, we believe that this is the best way of incorporating quickly the methodology proposed in this thesis to the world of phylogeny.

Next, we have investigated the feasibility of model checking techniques as a framework for hypothesis testing and phylogenetic analysis. We have shown how to translate phylogenetic trees into the syntax of NuSMV and described the performance of the model checker when using real phylogenetic data. We have seen that the initialization phase (creation of the associated Kripke structure) is much costlier than the verification process of a single formula. The experimental results show that the initialization time increases quadratically with the alignment size and linearly with the sequence length. Additionally, memory consumption is sublinear in both cases. Despite the resources needed for model checking, it is still competitive thanks to the symbolic manipulation of huge amounts of data.

The data set used there consists of proteins coded by genes from the mtDNA genome, which are substantially smaller than those from nuclear DNA. As the temporal cost increases mostly with respect to the sequence length, the phylogenetic analysis of large genes and genomes becomes the major bottleneck. Thus, scaling the model checking verification process both in time and memory has been one of our research priorities. We have already suggested some possible solutions for these limitations, such as sliced and distributed model checking or the storage of temporal properties in databases.

Sliced model checking consists of replicating the skeleton of the Kripke struc-

ture among several computers or threads, projecting an slice (portion) of the DNA to each worker, executing the verification in sychronization and finally composing the partial results. The inherent acyclicness of trees also facilitates the implementation, synchronization and composition of the verification results using traditional graph partition techniques (i.e., dividing the tree in subtrees instead of splitting the genome). The use of an external database management system enables the scalability in data storage and the introduction of optimizations during the retrieval of huge amounts of biological information. An hybrid system that combines the advantages of a distributed execution flow plus an external database system offers a promising framework. The viability (implementation and experimental results) of each one of these approaches over a real platform was showed in previous sections. The final speed up depends on the architecture and efficiency of the specific database management system.

Furthermore, phylogenetic model checking is not limited to qualitative temporal logics nor to a specific phylogenetic representation. The existence of a wide range of model checkers and compatible logics (quantitative, statistical or parametric) enlarges the expressiveness and scope of biological properties. It also ensures a considerable collection of tools that can be adapted for any specific study of phylogenetic analysis. In particular, we start our study with the usage of quantitative information in the definitions: more in detail, we employ explicit time (distances) and probabilities. The branches of the phylogeny and the phylogenetic properties are extended with this new information. The phylogenetic hypothesis are organized and classified according to the quantitative data they use.

Moreover, we have demonstrated the application of model checking in a real example for the evaluation of a population hypothesis with explicit time and probabilities. For instance, we have tried to detect the epoch of appearance and distribution impact of certain human diseases or phenotypes in a population tree. The phylogenetic tree requires an enriched labeling of the states for a correct association between phenotypes and genomes. The evaluation of phylogenetic properties expressing time and probabilities scales polynomially with respect to the size of the tree. The data set employed (phylogenies of up to 1000 leaves with 50 nucleotides per taxon) corroborates the viability of our approximation for small-medium size phylogenetic trees.

Additionally, the Kripke structures and almost every type of temporal logics support the generalization of a phylogenetic tree as a phylogenetic network with multiple roots, cycles and horizontal transferences. More powerful data structures such as Markov chains are also accepted by specialized model checking tools, which allows for the incorporation of time and probabilities together inside the same model. The calculation of probabilities depends on the kind of time we are working with: discrete or real time.

Besides embedding explicit time and probabilities in the model and specifications, one of our last objectives is the return of numerical values as result of the model checking process instead of a boolean. To this end, we apply arithmetic functions over the set of states that are identified symbolically by a temporal logic formula. Depending on the type of result, the output of the model checking algorithms is classified as integers (representing distances between elements) or a real numbers (representing probabilities and ratios). In this way, we have illustrated the translation of phylogenetic measures for calculating the balance and asymmetry of trees into a specific logic and notation compatible with the model checking tools. The execution of these specifications returns the temporal distances (or accumulated weight) through a path of the phylogeny connecting sets of states.

In opposition, the inspection of DNA substitution models, defined as Markov chains, enables the computation of probabilistic scores for phylogenies using maximum likelihood estimations (MLE). The phylogenetic tree is passed to the model checker as the specification of the DNA substitution model: the phylogeny reflects the mutation trace along the history of the DNA. The equations of MLE associated to a phylogeny are the specification of the phylogenetic tree that requires as input the model checking tool. The efficiency of evaluating the plain MLE equations is the main pitfall of PRISM due to the technology it uses. To solve this, we have introduced simple modifications in the code together with an heuristic to approximate the computation of the upper bound of the MLE formulas. In consequence, we have obtained a framework that is competitive in performance and, in addition, can easily switch of DNA mutation model or specification (phylogeny) thanks to the independence of the input files.

We can conclude that our approach to the analysis of phylogenies using model checking is innovative and encouraging in terms of methodology and efficiency. This framework and all its adaptations is, at least, a powerful and competitive approach for the qualitative analysis of mtDNA. Here, we integrate different kind of phylogenetic analysis under the same scope: we include the specification of a plethora of qualitative properties (organized according to the type of information they use), their extensions for discrete and continuous time, the addition of probabilities to the path operators, and finally the computation of numerical values as output of a model checking tool (both integers/distances and reals/probabilities). Moreover, we provide a framework that facilitates the symbolic manipulation of the states of the phylogenetic tree by means of temporal logic formulas. The optimization of our framework for the evaluation of probabilistic and parametric properties in larger phylogenies will be the subject of future work.

Chapter 18

Future Work: Parametric Temporal Logic

The supreme art of war is to subdue the enemy without fighting.

— Sun Tzu, *The Art of War*

18.1. Introduction

After the presentation of the conclusions of this thesis, this chapter introduces a preliminary exploration of a parametric framework for future works beyond this dissertation. As final remarks, biologists occasionally do not want to verify a specific hypothesis but let the computer automatically discover new ones. The definition of patterns is a common procedure for specifying properties, which intuitively leads to parametric model checking. This paradigm moves a step forward the previous model checking framework for quantitative properties. The parameters allow the formalization of patterns that simplify the specification of properties and help for the model exploration. Ultimately, we want to mine genome-wide, high-resolution phylogenies. In the examples presented in the previous sections, it can be appreciated that mining for knowledge without prior information requires a more or less thorough exploration of the structure, which can be combinatorial in some or all of its dimensions. The exhaustive inspection of potential solutions involves the checking of large sets of formulas and an intensive use of the topology and the information of each state.

For a phylogenetic property, parameters can be placed either in the DNA sequence (searching for the mutations activating a phenotype), in the time bound (searching for the date of appearance of the biological adaptation) or in the prob-

ability threshold (searching for the distribution of the lactose tolerance in African populations). Even, parameters can be placed in the branches labeled with time and probabilities in the phylogenetic tree. The introduction of parameters is not limited to the specifications, but also they can appear in the model in some cases. The sections of this chapter are organized according to the type of parameters we work with: characters, time or probabilities.

18.2. Parameters in Boolean Model Checking

Some theoretical approaches for CTL [38, 12] bridge the transition between verification and mining. Basically, they define a kind of CTL query with a hole or parameter, called the *placeholder*, that acts as a free variable. If the system verifies the query pattern for any combination of values, the model checking algorithm returns the set of subformulas that satisfy the placeholder. TLQSolver is a model checking tool that support this logical extension [12, 79].

Furthermore, the integration of relational algebras with temporal logics opens a new way for retrieving data from the models. The utilization of current database systems for storing the definition of a transition system model, and the use of database front-ends for processing temporal formulas reinterpreted as SQL queries, offer an intermediate method for obtaining the values satisfying an abstract relation [151, 140].

Many formulas presented in Table 4.1 will benefit of these technologies for the inference of boolean values or DNA characters in the equations. The discovery of columns or nucleotide values for which a sequence is conserved or for which a tree path presents a back mutation figure among the brightest properties.

A trivial approximation for implementing qualitative parametric model checking over finite domains consists of expanding a parametrized formula into multiple independent non-parametric equations where the variables are instantiated with all the possible values of their domain. In the case of finite domains, such as the family of bases of the DNA or the length of an alignment, the range of the variables representing the nucleotides and the columns of a sequence is bounded. This leads into a combinatorial number of equations and a verification process that, although intensive in computation, is completely parallelizable with traditional model checking tools due to the inter-independence of the formulas. The execution of the model checking framework in a high-performance architecture or cluster is feasible far beyond some synchronization points for the recollection of the results.

18.3. Parameters in Timed Model Checking

The use of parameters is not limited to qualitative model checking (parameters in genomical sequences), but also the inclusion of quantitative information has gained an increasing interest. The incorporation of timed or probabilistic bounds in the relations implicitly generates a set of inequalities and linear equations that are solvable with linear programming or, at least, can be approximated. Parameters in Timed CTL (TCTL) has already been studied from a theoretical point of view. The parameters are introduced for substituting clocks either in the formulas of temporal logic (a pure *parametric timed temporal logic* [3, 171]) or in the model of a timed automaton (a *parametrized timed automaton* [9]), and in some cases, in both formulas and model [32]. These variables are used for checking abstract temporal relations with no particular value of time.

The model checking algorithms for solving a parametric timed temporal formula are polynomial for any fixed number of parameters in the specifications [3]. Although there are situations in which the result is hard to compute over a parametrized timed automaton, there exist some cases where the emptiness problem is still decidable, for example, when the parametrized timed automaton has a single parametric clock. Nevertheless, the model checking problem becomes undecidable with the inclusion of more than three clocks to the model [9] or when the equality of formulas is allowed in the logic [32]. Moby/DC [58] is a tool for managing a simplified version of timed parametric model checking.

18.4. Parameters in Probabilistic Model Checking

Finally, current researches in probabilistic model checking are centered in systems in which the probabilities associated with the transitions of a model are also parameters [55, 108]. The studies of parameters in the model are specialized for discrete time Markov chains, although they oppose in the strategy: in [55] they represent the path probabilities of unnested operators with regular expressions restricted to the domain of rational numbers, while in [108] they work with polynomials and real numbers.

The regular expressions of the first case allow to manipulate parameters symbolically through the application of standard model checking algorithms. Particularly for the second approach, the complexity is comparable to a parametric timed automata: knowing the existence of an instance that satisfies a parametric probabilistic formula is decidable in exponential time with respect to the number of parameters. Searching for that particular instance of the parameters is generally unsolvable, but in some cases the valuations can be obtained by posing restrictions

to the system, for example, limiting the number of parameters in the polynomials to two [109].

The questions solved by parametric probabilistic model checking revolves around a) the search of valuations that maximizes (or minimizes) the probability of reaching a certain state, or b) the search of valuations that reach a certain state with an exact probability. The extraction of such valuations comes from the resolution of linear equation systems appeared from the concatenation of parameters in the states/transitions of the model.

PARAM [87] is a parametric add-on for PRISM that returns polynomials and rational functions as output results; it uses models where the transitions are defined with probabilistic parameters. A clear application domain is the definition of DNA substitution models: we use a DNA mutation template for scoring trees with MLE instead of giving particular valuations to the transitions. Probabilistic parameters in the specifications are already supported by model checking tools such as PRISM. PRISM helps for the inference of maximum or minimum likelihoods in Markov chains defined without parameters in the transitions.

Bibliography

[1] Susan M. Adams, Turi E. King, Elena Bosch, and Mark A. Jobling. The case of the unreliable SNP: Recurrent back-mutation of Y-chromosomal marker P25 through gene conversion. *Forensic Science International*, 159(1):14–20, 2006.

[2] Loredana Afanasiev, Massimo Franceschet, Maarten Marx, and Maarten de Rijke. CTL model checking for processing simple XPath queries. In *Proceedings 11th International Symposium on Temporal Representation and Reasoning*, pages 117–124. IEEE, 2004.

[3] Allen E. Emerson and Richard J. Trefler. Parametric quantitative temporal reasoning. In *Proceedings 14th Annual Symposium on Logic in Computer Science*, pages 336 –343. IEEE, 1999.

[4] Elizabeth S. Allman and John A. Rhodes. Trees, fast and accurate. *Science*, 327(5971):1334–1335, 2010.

[5] Musab AlTurki and José Meseguer. PVeStA;: A parallel statistical model checking and quantitative analysis tool. In Andrea Corradini, Bartek Klin, and Corina Cîrstea, editors, *Proceedings 4th International Conference on Algebra and Coalgebra in Computer Science*, volume 6859 of *LNCS*, pages 386–392. Springer, Berlin, 2011.

[6] Rajeev Alur, Costas Courcoubetis, and David Dill. Model-checking for real-time systems. In *Proceedings 5th Annual Symposium on Logic in Computer Science*, pages 414 –425. IEEE, 1990.

[7] Rajeev Alur, Costas Courcoubetis, and David Dill. Model-checking in dense real-time. *Information and Computation*, 104(1):2–34, 1993.

[8] Rajeev Alur and Thomas Henzinger. Logics and models of real time: A survey. In J.W. Bakker, Cornelis Huizing, Willem P. Roever, and Grzegorz Rozenberg, editors, *Proceedings REX Workshop Mook Real-Time: Theory in Practice*, volume 600 of *LNCS*, pages 74–106. Springer, Berlin, 1992.

[9] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. Parametric real-time reasoning. In *Proceedings 25th ACM Annual Symposium on Theory of Computing*, pages 592–601. ACM, 1993.

[10] Jorge Alvarez Jarreta, Elvira Mayordomo, and Eduardo Ruiz Pesini. PHYSER: An algorithm to detect sequencing errors from phylogenetic information. In Miguel P. Rocha, Nicholas Luscombe, Florentino Fdez-Riverola, and Juan M. Corchado Rodríguez, editors, *Proceedings 6th International Conference on Practical Applications of Computational Biology and Bioinformatics*, volume 154 of *Advances in Intelligent and Soft Computing*, pages 105–112. Springer, Berlin, 2012.

[11] Jeff Arendt and David Reznick. Convergence and parallelism reconsidered: What have we learned about the genetics of adaptation? *Trends in Ecology and Evolution*, 23(1):26–32, 2008.

[12] Arie Gurfinkel, Marsha Chechik, and Benet Devereux. Temporal logic query checking: A tool for model exploration. *IEEE Transactions on Software Engineering*, 29(10):898–914, 2003.

[13] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. The MIT Press, 2008.

[14] Hans-Jürgen Bandelt, Päivi Lahermo, Martin Richards, and Vincent Macaulay. Detecting errors in mtDNA data by phylogenetic analysis. *International Journal of Legal Medicine*, 115(2):64–69, 2001.

[15] Jiri Barnat, Lubos Brim, Milan Ceska, and Petr Rockai. DiVinE: Parallel distributed model checker. In *Proceedings 9th International Workshop on Parallel and Distributed Methods in Verification and 2nd International Workshop on High Performance Computational Systems Biology*, pages 4–7. IEEE, 2010.

[16] Jiri Barnat, Lubos Brim, Adam Krejci, Adam Streck, David Safranek, Martin Vejnar, and Tomas Vejpustek. On parameter synthesis by parallel model checking. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9(3):693–705, 2012.

[17] Timothy G. Barraclough and Sean Nee. Phylogenetics and speciation. *Trends in Ecology and Evolution*, 16(7):391–399, 2001.

[18] David Baum. Reading a phylogenetic tree: The meaning of monophyletic groups. *Nature Education*, 1(1):190, 2008.

[19] Cynthia M. Beall. Tibetan and Andean patterns of adaptation to high-altitude hypoxia. *Human Biology*, 72(1):201–228, 2000.

[20] Cynthia M. Beall, Kijoung Song, Robert C. Elston, and Melvyn C. Goldstein. Higher offspring survival among Tibetan women with high oxygen saturation genotypes residing at 4,000 m. *Proceedings of the National Academy of Sciences of the United States of America*, 101(39):14300–14304, 2004.

[21] Gerd Behrmann, Alexandre David, and Kim Larsen. A tutorial on Uppaal. In Marco Bernardo and Flavio Corradini, editors, *Formal Methods for the Design of Real-Time Systems*, volume 3185 of *LNCS*, pages 33–35. Springer, Berlin, 2004.

[22] Carlo Bellettini, Matteo Camilli, Lorenzo Capra, and Mattia Monga. Distributed CTL model checking in the cloud. *arXiv preprint arXiv:1310.6670*, 2013.

[23] Dennis A. Benson, Ilene Karsch Mizrachi, Karen Clark, David J. Lipman, James Ostell, and Eric W. Sayers. GenBank. *Nucleic Acids Research*, 40(D1):D48–D53, 2012.

[24] Girish Bhat, Rance Cleaveland, and Orna Grumberg. Efficient on-the-fly model checking for CTL. In *Proceedings 10th Annual Symposium on Logic in Computer Science*, pages 388 – 397. IEEE, 1995.

[25] Abigail Bigham, Marc Bauchet, Dalila Pinto, Xianyun Mao, Joshua M. Akey, Rui Mei, Stephen W. Scherer, Colleen G. Julian, Megan J. Wilson, David López Herráez, et al. Identifying signatures of natural selection in Tibetan and Andean populations using dense genome scan data. *PLoS Genetics*, 6(9):e1001116, 2010.

[26] Roberto Blanco, Gregorio de Miguel Casado, José Ignacio Requeno, and José Manuel Colom. Temporal logics for phylogenetic analysis via model checking. In *Proceedings IEEE International Workshop on Mining and Management of Biological and Health Data*, pages 152 – 157. IEEE, 2010.

[27] Roberto Blanco, Elvira Mayordomo, Julio Montoya, and Eduardo Ruiz Pesini. Rebooting the human mitochondrial phylogeny: An automated and scalable methodology with expert knowledge. *BMC Bioinformatics*, 12(1):174–186, 2011.

[28] Maria Luisa Bonet and Katherine St John. Efficiently calculating evolutionary tree measures using SAT. In Oliver Kullmann, editor, *Proceedings*

12th International Conference on Theory and Applications of Satisfiability Testing, volume 5584 of *LNCS*, pages 4–17. Springer, Berlin, 2009.

[29] Remco Bouckaert, Philippe Lemey, Michael Dunn, Simon J Greenhill, Alexander V Alekseyenko, Alexei J Drummond, Russell D Gray, Marc A Suchard, and Quentin D Atkinson. Mapping the origins and expansion of the Indo-European language family. *Science*, 337(6097):957–960, 2012.

[30] Mohand Cherif Boukala and Laure Petrucci. Distributed model-checking and counterexample search for CTL logic. *International Journal of Critical Computer-Based Systems*, 3(1/2):44–59, 2012.

[31] Dragan Bošnački and Stefan Edelkamp. Model checking software: On some new waves and some evergreens. *International Journal on Software Tools for Technology Transfer*, 12(2):89–95, 2010.

[32] Véronique Bruyere and Jean-François Raskin. Real-time model-checking: Parameters everywhere. In Paritosh K. Pandya and Jaikumar Radhakrishnan, editors, *Proceedings 23rd Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 2914 of *LNCS*, pages 100–111. Springer, Berlin, 2003.

[33] Randal E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, 1986.

[34] Kevin L. Campbell, Jason E.E. Roberts, Laura N. Watson, Jörg Stetefeld, Angela M. Sloan, Anthony V. Signore, Jesse W. Howatt, Jeremy R.H. Tame, Nadin Rohland, Tong-Jian Shen, et al. Substitutions in woolly mammoth hemoglobin confer biochemical properties adaptive for cold tolerance. *Nature Genetics*, 42(6):536–540, 2010.

[35] Sergio Campos, Edmund M. Clarke, Wilfredo Marrero, Marius Minea, and Hiromi Hiraishi. Computing quantitative characteristics of finite-state real-time systems. In *Proceedings 15th Real-Time Systems Symposium*, pages 266–270. IEEE, 1994.

[36] Luigi Luca Cavalli Sforza and Marcus W. Feldman. The application of molecular genetic approaches to the study of human evolution. *Nature Genetics*, 33:266–275, 2003.

[37] Luigi Luca Cavalli Sforza, Alberto Piazza, Paolo Menozzi, and Joanna Mountain. Reconstruction of human evolution: Bringing together genetic, archaeological, and linguistic data. *Proceedings of the National Academy of Sciences*, 85(16):6002–6006, 1988.

[38] William Chan. Temporal-logic queries. In E. Emerson and Aravinda Sistla, editors, *Proceedings 12th International Conference on Computer Aided Verification*, volume 1855 of *LNCS*, pages 450–463. Springer, Berlin, 2000.

[39] Krishnendu Chatterjee, Pallab Dasgupta, and P. P. Chakrabarti. Complexity of compositional model checking of computation tree logic on simple structures. In Arunabha Sen, Nabanita Das, Sajal K. Das, and Bhabani P. Sinha, editors, *Proceedings 6th International Workshop of Distributed Computing*, volume 3326 of *LNCS*, pages 102–113. Springer, Berlin, 2005.

[40] Marsha Chechik and Arie Gurfinkel. TLQSolver: A temporal logic query checker. In Warren A. Hunt Jr. and Fabio Somenzi, editors, *Proceedings 15th International Conference on Computer Aided Verification*, volume 2725 of *LNCS*, pages 210–214. Springer, Berlin, 2003.

[41] Gareth Chelvanayagam, Andreas Eggenschwiler, Lukas Knecht, Gaston H. Gonnet, and Steven A. Benner. An analysis of simultaneous variation in protein structures. *Protein Engineering*, 10(4):307–316, 1997.

[42] Anna Cho. *Constructing phylogenetic trees using maximum likelihood.* PhD thesis, Scripps Senior Theses, 2012.

[43] Jan Chomicki and David Toman. *Temporal logic in information systems.* Springer, 1998.

[44] Jan Chomicki, David Toman, and Michael H. Böhlen. Querying ATSQL databases with temporal logic. *ACM Transactions on Database Systems*, 26(2):145–178, 2001.

[45] Alessandro Cimatti, Edmund M. Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. NuSMV 2: An opensource tool for symbolic model checking. In Ed Brinksma and Kim Larsen, editors, *Proceedings 14th International Conference on Computer Aided Verification*, volume 2404 of *LNCS*, pages 241–268. Springer, Berlin, 2002.

[46] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In Dexter Kozen, editor, *Proceedings Workshop on Logics of Programs*, volume 131 of *LNCS*, pages 52–71. Springer, Berlin, 1981.

[47] Edmund M. Clarke, David E. Long, and Kenneth L. McMillan. Compositional model checking. In *Proceedings 4th Annual Symposium on Logic in Computer Science*, pages 353–362. IEEE, 1989.

[48] Jamieson M. Cobleigh, George S. Avrunin, and Lori A. Clarke. Breaking up is hard to do: An evaluation of automated assume-guarantee reasoning. *ACM Transactions on Software Engineering and Methodology*, 17(2):7:1–7:52, 2008.

[49] Peter J. A. Cock, Tiago Antao, Jeffrey T. Chang, Brad A. Chapman, Cymon J. Cox, Andrew Dalke, Iddo Friedberg, Thomas Hamelryck, Frank Kauff, Bartek Wilczynski, et al. Biopython: Freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423, 2009.

[50] Francisco M. Codoñer and Mario A. Fares. Why should we care about molecular coevolution? *Evolutionary Bioinformatics Online*, 4:29–38, 2008.

[51] Francis S. Collins. What we do and don't know about'race','ethnicity', genetics and health at the dawn of the genome era. *Nature Genetics*, 36:S13–S15, 2004.

[52] Daniel H. Huson, Regula Rupp, and Celine Scornavacca. *Phylogenetic networks: Concepts, algorithms and applications*. Cambridge University Press, 2011.

[53] Diego Darriba, Guillermo L Taboada, Ramón Doallo, and David Posada. jModelTest 2: More models, new heuristics and parallel computing. *Nature Methods*, 9(8):772–772, 2012.

[54] Charles Darwin. *On the origin of species by means of natural selection, or the preservation of favoured races in the struggle of life*. John Murray, 1859.

[55] Conrado Daws. Symbolic and parametric model checking of discrete-time Markov chains. In Zhiming Liu and Keijiro Araki, editors, *Proceedings 1st International Colloquium on Theoretical Aspects of Computing*, volume 3407 of *LNCS*, pages 280–294. Springer, Berlin, 2005.

[56] Elisabetta De Maria, François Fages, Aurélien Rizk, and Sylvain Soliman. Design, optimization and predictions of a coupled model of the cell cycle, circadian clock, DNA repair system, irinotecan metabolism and exposure control under temporal logic constraints. *Theoretical Computer Science*, 412(21):2108–2127, 2011.

[57] US Department of Health. *Healthy people 2010*. Government Printing Office, 2000.

[58] Henning Dierks and Josef Tapken. Moby/DC–a tool for model-checking parametric real-time specifications. In Hubert Garavel and John Hatcliff, editors, *Proceedings 9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 2619 of *LNCS*, pages 271–277. Springer, Berlin, 2003.

[59] Susanna Donatelli, Serge Haddad, and Jeremy Sproston. Model checking timed and stochastic properties with CSL^{TA}. *IEEE Transactions on Software Engineering*, 35(2):224–240, 2009.

[60] Agostino Dovier and Elisa Quintarelli. Model-checking based data retrieval. In Giorgio Ghelli and Gösta Grahne, editors, *Proceedings 8th International Workshop on Database Programming Languages*, volume 2397 of *LNCS*, pages 62–77. Springer, Berlin, 2002.

[61] Alexei J. Drummond and Andrew Rambaut. BEAST: Bayesian evolutionary analysis by sampling trees. *BMC Evolutionary Biology*, 7(1):214–221, 2007.

[62] Robert C. Edgar. MUSCLE: Multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32(5):1792–1797, 2004.

[63] Orna Grumberg Edmund M. Clarke and Doron A. Peled. *Model checking*. The MIT Press, 2000.

[64] Edward O. Wiley and Bruce S. Lieberman. *Phylogenetics: The theory and practice of phylogenetic systematics*. Wiley, 1981.

[65] Nathan Ellis, Susan Ciocci, and James German. Back mutation can produce phenotype reversion in Bloom syndrome somatic cells. *Human Genetics*, 108(2):167–173, 2001.

[66] E. Allen Emerson, Aloysius K. Mok, A. Prasad Sistla, and Jai Srinivasan. Quantitative temporal reasoning. In Edmund M. Clarke and Robert P. Kurshan, editors, *Proceedings 2nd International Workshop on Computer Aided Verification*, volume 531 of *LNCS*, pages 136–145. Springer, Berlin, 1991.

[67] Emmanuel Paradis. *Analysis of phylogenetics and evolution with R (Use R!)*. Springer, 2012.

[68] Laurent Excoffier, Guillaume Laval, and Stefan Schneider. Arlequin (version 3.0): An integrated software package for population genetics data analysis. *Evolutionary Bioinformatics Online*, 1:47–50, 2005.

[69] Joseph Felsenstein. Evolutionary trees from DNA sequences: A maximum likelihood approach. *Journal of Molecular Evolution*, 17(6):368–376, 1981.

[70] Joseph Felsenstein. PHYLIP-phylogeny inference package (version 3.2). *Cladistics*, 5:164–166, 1989.

[71] Joseph Felsenstein. *Inferring phylogenies*. Sinauer, 2003.

[72] D. Fenna, L. Mix, O. Schaefer, and J. A. L. Gilbert. Ethanol metabolism in various racial groups. *Canadian Medical Association Journal*, 105(5):472–475, 1971.

[73] Fernando A. F. Braz, Jader S. Cruz, Alessandra C. Faria Campos, and Sergio Campos. Probabilistic model checking analysis of Palytoxin effects on cell energy reactions of the Na+/K+ATPase. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 10(6):1530–1541, 2013.

[74] Walter M. Fitch. Toward defining the course of evolution: Minimum change for a specific tree topology. *Systematic Biology*, 20(4):406–416, 1971.

[75] Walter M. Fitch. Uses for evolutionary trees. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 349(1327):93–102, 1995.

[76] Peter Forster. Ice Ages and the mitochondrial DNA chronology of human dispersals: A review. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 359(1442):255–264, 2004.

[77] K. Fukami Kobayashi, D. R. Schreiber, and S. A. Benner. Detecting compensatory covariation signals in protein evolution using reconstructed ancestral sequences. *Journal of Molecular Biology*, 319(3):729–743, 2002.

[78] Dov M. Gabbay and Amir Pnueli. A sound and complete deductive system for CTL* verification. *Logic Journal of IGPL*, 16(6):499–536, 2008.

[79] Mihaela Gheorghiu and Arie Gurfinkel. Tlq: A query solver for states. In *Tools and Posters Session at the 14th International Symposium on Formal Methods*, 2006.

[80] Gonzalo Giribet. TNT: Tree analysis using new technology. *Systematic Biology*, 54(1):176–178, 2005.

[81] T. Ryan Gregory. *Genome size evolution in animals*, volume 1, chapter Chapter 1, pages 4–87. Elsevier, San Diego, CA, 2005.

[82] T. Ryan Gregory. *Genome size evolution in plants*, volume 1, chapter Chapter 2, pages 89–162. Elsevier, San Diego, CA, 2005.

[83] T Ryan Gregory. Animal genome size database. `http://www.genomesize.com`, 2014.

[84] Orna Grumberg, Tamir Heyman, and Assaf Schuster. Distributed symbolic model checking for μ-calculus. In Gérard Berry, Hubert Comon, and Alain Finkel, editors, *Proceedings 13th International Conference on Computer Aided Verification*, volume 2102 of *LNCS*, pages 350–362. Springer, Berlin, 2001.

[85] Orna Grumberg and Helmut Veith. *25 years of model checking: History, achievements, perspectives*. Springer, 2008.

[86] Jenny Gu and Philip E. Bourne. *Structural bioinformatics*. Wiley-Blackwell, 2009.

[87] Ernst Moritz Hahn, Holger Hermanns, Björn Wachter, and Lijun Zhang. PARAM: A model checker for parametric Markov models. In Tayssir Touili, Byron Cook, and Paul Jackson, editors, *Proceedings 22nd International Conference on Computer Aided Verification*, volume 6174 of *LNCS*, pages 660–664. Springer, Berlin, 2010.

[88] Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.

[89] Willi Hennig. *Phylogenetic systematics*. University of Illinois Press, 1999.

[90] Thomas A Henzinger, Zohar Manna, and Amir Pnueli. Timed transition systems. In J.W. Bakker, Cornelis Huizing, Willem P. Roever, and Grzegorz Rozenberg, editors, *Proceedings REX Workshop Mook Real-Time: Theory in Practice*, volume 600 of *LNCS*, pages 226–251. Springer, Berlin, 1992.

[91] David M. Hillis and James J. Bull. An empirical test of bootstrapping as a method for assessing confidence in phylogenetic analysis. *Systematic Biology*, 42(2):182–192, 1993.

[92] Clare Holden and Ruth Mace. Phylogenetic analysis of the evolution of lactose digestion in adults. *Human Biology*, 81(5/6):597–619, 2009.

[93] Gerard J. Holzmann. The model checker SPIN. *IEEE Transactions on Software Engineering*, 23(5):279–295, 1997.

[94] Richard R. Hudson. Gene genealogies and the coalescent process. *Oxford Surveys in Evolutionary Biology*, 7(1):1–44, 1990.

[95] Keith Hunley, Claire Bowern, and Meghan Healy. Rejection of a serial founder effects model of genetic and linguistic coevolution. *Proceedings of the Royal Society B: Biological Sciences*, 279(1736):2281–2288, 2012.

[96] Cornelia Inggs and Howard Barringer. CTL* model checking on a shared-memory architecture. *Formal Methods in System Design*, 29(2):135–155, 2006.

[97] Catherine J. E. Ingram, Charlotte A. Mulcare, Yuval Itan, Mark G. Thomas, and Dallas M. Swallow. Lactose digestion and the evolutionary genetics of lactase persistence. *Human Genetics*, 124(6):579–591, 2009.

[98] David N. Jansen, Joost-Pieter Katoen, Marcel Oldenkamp, Mariëlle Stoelinga, and Ivan Zapreev. How fast and fat is your probabilistic model checker? an experimental performance comparison. In Karen Yorav, editor, *Proceedings 3rd International Haifa Verification Conference on Hardware and Software, Verification and Testing*, volume 4899 of *LNCS*, pages 69–85. Springer, Berlin, 2008.

[99] Jerry R. Burch, Edmund M. Clarke, David E. Long, Kenneth L. Mcmillan, and David L. Dill. Symbolic model checking for sequential circuit verification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(4):401–424, 1994.

[100] Sumit K. Jha, Edmund M. Clarke, Christopher J. Langmead, Axel Legay, André Platzer, and Paolo Zuliani. A Bayesian approach to model checking biological systems. In Pierpaolo Degano and Roberto Gorrieri, editors, *Proceedings 7th International Conference on Computational Methods in Systems Biology*, volume 5688 of *LNCS*, pages 218–234. Springer, Berlin, 2009.

[101] Mark A. Jobling and Chris Tyler Smith. The human Y chromosome: An evolutionary marker comes of age. *Nature Reviews Genetics*, 4(8):598–612, 2003.

[102] Katherine St John. Comparing phylogenetic trees. In *EMBO Workshop on Current Challenges and Problems in Phylogenetics*, Cambridge, UK, 2007. Isaac Newton Institute.

[103] John Heath, Marta Kwiatkowska, Gethin Norman, David. Parker, and Oksana Tymchyshyn. Probabilistic model checking of complex biological pathways. In Corrado Priami, editor, *Proceedings 4th International Conference on Computational Methods in Systems Biology*, volume 4210 of *Lecture Notes in Bioinformatics*, pages 32–47. Springer, Berlin, 2006.

[104] John Heath, Marta Kwiatkowska, Gethin Norman, David Parker, and Oksana Tymchyshyn. Probabilistic model checking of complex biological pathways. *Theoretical Computer Science*, 319(3):239–257, 2008.

[105] Rob Knight, Peter Maxwell, Amanda Birmingham, Jason Carnes, J. Gregory Caporaso, Brett C. Easton, Michael Eaton, Micah Hamady, Helen Lindsay, Zongzhi Liu, et al. PyCogent: A toolkit for making sense from sequence. *Genome Biology*, 8(8):1–16, 2007.

[106] Marta Kwiatkowska, Gethin Norman, and David Parker. Stochastic model checking. In Marco Bernardo and Jane Hillston, editors, *7th International School on Formal Methods for Performance Evaluation*, volume 4486 of *LNCS*, pages 220–270. Springer, Berlin, 2007.

[107] Christopher James Langmead and Sumit Kumar Jha. Predicting protein folding kinetics via temporal logic model checking. In Raffaele Giancarlo and Sridhar Hannenhalli, editors, *Proceedings 7th Workshop on Algorithms in Bioinformatics*, volume 4645 of *LNCS*, pages 252–264. Springer, Berlin, 2007.

[108] Ruggero Lanotte, Andrea Maggiolo Schettini, and Angelo Troina. Decidability results for parametric probabilistic transition systems with an application to security. In *Proceedings 2th International Conference on Software Engineering and Formal Methods*, pages 114–121. IEEE, 2004.

[109] Ruggero Lanotte, Andrea Maggiolo Schettini, and Angelo Troina. Parametric probabilistic transition systems for system design and analysis. *Formal Aspects of Computing*, 19(1):93–109, 2007.

[110] Guillaume Lecointre and Hervé Le Guyader. *The tree of life: A phylogenetic classification*. Harvard University Press, 2006.

[111] Ivica Letunic and Peer Bork. Interactive Tree Of Life (iTOL): An online tool for phylogenetic tree display and annotation. *Bioinformatics*, 23(1):127–128, 2007.

[112] Orna Lichtenstein, Amir Pnueli, and Lenore D. Zuck. The glory of the past. In Rohit Parikh, editor, *Proceedings Workshop on Logic of Programs*, volume 193 of *LNCS*, pages 196–218. Springer, Berlin, 1985.

[113] Carl von Linné and Laurentii Salvii. *Caroli Linnaei ... Species plantarum*, volume v. 1. Holmiae: Impensis Laurentii Salvii, 1753.

[114] Carl von Linné and Laurentii Salvii. *Caroli Linnaei ... Species plantarum*, volume v. 2. Holmiae: Impensis Laurentii Salvii, 1753.

[115] Carl von Linné and Laurentii Salvii. *Caroli Linnaei...Systema naturae per regna tria naturae*, volume v.1. Holmiae: Impensis Laurentii Salvii, 1758.

[116] Carl von Linné and Laurentii Salvii. *Caroli Linnaei...Systema naturae per regna tria naturae*, volume v.2. Holmiae: Impensis Laurentii Salvii, 1759.

[117] Pietro Lio and Nick Goldman. Models of molecular evolution and phylogeny. *Genome Research*, 8(12):1233–1244, 1998.

[118] Mark V. Lomolino, Brett R. Riddle, and James H. Brown. *Biogeography*. Sinauer, 2006.

[119] Wayne P. Maddison. Gene trees in species trees. *Systematic Biology*, 46(3):523–536, 1997.

[120] Wayne P. Maddison and David R. Maddison. *Mesquite: A modular system for evolutionary analysis. Version 2.75*, 2011.

[121] Harry Mangalam. The Bio* toolkits - a brief overview. *Briefings in Bioinformatics*, 3(3):296–302, 2002.

[122] Zohar Manna and Amir Pnueli. *The temporal logic of reactive and concurrent systems: Specification*. Springer, 1991.

[123] Marta Kwiatkowska, Gethin Norman, and David Parker. Using probabilistic model checking in systems biology. *ACM SIGMETRICS Performance Evaluation Review*, 35(4):14–21, 2008.

[124] Marta Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *Proceedings 23rd International Conference on Computer Aided Verification*, volume 6806 of *LNCS*, pages 585–591. Springer, Berlin, 2011.

[125] Kenneth L. McMillan. A methodology for hardware verification using compositional model checking. *Science Computational Programming*, 37(1–3):279–309, 2000.

[126] Igor Melatti, Robert Palmer, Geoffrey Sawaya, Yu Yang, Robert Mike Kirby, and Ganesh Gopalakrishnan. Parallel and distributed model checking in Eddy. In Antti Valmari, editor, *Proceedings 3th International Conference on Model Checking Software*, volume 3925 of *LNCS*, pages 108–125. Springer, Berlin, 2006.

[127] Igor Melatti, Robert Palmer, Geoffrey Sawaya, Yu Yang, Robert Mike Kirby, and Ganesh Gopalakrishnan. Parallel and distributed model checking in Eddy. *International Journal on Software Tools for Technology Transfer*, 11(1):13–25, 2009. Springer.

[128] Pedro T. Monteiro, Delphine Ropers, Radu Mateescu, Ana T. Freitas, and Hidde Jong. Temporal logic patterns for querying dynamic models of cellular interaction networks. *Bioinformatics*, 24(16):i227–i233, 2008.

[129] Julio Montoya, Ester López Gallardo, Carmen Díez Sánchez, Manuel J. López Pérez, and Eduardo Ruiz Pesini. 20 years of human mtDNA pathologic point mutations: Carefully reading the pathogenicity criteria. *Biochimica et Biophysica Acta*, 1787(5):476–483, 2009.

[130] Arne O. Mooers and Stephen B. Heard. Inferring evolutionary process from phylogenetic tree shape. *Quarterly Review of Biology*, 72(21):31–54, 1997.

[131] Richard Nichols. Gene trees and species trees are not the same. *Trends in Ecology and Evolution*, 16(7):358–364, 2001.

[132] Pekka Pamilo and Masatoshi Nei. Relationships between gene trees and species trees. *Molecular Biology and Evolution*, 5(5):568–583, 1988.

[133] John M. Pearce. Minding the gap: Frequency of indels in mtDNA control region sequence data and influence on population genetic analyses. *Molecular Ecology*, 15(2):333–341, 2006.

[134] Amir Pnueli. The temporal logic of programs. In *Proceedings 18th Annual Symposium on Foundations of Computer Science*, pages 46–57. IEEE, 1977.

[135] Amir Pnueli. *Logics and Models of Concurrent Systems*, chapter In transition from global to modular temporal reasoning about programs, pages 123–144. Springer, New York, NY, 1985.

[136] Simone Pompei, Vittorio Loreto, and Francesca Tria. Phylogenetic properties of RNA viruses. *PloS One*, 7(9):e44849, 2012.

[137] Andrew Rambaut. How to read a phylogenetic tree, 2013.

[138] José Ignacio Requeno, Roberto Blanco, Gregorio de Miguel Casado, and José Manuel Colom. Phylogenetic analysis using an SMV tool. In Miguel P. Rocha, Juan M. Corchado Rodrìguez, Florentino Fdez-Riverola, and Alfonso Valencia, editors, *Proceedings 5th International Conference on Practical Applications of Computational Biology and Bioinformatics*, volume 93 of *Advances in Intelligent and Soft Computing*, pages 167–174. Springer, Berlin, 2011.

[139] José Ignacio Requeno, Roberto Blanco, Gregorio de Miguel Casado, and José Manuel Colom. Sliced model checking for phylogenetic analysis. In Miguel P. Rocha, Nicholas Luscombe, Florentino Fdez-Riverola, and Juan M. Corchado Rodríguez, editors, *Proocedings 6th International Conference on Practical Applications of Computational Biology and Bioinformatics*, volume 154 of *Advances in Intelligent and Soft Computing*, pages 95–103. Springer, Berlin, 2012.

[140] José Ignacio Requeno and José Manuel Colom. Model checking software for phylogenetic trees using distribution and database methods. *Journal of Integrative Bioinformatics*, 10(3):229–233, 2013.

[141] José Ignacio Requeno and José Manuel Colom. Speeding up phylogenetic model checking. In Mohd Saberi Mohamad, Loris Nanni, Miguel P. Rocha, and Florentino Fdez-Riverola, editors, *Proceedings 7th International Conference on Practical Applications of Computational Biology and Bioinformatics*, volume 222 of *Advances in Intelligent Systems and Computing*, pages 119–126. Springer, Berlin, 2013.

[142] José Ignacio Requeno and José Manuel Colom. Timed and probabilistic model checking over phylogenetic trees. In Miguel P. Rocha et al., editors, *Proceedings 8th International Conference on Practical Applications of Computational Biology and Bioinformatics*, Advances in Intelligent and Soft Computing. Springer, Berlin, 2014.

[143] José Ignacio Requeno, Gregorio de Miguel Casado, Roberto Blanco, and José Manuel Colom. Temporal logics for phylogenetic analysis via model checking. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 10(4):1058–1070, 2013.

[144] Martin B. Richards, Vincent A. Macaulay, Hans-Jürgen Bandelt, and Bryan C. Sykes. Phylogeography of mitochondrial DNA in western Europe. *Annals of Human Genetics*, 62(3):241–260, 1998.

[145] Aurélien Rizk, Grégory Batt, François Fages, and Sylvain Soliman. On a continuous degree of satisfaction of temporal logic formulae with applications to systems biology. In Monika Heiner and Adelinde M. Uhrmacher, editors, *Proceedings 6th International Conference on Computational Methods in Systems Biology*, volume 5307 of *LNCS*, pages 251–268. Springer, Berlin, 2008.

[146] D. F. Robinson and Leslie R. Foulds. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53(1):131–147, 1981.

[147] Fredrik Ronquist, Maxim Teslenko, Paul van der Mark, Daniel L. Ayres, Aaron Darling, Sebastian Höhna, Bret Larget, Liang Liu, Marc A. Suchard, and John P. Huelsenbeck. MrBayes 3.2: Efficient Bayesian phylogenetic inference and model choice across a large model space. *Systematic Biology*, 61(3):539–542, 2012.

[148] Naruya Saitou and Masatoshi Nei. The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406–425, 1987.

[149] Philippe Schnoebelen. The complexity of temporal logic model checking. *Advances in Modal Logic*, 4:393–436, 2002.

[150] Roberto Segala and Nancy Lynch. Probabilistic simulations for probabilistic processes. In Bengt Jonsson and Joachim Parrow, editors, *Proceedings 5th International Conference on Concurrency Theory*, volume 836 of *LNCS*, pages 481–496. Springer, Berlin, 1994.

[151] German Shegalov. CTL model checking in database cloud. Oracle Corporation.

[152] Tanja Stadler. *Evolving trees: Models for speciation and extinction in phylogenetics*. PhD thesis, Zentrum Mathematik, Technische Universität München, 2008.

[153] Jason E. Stajich, David Block, Kris Boulez, Steven E. Brenner, Stephen A. Chervitz, Chris Dagdigian, Georg Fuellen, James G. R. Gilbert, Ian Korf, Hilmar Lapp, et al. The Bioperl toolkit: Perl modules for the life sciences. *Genome Research*, 12(10):1611–1618, 2002.

[154] Alexandros P. Stamatakis, Thomas Ludwig, and Harald Meier. The AxML program family for maximum likelihood-based phylogenetic tree inference. *Concurrency and Computation: Practice and Experience*, 16(9):975–988, 2004.

[155] Alexandros P. Stamatakis, Thomas Ludwig, and Harald Meier. RAxML-III: A fast program for maximum likelihood-based inference of large phylogenetic trees. *Bioinformatics*, 21(4):456–463, 2005.

[156] Mike Steel and Andy McKenzie. Properties of phylogenetic trees generated by Yule-type speciation models. *Mathematical Biosciences*, 170(1):91–112, 2001.

[157] William J. Stewart. *Probability, Markov chains, queues, and simulation: The mathematical basis of performance modeling.* Princeton University Press, 2009.

[158] M. Stich and S. C. Manrubia. Topological properties of phylogenetic trees in evolutionary models. *The European Physical Journal B*, 70(4):583–592, 2009.

[159] Seung-Jin Sul, Suzanne Matthews, and Tiffani L. Williams. Using tree diversity to compare phylogenetic heuristics. *BMC Bioinformatics*, 10(Suppl 4):S3, 2009.

[160] Dallas M. Swallow. Genetics of lactase persistence and lactose intolerance. *Annual Review of Genetics*, 37(1):197–219, 2003.

[161] David L. Swofford. *PAUP*. Phylogenetic analysis using parsimony (* and other methods). Version 4*, 2003.

[162] Julie D. Thompson, Toby Gibson, and Des G. Higgins. Multiple sequence alignment using ClustalW and ClustalX. *Current Protocols in Bioinformatics*, 2:2–3, 2002.

[163] Sarah A. Tishkoff, Floyd A. Reed, Alessia Ranciaro, Benjamin F. Voight, Courtney C. Babbitt, Jesse S. Silverman, Kweli Powell, Holly M. Mortensen, Jibril B. Hirbo, Maha Osman, et al. Convergent adaptation of human lactase persistence in Africa and Europe. *Nature Genetics*, 39(1):31–40, 2006.

[164] Chris Tuffley and Mike Steel. Links between maximum likelihood and maximum parsimony under a simple model of site substitution. *Bulletin of Mathematical Biology*, 59(3):581–607, 1997.

[165] Ulrich Stern and David L. Dill. Parallelizing the Murφ verifier. *Formal Methods in System Design*, 18(2):117–129, 2001.

[166] Peter A. Underhill and Toomas Kivisild. Use of Y chromosome and mitochondrial DNA population structure in tracing human migrations. *Annual Review of Genetics*, 41:539–564, 2007.

[167] University of California Museum of Paleontology and the National Center for Science Education. Understanding Evolution, 2004.

[168] Mannis Van Oven and Manfred Kayser. Updated comprehensive phylogenetic tree of global human mitochondrial DNA variation. *Human Mutation*, 30(2):E386–E394, 2009.

[169] Kimmo Varpaaniemi, Jaakko Halme, Kari Hiekkanen, and Tino Pyssysalo. PROD reference manual. Technical report, Helsinki University of Technology, Digital Systems Laboratory, Espoo, Finland, 1995.

[170] Johann-Wolfgang Wägele. *Foundations of phylogenetic systematics*. Pfeil, 2005.

[171] Farn Wang. Parametric timing analysis for real-time systems. *Information and Computation*, 130(2):131–150, 1996.

[172] Yufeng Wu. Close lower and upper bounds for the minimum reticulate network of multiple phylogenetic trees. *Bioinformatics*, 26(12):i140–i148, 2010.

[173] Ming Yang, Yan Ge, Jiayan Wu, Jingfa Xiao, and Jun Yu. Coevolution study of mitochondria respiratory chain proteins: Toward the understanding of protein–protein interaction. *Journal of Genetics and Genomics*, 38(5):201–207, 2011.

[174] Ziheng Yang. *Computational molecular evolution*. Oxford University Press, 2006.

[175] Ziheng Yang and Bruce Rannala. Bayesian phylogenetic inference using DNA sequences: A Markov chain Monte Carlo method. *Molecular Biology and Evolution*, 14(7):717–724, 1997.

[176] Ziheng Yang and Bruce Rannala. Molecular phylogenetics: Principles and practice. *Nature Reviews Genetics*, 13(5):303–314, 2012.

[177] Xin Yi, Yu Liang, Emilia Huerta Sanchez, Xin Jin, Zha Xi Ping Cuo, John E Pool, Xun Xu, Hui Jiang, Nicolas Vinckenbosch, Thorfinn Sand Korneliussen, et al. Sequencing of 50 human exomes reveals adaptation to high altitude. *Science*, 329(5987):75–78, 2010.

[178] S.-J. Yin, T.-C. Cheng, C.-P. Chang, Y.-J. Chen, Y.-C. Chao, H.-S. Tang, T.-M. Chang, and C.-W. Wu. Human stomach alcohol and aldehyde dehydrogenases (ALDH): A genetic model proposed for ALDH III isozymes. *Biochemical Genetics*, 26(5-6):343–360, 1988.

[179] Tatiana Zerjal, Yali Xue, Giorgio Bertorelle, R. Spencer Wells, Weidong Bao, Suling Zhu, Raheel Qamar, Qasim Ayub, Aisha Mohyuddin, Songbin Fu, et al. The genetic legacy of the Mongols. *The American Journal of Human Genetics*, 72(3):717–721, 2003.

[180] Jianzhi Zhang and Sudhir Kumar. Detection of convergent and parallel evolution at the amino acid sequence level. *Molecular Biology and Evolution*, 14(5):527–536, 1997.

[181] Christian M. Zmasek and Sean R. Eddy. ATV: Display and manipulation of annotated phylogenetic trees. *Bioinformatics*, 17(4):383–384, 2001.