Temporal Logics for Phylogenetic Analysis via Model Checking

José Ignacio Requeno, Gregorio de Miguel Casado, Roberto Blanco, and José Manuel Colom

Abstract—The need for general-purpose algorithms for studying biological properties in phylogenetics motivates research into formal verification frameworks. Researchers can focus their efforts exclusively on evolution trees and property specifications. To this end, model checking, a mature automated verification technique originating in computer science, is applied to phylogenetic analysis. Our approach is based on three cornerstones: a logical modeling of the evolution with transition systems; the specification of both phylogenetic properties and trees using flexible temporal logic formulas; and the verification of the latter by means of automated computer tools. The most conspicuous result is the inception of a formal framework which allows for a symbolic manipulation of biological data (based on the codification of the taxa). Additionally, different logical models of evolution can be considered, complex properties can be specified in terms of the logical composition of others, and the refinement of unfulfilled properties as well as the discovery of new properties can be undertaken by exploiting the verification results. Some experimental results using a Symbolic Model Verifier support the feasibility of the approach.

Index Terms—phylogenetic analysis; formal verification; temporal logic; model checking

1 Introduction

During the last century, taxonomy and systematics have provided a wide range of methodologies which have contributed to a better understanding of the tree of life. Both disciplines make heavy use of empirical data, with proposition, verification and generalization of hypotheses over the reconstructed tree playing a central role in their application [1]. Phylogenetic trees continue to be useful abstractions for modeling evolution over time [2]. Computer science tools have upgraded the capabilities of biologists for their construction as well as for extracting and analyzing the implicit biological messages embedded on them [3], [4], [5]. To this day, one of the most relevant challenges that remains to be resolved is guaranteeing the validity of the inferred tree itself.

The wide range of heterogeneous methods and tools used by biologists in phylogenetic analysis tasks suggests the possibility of researching in a generic framework for heterogeneous hypothesis verification over trees. In this paper 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 [6]. 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. 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 to the property).

The application of model checking techniques in this paper assumes that a system is abstracted by a set of discrete reachable states and the transitions that allow the movement from one state to another. Therefore the model checking approach which we propose in the context of phylogeny assumes that a phylogenetic tree can be considered 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 model checking technique introduced in this paper should not be confused (or be identified) with the classic phylogenetic procedures focused on guaranteeing the validity of the inferred tree. In this last context, the validation of a phylogeny determines the goodness of fit of mutation models under different biological hypothesis. These biological models are based on the concept of evolutionary distance, allowing to explore tangible numerical relationships between sets of populations or individuals characterized by biological features such as DNA. This has led to tree-building through distance and character based methods so as to infer structural properties of interest to biologists [3], [4]. We will show how this validation can also be carried out by means of the proposed model checking techniques.

In computational biology, model checking has been applied to other related fields in which (mostly) quantitative properties over temporal data are analyzed [7], [8], [9], [10]. Notice that temporal logics embedding

The authors are with the Department of Computer Science and Systems Engineering (DIIS) and the Aragon Institute of Engineering Research (I3A), University of Zaragoza, c/María de Luna 1, 50018 Zaragoza, Spain E-mail: {nrequeno, gmiguel, robertob, jm}@unizar.es

the concept of evolutionary distance under the notion of time have already been proposed in biological system modeling [11]. This fact highlights the flexibility of formal logics for modeling biological processes by defining a suitable logic for each specific problem domain. To our best knowledge, the application of model checking techniques with qualitative temporal logics for phylogenetic analysis was firstly proposed in [12], [13].

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" [14] and there already exists a big pool of software tools for automated verification [6]. At first sight, the intrinsic complexity of the tasks under consideration is the main limitation of our approach.

This paper sets up the basis for introducing model checking as an unifying formalism that permits the phylogeneticist to focus on phylogenies and biological specifications using propositions of temporal logic, instead of focusing on implementation issues concerned with particular algorithms. The formalism is flexible enough to consider different kind of data structures and evolutionary models. It represents an opportunity to handle the increasing complexity of the structural properties required by biologists (e.g. the detection of potentially back mutations). Moreover, the idea of hypothesis verification in our approach works in a closed-loop helping the phylogeneticist to *refine* or *discover* properties using counterexamples obtained from unfulfilled properties.

The paper is arranged in six sections. Following this introduction, Section 2 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. Section 3 describes the logical specification of non-trivial structural properties of cladistic classification and sequence composition. The logic presented in this section can be extended for quantitative purposes. Section 4 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. Alternative representations of evolution and future work are explained in Section 5. Finally, Section 6 gathers the conclusions drawn from this research.

2 Bridging Worlds: Phylogenetics and Model Checking

Transition systems are considered powerful formal models for the study of concurrent systems [14, Def. 2.1]. 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. 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.

2.1 Evolution as a Transition System

At the highest level of abstraction, phylogenies represent partial models of the evolution of sets of living organisms. They can be represented as directed graphs, though phylogenetic trees suffice for most common purposes and will be adopted in this paper.

Trees offer a realistic model of aggregated evolution, in which each vertex represents an inferred state of the evolution characterized by biological sequences (e.g., DNA) or other information. Transformative events that modify heritable information give rise to new states which are reflected in oriented parent-child edges in the graph. Note that neither 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 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 sequence.

Trees are typically built from finite sets of words of uniform length resulting from alignment algorithms. These words, which represent present-time taxa, are required to be in bijective correspondence with the set of leaves of the trees.

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.

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 $M = (S, S_0, R, L)$, 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 $s \in S$, there exists $t \in S$ such that $(s,t) \in R$, 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 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*.

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:

- 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:

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

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

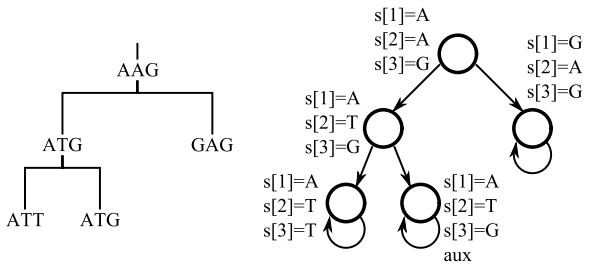

Fig. 1. Translation from a tree to a Kripke structure.

- R is the transition relation composed of the set of tree edges (directed from r) plus self-loops on the leaves: $R = E \cup \{(v,v) : \nexists (v,w) \in E \wedge v, w \in V\}$.
- 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 phylogenetic tree. The labeling of states with atomic propositions 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 unique state identifiers ensures uniqueness.

Fig. 1 illustrates this translation process from a (phylogenetic) rooted labeled tree (Def. 1) to a Kripke structure (Def. 2). It should be emphasized that our proposal can be used for other phylogenetic structures such as phylogenetic networks by reformulating Defs. 1–3 as required.

2.2 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 [14]. 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) 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

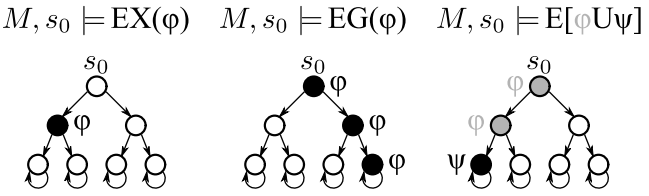

Fig. 2. Evaluation of temporal logic operators.

globally about the computation tree. *Computational Tree Logic* (CTL), a versatile exponent of the latter, has been widely adopted by the model checking community [15]. As phylogenies represent evolutionary processes that are mainly branching in nature, branching-time logics in general, and CTL in particular, are remarkably well suited to their description. 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 subset of logical operators (see Fig. 2).

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

$$\phi ::= p \mid \neg\phi \mid \phi \vee \phi \mid \mathbf{EX}(\phi) \mid \mathbf{EG}(\phi) \mid \mathbf{E}[\phi \mathbf{U} \phi] \quad (2)$$

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

A CTL formula ϕ represents a property that may be verified at certain states in the computation tree. A logic thus defined permits the formal expression of generic properties on evolving biological sequences and their eventual automated verification.

2.3 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 verification even for small systems, for which symbolic manipulation of systems and formulas are required [16]. 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 feedback provided by the model checking process helps us to discriminate between several phylogenetic structures expressed as a set of logical properties.

Going into greater detail, the key step of the verification process consists of the generation of a satisfiable set. Starting with a set of 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. The verification of a PTL formula consists of checking a simple property in the set generated by the fixed-point algorithm. For example, we can test if the initial state of the Kripke structure belongs to the result set. Counterexamples can be obtained by tracing back the appropriate path over the complementary set.

Summarizing, the variable assignments that make a logical formula true characterize the set of states satisfying the logical specification: the states are coded with the same variables used in the logical formulas. We are symbolically characterizing and manipulating sets of states by means of the properties satisfied by their components.

3 Phylogenetic Properties

This section describes a methodology to specify biological properties of phylogenies employing the previous logical framework. By abstraction, a non-trivial property can be broken down into simpler, meaningful ones, and synthesized from these. The benefits of such logical decompositions are twofold: firstly, they simplify formalization and favor readability; secondly, 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 the detection of common ancestors or monophyletic clades ("ape" [17]), speciation rates (extinction and diversification, Mesquite [18]) and the definition of mutation rates in DNA substitution models (Pycogent [19]).

Four classes of queries can be identified: *global* 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 placement constraints; *a combination of both*, where the tree topology and the sequence alignment are simultaneously required; and *quantitative properties*, where phylogenetic trees obtained from common inference methods (NJ, MLE or Bayesian) are analyzed under structural metrics. Table 1 summarizes relevant phylogenetic properties according to the classification introduced.

3.1 Tree Properties

Many typical tree properties are of a cladistic nature. 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 whose leaves form a clade? Formally, the PTL formula over the Kripke structure will be true if there exists 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.

$$clade\,(S) \equiv \mathbf{EF}\,(in\,(S) \wedge out\,(S)) \tag{3}$$

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.

$$in\,(S) \equiv \textstyle\bigwedge_{s\in S} \mathbf{EF\,AG}\,(s) \tag{4}$$

$$out\,(S) \equiv \mathbf{AF\,AG}\left(\textstyle\bigvee_{s\in S} s\right) \tag{5}$$

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).

$$CA(S) \equiv \neg terminal \wedge in(S) \tag{6}$$

Moreover, if the clade property is extended to both ancestral and leaf sequences, the structure of the formula remains unaltered, and 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.

$$in'\,(S) \equiv \textstyle\bigwedge_{s\in S} \mathbf{EF}\,(s) \tag{7}$$

$$out'\,(S) \equiv \mathbf{AG}\left(\textstyle\bigvee_{s\in S} s\right) \tag{8}$$

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: symbolic representation for manipulating sets, and boolean functions.

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 identifiable by characteristic polymorphisms; thus, they define genetic populations and usually mark these geographically as well. Their study is focused on the non-recombining regions of the genome, in particular mitochondrial DNA, where the original cladistic notation for haplogroups originated [20].

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.

$$classifier\,(S_1, S_2, \ldots, S_h) \equiv \textstyle\bigwedge_{i=1}^{h} haplogroup\,(S_i) \tag{9}$$

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.

$$haplogroup'\,(S) \equiv clade\,(S \cup children\,(S)) \tag{10}$$

However, it is often interesting to allow for flexibility in haplogroup placement in the ongoing study and refinement of coarse haplogroups and the exploration of alternative hypotheses. Whereas (10) 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).

$$haplogroup\,(S,h) \equiv \mathbf{EF}\,(in\,(S) \wedge nested\,(S,h)) \quad (11)$$

$$nested\,(S,h) \equiv \mathbf{A}\,[h\,\mathbf{U}\,out\,(S) \vee nesting\,(S)] \quad (12)$$

$$nesting\,(S) \equiv \mathbf{AF}\,\mathbf{AG}\,\left(\neg \bigvee\nolimits_{s\in S} s\right) \quad (13)$$

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), which is a restriction of the common ancestor set:

$$MRCA(S) \equiv CA(S) \wedge \neg\mathbf{EX}(CA(S)) \quad (14)$$

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.

$$clade(S) \equiv MRCA(S) \wedge out'\,(S) \quad (15)$$

In addition, a polyphyletic group is a "group whose members' last common ancestor is not a member of the group" [21]. The set nodes of the phylogenetic tree that validate the following formula form a polyphyletic group.

$$polyphyletic(S) \equiv MRCA(S) \notin S \quad (16)$$

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" [21]. 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:

$$paraphyletic(S,S_1,\ldots,S_h) \equiv MRCA(S)$$
$$\wedge\, disjoint(S,S_1,\ldots,S_h) \wedge \mathbf{AG}(\textstyle\bigvee_{i=1}^{h} clade(S_i))$$

Further topological-related properties of phylogenetic trees are described in [21]. In particular, synapomorphies, symplesiomorphies and autopomorphies focus on the length of 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.

A taxonomy of important biological properties is summarized in Table 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 [22]. 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 evolution divergence of a set of species.

3.2 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.

A first group of 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

Type	Name	Ref	PTL Formula
T	CA	Sect. 3.1	$CA(S) \equiv \neg terminal \wedge in(S)$
T	MRCA	Sect. 3.1	$MRCA(S) \equiv CA(S) \wedge \neg \mathbf{EX}(CA(S))$
T	Monophyly	[21]	$clade(S) \equiv MRCA(S) \wedge out'(S)$
T	Polyphyly	[21]	$polyphyletic(S) \equiv MRCA(S) \notin S$
T	Paraphyly	[21]	$paraphyletic(S, S_1, \ldots, S_h) \equiv MRCA(S) \wedge \mathbf{AG}(\bigvee_{i=1}^{h} clade(S_i)) \wedge disjoint(S, S_1, \ldots, S_h)$
T	Homology	[21]	$homology(S, col) \equiv \exists \sigma \in \Sigma,\ \forall s \in S\,(seq_s[col] = \sigma) \wedge \left(seq_{MRCA(S)}[col] = \sigma\right)$
T	Homoplasy	[21]	$homoplasy(S, col) \equiv \neg homology(S, col)$
T	Parallel Evolution	[21]	$parallel(S, col) \equiv homoplasy(S, col) \wedge distance(S) \leq threshold$
T	Convergent Evolution	[21]	$convergent(S, col) \equiv homoplasy(S, col) \wedge distance(S) > threshold$
T	Apomorphy	[21]	$apomorphy(S, col) \equiv \exists \sigma \in \Sigma,\ \forall s \in S\,(seq_s[col] = \sigma) \wedge (seq_{\mathbf{X}^{-1}s}[col] \neq \sigma)$
T	Plesiomorphy	[21]	$plesiomorphy(S, col) \equiv \neg apomorphy(S, col)$
T	Synapomorphy	[21]	$synapomorphy(S, col) \equiv homology(S, col) \wedge \left(seq_{\mathbf{X}^{-1}MRCA(S)}[col] \neq \sigma\right)$
T	Symplesiomorphy	[21]	$symplesiomorphy(S, col) \equiv homology(S, col) \wedge \neg synapomorphy(S, col)$
T	Autapomorphy	[21]	$autapomorphy(s, col) \equiv \exists \sigma \in \Sigma,\ (seq_s[col] = \sigma) \wedge (seq_{\mathbf{X}^{-1}s}[col] \neq \sigma) \wedge terminal(s)$
S	Covariation	[12]	$covariation(i, j) \equiv \mathbf{EF}\,\mathbf{AG}\,(seq[i] = \sigma_i \rightarrow seq[j] = \sigma_j)$
S	Conservation	[12]	$conservation(i, j) \equiv \mathbf{EF}\,\mathbf{AG}\,(seq[i, \ldots, j] = \sigma_i \ldots \sigma_j)$
T & S	Back Mutation	Sect. 3.4	$detectBM \equiv \forall j \in \{1, l\},\ \mathbf{AG}\,(\neg hasBM\,(j))$; $hasBM\,(col) \equiv \exists \sigma \in \Sigma,\ (seq[col] = \sigma) \wedge \mathbf{EF}\,(seq[col] \neq \sigma \wedge \mathbf{EF}\,(seq[col] = \sigma))$
T & S	Multiple Nucleotide Polymorphism	Sect. 3.4	$detectMNP \equiv \forall j \in \{1, l-1\},\ \mathbf{AG}\,(\neg startsMNP\,(j))$; $startsMNP\,(col) \equiv \exists \sigma, \tau, \upsilon \in \Sigma,\ (seq[col-1] = \sigma) \wedge (seq[col] = \tau) \wedge (seq[col+1] = \upsilon) \wedge \mathbf{EX}[(seq[col-1] \neq \sigma) \wedge (seq[col] \neq \tau) \wedge (seq[col+1] \neq \upsilon)]$
Q	Distance	3.4	$distance(d, event) \equiv \mathbf{AF}^{\leq d} event$

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

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 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.

$$global\,(p) \equiv \mathbf{AG}\,(p) \tag{17}$$

Exceptions to the aforementioned properties may in fact indicate suspicious or potentially deleterious mutations, which are of great interest in applied phylogenetic studies [23]. 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 ($\mathbf{AX}^k$).

$$terminal\,(p) \equiv \mathbf{AG}\,(p \rightarrow leaf) \tag{18}$$

$$terminal\,(p, k) \equiv \mathbf{AG}\left(p \rightarrow \mathbf{AX}^k\,(leaf)\right) \tag{19}$$

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.

$$leaf \equiv \textstyle\bigwedge_{p \in AP} p \leftrightarrow \mathbf{AX}\,(p) \tag{20}$$

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, though their study is beyond the scope of this paper. Notice that pattern-based checking of haplogroup classifications falls within this category.

3.3 A Combination of Both

Some properties do not fit exclusively into one of the previous classification groups but are a mix of 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 $seq[i] = \sigma$ ($seq[i] \neq \sigma$) signifies that $\sigma \in \Sigma$ appears (not) in position i in a state sequence (recall the AP definition in Def. 3).

For legibility 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).

$$detectBM \equiv \bigwedge_{j=1}^{l} \mathbf{AG}\left(\neg hasBM\left(j\right)\right) \quad (21)$$

$$hasBM\left(col\right) \equiv \bigvee_{\sigma \in \Sigma} \sigma_{col} \wedge \mathbf{EF}\left(\overline{\sigma}_{col} \wedge \mathbf{EF}\left(\sigma_{col}\right)\right) \quad (22)$$

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. 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 (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).

$$detectMNP \equiv \bigwedge_{j=1}^{l-1} \mathbf{AG}\left(\neg startsMNP\left(j\right)\right) \quad (23)$$

$$startsMNP\left(col\right) \equiv \bigvee_{\sigma,\tau,\upsilon \in \Sigma} \left(\sigma_{col-1} \wedge \tau_{col} \wedge \upsilon_{col+1} \wedge \mathbf{EX}\left(\overline{\sigma}_{col-1} \wedge \overline{\tau}_{col} \wedge \overline{\upsilon}_{col+1}\right)\right) \quad (24)$$

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 Eq. 11–13 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 haplogroup fitness of each candidate and select the ancestor with the maximum group of descendants. Other related biological properties would have different maximization (minimization) purposes.

3.4 Quantitative Properties

The result of some phylogenetic properties might not necessarily be constrained to a boolean result. As well as checking the validity of a formula with *qualitative* model checking, biologists often need quantitative information such as the number of species in which two positions covariate, the number of mutations along a tree path or the percentage of conservation of a sequence fragment in the phylogenetic tree. We may even wish to measure the fitness of a set of trees according to different parameters. The ratio of descendants per taxon (tree balance) and the analysis of branch lengths help to study, compare and refine the tree structure [1], [33], [34]. These supplementary metrics are of particular interest in the case of several phylogenies sharing identical scores under the same inference method [35]. Quantitative, parametric and stochastic temporal logics contribute to the definition and verification of such properties.

Real-time [36] and quantitative temporal logics [37] cater for the expression of quantitative bounds such as the maximum or minimum number of steps between events, considered as sets of states characterized by a logical formula. Such logics allow operators of the form $\mathbf{AF}^{\leq 5}p$ meaning that inevitably event p will occur within five time steps. This feature provides an explicit way for the computation of branch lengths and the definition of mutation and evolution clocks. Distance-based inference methods can also take advantage of this particularity for the construction of a phylogenetic tree [4]. An extension of quantitative

Name	Main feature	Property Language	Platform
NuSMV [24]	OpenSource	LTL, CTL, PSL	Win, Unix, MacOS
PROD [25]	On-the-fly Model Checker	CTL	Linux
SPIN [26]	Generic Model Checker	LTL	Win, Unix related
DiViNe Tool [27]	Distributed Multi-Core Model Checker	LTL	Unix
Eddy Murphi [28]	Distributed Multi-Core Model Checker	Assertions	Unix
PVeSta [29]	Statistical Parallel and Multi-Core Model Checker	PCTL	Win, Linux, MacOS
PRISM [30]	Probabilistic and Quantitative Logics	PCTL, CSL^{TA}, LTL, PCTL*	Win, Linux, MacOS
UPPAAL [31]	Commercial Software for Real-Time Systems	TCTL	Win, Linux
TLQSolver [32]	Temporal Logic Query Checker for Mining Properties	Query CTL	Linux

TABLE 2
List of some available Model Checkers.

model checking where integer values are replaced by parametric variables is detailed in [38].

Next, Bayesian model checking methods allow for analyzing stochastic systems [39]. Probabilistic CTL (PCTL) [14], [40] and CSL^{TA} [41] help to formulate conditions on a state of a Markov chain. PCTL allows enriched queries of temporal formulas such as $\mathbb{P}_J(\phi)$. Given an initial state s and a bound $J \in [0,1]$, $\mathbb{P}_J(\phi)$ returns the probability for a set of paths satisfying ϕ. Mutations seem to act in this stochastic way. Thus, models of DNA substitution such as Jukes-Cantor, Kimura or GTR can be tested with this logic [42] (Section 5.2). The definition of transitions and transversions, as well as the branching (mutation) rate J must be explicitly provided by the user. Further questions such as the cumulative probability of reaching a specific leaf can be queried by the biologist using these logics over trees modeled as Markov chains.

Occasionally, biologists do not want to verify a specific hypothesis but let the computer automatically discover new ones. 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. Some theoretical approaches for CTL bridge the transition between verification and mining [43], [44]. Basically, they define a kind of CTL query with a hole, 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.

4 Implementation Issues

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.

4.1 Model Checking Tools

A consideration of the different types of temporal logics involves the implementation of a pool of model checker packages for supporting them (see Table 2). 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" [30]. In addition, PRISM has been successfully applied in a nearby application domain such as biological pathway regulation. It seems suitable for 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" [31]. This feature can be exploited in biology so as to provide an explicit definition of evolution clocks. Furthermore, NuSMV [24] includes quantitative extensions related with the computation of paths with maximum (minimum) lengths that satisfy a property. It also includes a simple database manager for reusing the results of previous properties. In addition, the classic open-source model checker SPIN offers a framework that can be tuned as desired [26]. Sliced model checking can be coded over these two packages [45], [46].

In conjunction with the previous logics, on-the-fly model checking improves performance by exploring the state space on demand [47]. One of the best model checkers in this field is PROD [25]. Moreover, multicore and network-oriented tools such as DiViNe [27] (for LTL), Murphi [28] or PVeStA [29] (stochastic) are available in order to speed up efficiency.

Finally, TLQSolver [32] is a model checker that manages CTL queries and allows the mining of properties that match a specific pattern. The definition of patterns helps to select sets of nodes of the phylogenetic tree satisfying an abstract relation in a similar way to a SQL query. These sets can then be postprocessed in order to extract the differences among them.

4.2 Codification of Branching-time Phylogenies

We have used NuSMV, a well-known model checking software tool [24], for the implementation of the branching-time phylogenetic tree [13]. A description

of the Kripke structure and the atomic propositions in NuSMV syntax must be provided by the user as the input for the model checker. To this end, we preprocess the sequence alignment and the phylogenetic tree. Translation from the phylogenetic tree to the SMV Kripke structure syntax has been performed automatically by a Bioperl script [48]. The inclusion of arrays, sets and macros in the NuSMV syntax facilitates a compact characterization of DNA and protein sequences as strings of characters.

Fig. 3 shows the implementation of the branching-time phylogenetic tree from Fig. 1 in SMV code. The main module describes the topology of the evolutionary tree, where the names of the tree nodes (taxa) are defined as $N1, \ldots, N5$, and variables label the states. 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 the function returning the DNA string associated to each node.

Usually, model checking packages store the previous description of the transition system in a BDD structure [16]. Internally, each state is composed of a set of boolean variables that are true for the DNA sequence that it represents. This feature provides an easy scheme for the manipulation of sets in a symbolic way because a DNA 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 the next section, we verify phylogenetic properties using the description in Fig. 3 as input data.

	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 3
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 4
Megabytes needed for the creation of the Kripke structure and the storage of protein sequences.

4.3 Performance Results

The performance evaluation of our system has been measured with human protein alignments retrieved from GenBank [49]. 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, O.S. Debian Linux. Notice that 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 3 and Table 4 show time and memory consumption with respect to sequence length and alignment size.

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. 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 (Eq. 21) 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 and 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 consumes 19.2 megabytes and 4 seconds in NuSMV.

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 sequence length will be the main bottleneck if we straightforwardly apply our

```
MODULE main
VAR n: {N1,N2,N3,N4,N5};          /* States (taxa) of the phylogenetic tree */
    dna: process dna_sequence(n); /* DNA Labeling function */
ASSIGN
  init(n) := N1;                /* Tree root */
  next(n) := case               /* Definition of the successors of each node */
         n=N1: {N2, N3};        /* Nodes N2 and N3 are the successors of N1 */
         n=N2: {N4, N5};        /* Nodes N4 and N5 are the successors of N2 */
         TRUE: n;               /* Self-loops in leaf nodes */
      esac;
MODULE dna_sequence(n)
/* Definition of the array of characters and the DNA alphabet */
VAR sequence: array 1..3 of {A,C,G,T};
ASSIGN
  init(sequence[1]):= A;     init(sequence[2]):= A;     init(sequence[3]):= G;
  next(sequence[1]):= case
         n=N2: G;    n=N3: A;    n=N4: A;    n=N5: A;    TRUE: A;
  esac; ...
```

Fig. 3. Mapping of the phylogenetic tree of Fig. 1 in SMV.

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 [46], and sliced [45] or distributed model checking [27], [28], [29] 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.

5 Future Work

Finally, the analysis of phylogenetic networks and the evaluation of quantitative and stochastic properties in DNA mutation models will be the subject of our future work. We believe that model checking will be competitive in this field for the analysis of cycles, the computation of the maximum likelihood estimations or distances between clusters of nodes. Model checking techniques support the analysis and verification of different systems in computational biology.

5.1 Phylogenetic Networks

As well as phylogenetic trees, other structures such as phylogenetic networks are suitable for phylogenetic analysis as they represent a generalization over trees [50]. Principally, 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, rootedness is an important factor in practice and should not be taken for granted, even for phylogenetic trees. To this end, model checking is useful as a method to locate the root. Given a set of properties, model checking will consist of detecting the set of potential roots that verify them. 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 philosophy underlying this approach is similar to bootstrapping, where the validity of a set of properties over the tree is chosen as a fitness parameter for the root under consideration.

Another useful idea would be the inclusion of cycles in the phylogenetic network. This involves the addition of new semantic information to the phylogeny and the matching of multiple phylogenetic interpretations. For example, cycles in 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. This feature is 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.

5.2 Models of DNA Evolution

The evolution drift leading the substitution of nucleotides in the DNA is usually modeled with a Markov chain. 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 [42]. The existence of adapted tools, data structures and probabilistic temporal logics that capture the underlying randomness of these systems allows for inspecting stochastic properties in probabilistic timed automates. Thus, our framework based on formal methods techniques will also accept the characterization of this kind of DNA models as input. The analysis of these models opens a new perspective that extends the work presented in this paper to other phylogenetic fields.

Complementary, 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. Hence, 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. Therefore, a phylogenetic tree, implicitly represented as a succession of transitions and transversions by a set of temporal logic properties, can be checked over a particular DNA model.

The verification of the phylogenetic tree will return the agreement with respect to a DNA evolution model. This can be done with probabilistic temporal logics in order to obtain a confidence value between $[0, 1]$ for the tree under consideration. 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 [51, Fig. 2]. The script presented in Section 4.2 accepts extensions for generating multiple bootstraps that feed the iterative refinement process.

Model checkers such as PRISM are suitable for the computation of maximum likelihood in stochastic systems. Besides, PRISM is implemented in Java, which ensures portability, but penalizes the performance in large computations. Future optimizations should be focused on the implementation of more efficient libraries and dynamic programming techniques. The translation of the MLE equations to the syntax of the stochastic logic supported by PRISM, and the performance tests are omitted due to space constraints.

To sum up, model checking can be placed as an intermediate step in the iterative refinement process of phylogenetic inference. The main advantages of our approach are still valid for this new situation: generality, modularity, independence of the model (DNA mutations) from the specifications (phylogenetic tree) and the availability of powerful model checking tools.

6 Conclusions

The aim of this paper 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. Temporal properties inquire about logical or numerical relations in the tree.
- The use of standard model checking techniques 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 of the states in the Kripke structure allows for the evaluation of phylogenetic properties and the symbolic manipulation of sets using a suitable temporal logic.

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) enlarge the expressiveness of biological properties. It also ensures a substantial collection of tools that can be adapted for any specific study of phylogenetic analysis.

Complementary, the definition of Kripke structures and almost every type of temporal logic support the generalization of a phylogenetic tree as a phylogenetic network with multiple roots, cycles and horizontal transferences. Models of DNA substitution defined by more powerful data structures such as Markov chains are also supported by specialized model checking tools, which allow for a quantitative and probabilistic analysis of the phylogenies.

In the second part of this paper, 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 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 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 could become the major bottleneck in the near future. Thus, scaling the model checking verification process both in time and memory will be 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.

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.

We can conclude that our first approach to phylogenetic analysis using model checking is innovative and encouraging in terms of methodology and efficiency. Although this framework cannot yet be applied directly to bigger systems, it is at least a powerful and competitive approach for the analysis of mtDNA. The optimization of our framework and the evaluation of qualitative, quantitative and probabilistic properties in larger phylogenies as well as integration with statistical problems will be the subject of future work.

ACKNOWLEDGMENTS

The authors are indebted to five anonymous reviewers and the fruitful discussions with J. Álvarez-Jarreta and E. Ruiz-Pesini. This work was supported by the Spanish Ministry of Science and Innovation (MICINN) [TIN2011-27479-C04-01], the Spanish Ministry of Education [AP2008-03447], the Government of Aragon [B117/10] and the GISED research group.

REFERENCES

[1] A. O. Mooers and S. B. Heard, "Inferring evolutionary process from phylogenetic tree shape," *Quarterly Review of Biology*, pp. 31–54, 1997.

[2] E. S. Allman and J. A. Rhodes, "Trees, fast and accurate." *Science*, vol. 327, no. 5971, pp. 1334–1335, 2010.

[3] J. Felsenstein, *Inferring phylogenies*. Sunderland, MA: Sinauer, 2003.

[4] Z. Yang and B. Rannala, "Molecular phylogenetics: principles and practice," *Nature Reviews Genetics*, vol. 13, no. 5, pp. 303–314, 2012.

[5] W. M. Fitch, "Uses for Evolutionary Trees," *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, vol. 349, no. 1327, pp. 93–102, 1995.

[6] O. Grumberg *et al.*, Eds., *25 years of model checking: history, achievements, perspectives*. Berlin: Springer, 2008.

[7] J. Barnat, L. Brim, A. A. Krejci, A. Streck, D. Safranek, M. Vejnar, and T. Vejpustek, "On Parameter Synthesis by Parallel Model Checking," *IEEE/ACM Trans. on Computational Biology and Bioinformatics*, vol. 9, no. 3, pp. 693–705, 2012.

[8] P. T. Monteiro, D. Ropers, R. Mateescu, A. T. Freitas, and H. Jong, "Temporal logic patterns for querying dynamic models of cellular interaction networks," *Bioinformatics*, vol. 24, pp. i227–i233, 2008.

[9] A. Rizk, G. Batt, F. Fages, and S. Soliman, "On a continuous degree of satisfaction of temporal logic formulae with applications to systems biology," in *Procs. Computational Methods in Systems Biology*, Rostock, 2008, pp. 251–268.

[10] C. J. Langmead and S. K. Jha, "Predicting protein folding kinetics via temporal logic model checking," in *Procs. 7th Workshop on Algorithms in Bioinformatics*, Philadelphia, PA, 2007, pp. 252–264.

[11] E. De Maria, F. Fages, A. Rizk, and S. 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*, vol. 412, no. 21, pp. 2108–2127, 2011.

[12] R. Blanco, G. de Miguel Casado, J. I. Requeno, and J. M. Colom, "Temporal logics for phylogenetic analysis via model checking," in *Procs. IEEE Int. Workshop on Mining and Management of Biological and Health Data*, Los Alamitos, CA, 2010.

[13] J. I. Requeno, R. Blanco, G. de Miguel Casado, and J. M. Colom, "Phylogenetic Analysis Using an SMV Tool," in *5th Int. Conf. on Practical Applications of Computational Biology & Bioinformatics*, ser. Advances in Intelligent and Soft Computing, M. Rocha *et al.*, Eds. Berlin: Springer, 2011, vol. 93, pp. 167–174.

[14] C. Baier and J.-P. Katoen, *Principles of model checking*. Cambridge, MA: The MIT Press, 2008.

[15] E. M. Clarke and E. A. Emerson, "Design and synthesis of synchronization skeletons using branching time temporal logic," in *Procs. Workshop on Logics of Programs*, Yorktown Heights, NY, 1981, pp. 52–71.

[16] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE T. Comput.*, vol. C-35, pp. 677–691, 1986.

[17] E. Paradis, *Analysis of Phylogenetics and Evolution with R (Use R!)*, R. Gentleman *et al.*, Eds. Berlin: Springer, 2012.

[18] W. P. Maddison and D. Maddison., "Mesquite: a modular system for evolutionary analysis. Version 2.75," 2011.

[19] R. Knight *et al.*, "PyCogent: a toolkit for making sense from sequence," *Genome Biology*, vol. 8, pp. 1–16, 2007.

[20] M. B. Richards, V. A. Macaulay, H.-J. Bandelt, and B. C. Sykes, "Phylogeography of mitochondrial DNA in western Europe," *Ann. Hum. Genet.*, vol. 62, pp. 241–260, 1998.

[21] E. O. Wiley, *Phylogenetics : the theory and practice of phylogenetic systematics*. New York: Wiley, 1981.

[22] O. Lichtenstein, A. Pnueli, and L. D. Zuck, "The Glory of the Past," in *Procs. Conf. on Logic of Programs*. London, UK: Springer, 1985, pp. 196–218.

[23] J. Montoya, E. López Gallardo, C. Díez Sánchez, M. J. López Pérez, and E. Ruiz Pesini, "20 years of human mtDNA pathologic point mutations: carefully reading the pathogenicity criteria," *Biochim. Biophys. Acta*, vol. 1787, pp. 476–483, 2009.

[24] A. Cimatti *et al.*, "NuSMV 2: An OpenSource Tool for Symbolic Model Checking," in *Computer Aided Verification*, ser. LNCS, E. Brinksma *et al.*, Eds. Berlin: Springer, 2002, vol. 2404, pp. 241–268.

[25] K. Varpaaniemi, J. Halme, K. Hiekkanen, and T. Pyssysalo, "PROD reference manual," Helsinki University of Technology, Digital Systems Laboratory, Espoo, Finland, Tech. Rep., 1995.

[26] G. J. Holzmann, "The Model Checker SPIN," *IEEE Trans. on Software Engineering*, vol. 23, pp. 279–295, 1997.

[27] J. Barnat, L. Brim, M. Ceska, and P. Rockai, "DiVinE: Parallel Distributed Model Checker," *9th Int. Workshop on Parallel and Distributed Methods in Verification and 2nd Int. Workshop on High Performance Computational Systems Biology*, pp. 4–7, 2010.

[28] I. Melatti *et al.*, "Parallel and distributed model checking in Eddy," *International Journal on Software Tools for Technology Transfer*, vol. 11, no. 1, pp. 13–25, 2009, springer.
[29] M. AlTurki and J. Meseguer, "PVeStA: A Parallel Statistical Model Checking and Quantitative Analysis Tool," in *Algebra and Coalgebra in Computer Science*, ser. LNCS, A. Corradini *et al.*, Eds. Berlin: Springer, 2011, vol. 6859, pp. 386–392.
[30] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of Probabilistic Real-time Systems," in *Computer Aided Verification*, ser. LNCS, G. Gopalakrishnan *et al.*, Eds. Berlin: Springer, 2011, vol. 6806, pp. 585–591.
[31] G. Behrmann, A. David, and K. Larsen, "A Tutorial on Uppaal," in *Formal Methods for the Design of Real-Time Systems*, ser. LNCS, M. Bernardo *et al.*, Eds. Springer, 2004, vol. 3185, pp. 33–35.
[32] M. Chechik and A. Gurfinkel, "TLQSolver: A Temporal Logic Query Checker," in *Computer Aided Verification*, ser. LNCS, W. A. Hunt Jr. *et al.*, Eds. Springer, 2003, vol. 2725, pp. 210–214.
[33] D. Robinson and L. R. Foulds, "Comparison of phylogenetic trees," *Mathematical Biosciences*, vol. 53, no. 1, pp. 131–147, 1981.
[34] M. Stich and S. Manrubia, "Topological properties of phylogenetic trees in evolutionary models," *The European Physical Journal B*, vol. 70, no. 4, pp. 583–592, 2009.
[35] S.-J. Sul, S. Matthews, and T. L. Williams, "Using tree diversity to compare phylogenetic heuristics," *BMC bioinformatics*, vol. 10, no. Suppl 4, p. S3, 2009.
[36] R. Alur and T. Henzinger, "Logics and models of real time: A survey," in *Real-Time: Theory in Practice*, ser. LNCS, J. de Bakker *et al.*, Eds. Berlin: Springer, 1992, vol. 600, pp. 74–106.
[37] S. Campos, E. Clarke, W. Marrero, M. Minea, and H. Hiraishi, "Computing quantitative characteristics of finite-state real-time systems," in *Procs. Real-Time Systems Symposium.*, 1994, pp. 266–270.
[38] E. Emerson and R. Trefler, "Parametric quantitative temporal reasoning," in *Procs. 14th IEEE Symposium on Logic in Computer Science*, 1999, pp. 336 –343.
[39] S. K. Jha, E. M. Clarke, C. J. Langmead, A. Legay, A. Platzer, and P. Zuliani, "A bayesian approach to model checking biological systems," in *Computational Methods in Systems Biology*. Springer, 2009, pp. 218–234.
[40] R. Segala and N. Lynch, "Probabilistic simulations for probabilistic processes," in *Int. Conf. on Concurrency Theory*, ser. LNCS, B. Jonsson *et al.*, Eds. Berlin: Springer, 1994, vol. 836, pp. 481–496.
[41] S. Donatelli, S. Haddad, and J. Sproston, "Model Checking Timed and Stochastic Properties with CSL^{ta}," *IEEE Trans. on Software Engineering*, vol. 35, no. 2, pp. 224–240, 2009.
[42] Z. Yang, *Computational Molecular Evolution*, P. Harvey and R. M. May, Eds. Oxford University Press, 2006.
[43] W. Chan, "Temporal-logic Queries," in *Computer Aided Verification*, ser. LNCS, E. Emerson *et al.*, Eds. Berlin: Springer, 2000, vol. 1855, pp. 450–463.
[44] A. Gurfinkel, M. Chechik, and B. Devereux, "Temporal Logic Query Checking: A Tool for Model Exploration," *IEEE Trans. on Software Engineering*, vol. 29, p. 2003, 2003.
[45] J. I. Requeno, R. Blanco, G. de Miguel Casado, and J. M. Colom, "Sliced Model Checking for Phylogenetic Analysis," in *6th Int. Conf. on Practical Applications of Computational Biology & Bioinformatics*, ser. Advances in Intelligent and Soft Computing, M. Rocha *et al.*, Eds. Springer, 2012, vol. 154, pp. 95–103.
[46] J. I. Requeno and J. M. Colom, "Speeding Up Phylogenetic Model Checking," in *7th Int. Conf. on Practical Applications of Computational Biology & Bioinformatics*, ser. Advances in Intelligent and Soft Computing, M. S. Mohamad *et al.*, Eds. Spring, 2013, vol. 222, pp. 119–126.
[47] G. Bhat, R. Cleaveland, and O. Grumberg, "Efficient on-the-fly model checking for CTL," in *Procs. 10th IEEE Symposium on Logic in Computer Science*, 1995, pp. 388 –397.
[48] J. E. Stajich *et al.*, "The Bioperl Toolkit: Perl Modules for the Life Sciences," *Genome Research*, vol. 12, no. 10, pp. 1611–1618, 2002.
[49] D. A. Benson *et al.*, "GenBank," *Nucleic Acids Research*, vol. 40, no. D1, pp. D48–D53, 2012.
[50] D. H. Huson *et al.*, *Phylogenetic Networks: Concepts, Algorithms and Applications*, C. U. Press, Ed., 2011.
[51] A. P. Stamatakis, T. Ludwig, and H. Meier, "The AxML program family for maximum likelihood-based phylogenetic tree inference," *Concurrency and Computation: Practice and Experience*, vol. 16, no. 9, pp. 975–988, 2004.

José Ignacio Requeno is currently a Ph.D. student in the Department of Computer Science and Systems Engineering, University of Zaragoza. He received his MS degree in Computer Science Engineering in 2009 from the University of Zaragoza, Spain. He is also a member of the Group of Discrete Event Systems Engineering (GISED). His current research focuses on computational biology and the application of formal methods to bioinformatics.

Gregorio de Miguel Casado received a BS degree in Computer Science Engineering (2001), a Master degree in Business Administration (2003) and a Ph.D (2010) from the University of Alicante. Since 2007 he has been Assistant Professor in the Department of Computer Science and Systems Engineering, University of Zaragoza. He is also a member of the Group of Discrete Event Systems Engineering (GISED) as well as the Aragon Institute for Engineering Research(I3A). At present, his research interests are computable analysis and computer arithmetics in scientific computing, computational biology and general artificial intelligence.

Roberto Blanco is currently a Ph.D. student in the Department of Computer Science and Systems Engineering, University of Zaragoza. He received his MS degree in Computer Science Engineering in 2008 from the University of Zaragoza, Spain. He has published several articles related to the scalability and reconstruction of phylogenies.

José Manuel Colom received a Ph.D. degree in Industrial-electrical Engineering from the University of Zaragoza, Zaragoza, Spain, in 1989. He is currently a Professor in the Department of Computer Science and Systems Engineering, University of Zaragoza. He has co-authored over 100 research papers in technical journals and conferences. His research interests include modeling, qualitative, and performance analysis, and implementation of discrete event systems using Petri nets. Dr. Colom has served on the technical committees of several international conferences in the field of formal methods and concurrent systems. He has organized, as a Conference Co-Chair, several international conferences in the field of Petri nets and discrete event systems.