J Autom Reasoning (2007) 39:471–511
DOI 10.1007/s10817-007-9082-1

Inferring Phylogenetic Trees Using Answer Set Programming

Daniel R. Brooks · Esra Erdem · Selim T. Erdoğan · James W. Minett · Don Ringe

Received: 1 January 2006 / Accepted: 1 April 2007 / Published online: 17 July 2007

Abstract We describe the reconstruction of a phylogeny for a set of taxa, with a character-based cladistics approach, in a declarative knowledge representation formalism, and show how to use computational methods of answer set programming to generate conjectures about the evolution of the given taxa. We have applied this computational method in two domains: historical analysis of languages and historical analysis of parasite-host systems. In particular, using this method, we have computed some plausible phylogenies for Chinese dialects, for Indo-European language groups, and for *Alcataenia* species. Some of these plausible phylogenies are different from the ones computed by other software. Using this method, we can easily describe domain-specific information (e.g., temporal and geographical constraints), and thus prevent the reconstruction of some phylogenies that are not plausible.

Keywords Answer set programming · Phylogeny · Cladistics

This paper is a revised and extended version of [3].

D. R. Brooks
Department of Ecology and Evolutionary Biology, University of Toronto,
Toronto, Ontario, Canada

E. Erdem (✉)
Faculty of Engineering and Natural Sciences, Sabancı University, Istanbul, Turkey
e-mail: esraerdem@sabanciuniv.edu

S. T. Erdoğan
Department of Computer Sciences, University of Texas at Austin, Austin, TX, USA

J. W. Minett
Department of Electronic Engineering, Chinese University of Hong Kong, Shatin, Hong Kong

D. Ringe
Department of Linguistics, University of Pennsylvania, Philadelphia, PA, USA

1 Introduction

Cladistics (or phylogenetic systematics), developed by Willi Hennig [23–25], is the study of evolutionary relations between species based on their shared traits. Represented diagrammatically, these relations can form a tree whose leaves represent the species, internal vertices represent their ancestors, and edges represent the genetic relationships between them. Such a tree is called a "phylogenetic tree" (or a "phylogeny"). In this paper, we study the problem of reconstructing phylogenies for a set of taxa (taxonomic units) with a character-based cladistics approach.[1]

In character-based cladistics, each taxonomic unit is described with a set of "(qualitative) characters" – traits that every taxonomic unit can instantiate in a variety of ways. The taxonomic units that instantiate the character in the same way are assigned the same "state" of that character. Here is an example from [38]. Consider the languages English, German, French, Spanish, Italian, and Russian. A character for these languages is the basic meaning of "hand":

English	German	French	Spanish	Italian	Russian
hand	*Hand*	*main*	*mano*	*mano*	*ruká*

Since the English and German words descended from the same word in their parent language, namely Proto-Germanic **handuz*, by direct linguistic inheritance, those languages must be assigned the same state for this character. The three Romance languages must likewise be assigned a second state (since their words are all descendants of Latin *manus*), and Russian must be assigned a third:

English	German	French	Spanish	Italian	Russian
1	1	2	2	2	3

In character-based cladistics, after describing each taxonomic unit with a set of characters, and determining the character states, the phylogenies are reconstructed by analyzing the character states. There are two main approaches: one is based on the "maximum parsimony" criterion [10], and the other is based on the "maximum compatibility" criterion [6]. According to the former, the goal is to infer a phylogeny with the minimum number of character state changes along the edges. With the latter approach, the goal is to reconstruct a phylogeny with the maximum number of "compatible" characters. Intuitively, a character is compatible if it evolves without backmutation (i.e., it does not evolve from one state to another and then back to the earlier state) or parallel evolution (i.e., if no state appears independently in different lines of descent). Both problems are NP-hard [8, 20]. In this paper we present a method for reconstructing a phylogenetic tree for a set of taxa, with the latter approach.

Our method is based on the programming methodology called answer set programming (ASP) [28, 33, 40]. It provides a declarative representation of the problem as a logic program whose answer sets [21, 22] correspond to solutions. The answer sets for the given formalism can be computed by special systems called answer set

[1] See [15] for a survey on the other methods for phylogeny reconstruction.

solvers. For instance, CMODELS,[2] SMODELS,[3] and DLV[4] are some of the answer set solvers that are currently available. Answer set solvers are similar to SAT solvers in terms of speed, but they have a more expressive input language. In particular, they allow recursive definitions, which may be used to describe trees – crucial to the work presented here.

We apply our method of reconstructing phylogenies using ASP to historical analysis of languages and to historical analysis of parasite-host systems.

Histories of individual languages give us information from which we can infer principles of language change. This information is of interest not only to historical linguists but also to archaeologists, human geneticists, and physical anthropologists. For instance, an accurate reconstruction of the evolutionary history of certain languages can help us answer questions about human migrations, the time that certain artifacts were developed, when ancient people began to use horses in agriculture [31, 32, 39, 42].

Parasites occur worldwide, causing malnutrition, sickness, and even sometimes the death of their hosts. Historical analysis of parasites gives us information on where they come from and when they first started infecting their hosts. The phylogenies of parasites, with the phylogenies of their hosts and with the geographical distribution of their hosts, can be used to understand the changing dietary habits of a host species, to understand the structure and the history of ecosystems, and to identify the history of animal and human diseases. This information allows predictions about the age and duration of specific groups of animals of a particular region or period and the identification of regions of evolutionary "hot spots" [5] and thus can be useful to assess the importance of specific habitats, geographic regions, and biotas – all the plant and animal life of a particular region – and areas of critical genealogical and ecological diversity [4, 5]. Identification of the most vulnerable members of a community by this way allows us to make more reliable predictions about the impacts of perturbations (natural or caused by humans) on ecosystem structure and stability [4].

With this method, using the answer set solver CMODELS, we have computed 33 phylogenetic trees for seven Chinese dialects based on 15 lexical characters, and 45 phylogenetic trees for 24 Indo-European languages based on 248 lexical, 22 phonological, and 12 morphological characters. Some of these phylogenies are plausible from the viewpoint of historical linguistics. We have also computed 21 phylogenetic trees for nine species of *Alcataenia* (a tapeworm genus) based on their 15 morphological characters, some of which are plausible from the viewpoint of coevolution – the evolution of two or more interdependent species each adapting to changes in the other, and from the viewpoint of historical biogeography – the study of the geographic distribution of organisms.

We have also computed most parsimonious trees for these three sets of taxa, using PARS (available with PHYLIP [16]). Considering also the most parsimonious trees published in [37] (for Indo-European languages), [34] (for Chinese dialects), and [26, 27] (for *Alcataenia* species), we have observed that some of the plausible

[2] http://www.cs.utexas.edu/users/tag/cmodels.html.

[3] http://www.tcs.hut.fi/Software/smodels/.

[4] http://www.dbai.tuwien.ac.at/proj/dlv/.

trees we have computed using the compatibility criterion are different from the most parsimonious ones. This situation shows that the availability of our computational method based on maximum compatibility can be useful for generating conjectures that cannot be found by other computational tools based on maximum parsimony.

As for related work, one available software system that can compute phylogenies for a set of taxa based on the maximum compatibility criterion is CLIQUE (available with PHYLIP), which is applicable only to sets of taxa where a taxonomic unit is mapped to state 0 or state 1 for each character. This feature prevents us from using CLIQUE to reconstruct phylogenies for the three sets of taxa mentioned above because, in each set, some taxonomic unit is mapped to state 2 for some character. Another system is the Perfect Phylogeny software of [38], which can compute a phylogeny with the maximum number of compatible characters only when all characters are compatible. Otherwise, it computes an approximate solution. In this sense, our method is more general than the existing ones that compute trees based on maximum compatibility.

Another advantage of our method over the existing ones mentioned above is that we can easily include in the program domain-specific information (e.g., temporal and geographical constraints) and thus prevent the reconstruction of some trees that are not plausible.

We consider reconstruction of phylogenies as the first step of reconstructing the evolutionary history of a set of taxa. The idea is then to reconstruct (temporal) phylogenetic networks, which also explain the contacts (or borrowings) between taxonomic units, from the reconstructed phylogenies. The second step is studied in [12, 13, 36].

In the following, first we precisely describe the problem we address in this paper, that is, the *Maximum Compatibility Problem* (Section 2), and we define answer sets (Section 3). After describing the ASP programming methodology (Section 4), we formulate the problem as a logic program (Section 5) and address the correctness of the program (Section 6). After describing some useful heuristics to make the computation more efficient (Section 7), we discuss how we compute phylogenies in ASP and with respect to which assumptions, and how we evaluate the results (Section 8). Then we present the phylogenies reconstructed for Chinese dialects (Section 9), for Indo-European languages (Section 10), and for *Alcataenia* species (Section 11) and elaborate on their plausibility. We conclude with a comparison of our approach to related work (Section 12). Proofs of theorems are presented in the Appendix.

2 Problem Description

We use graphs to describe the problem of computing phylogenies. Therefore, we first introduce some definitions related to graphs.

A *directed graph (digraph)* is an ordered pair $\langle V, E\rangle$, where V is a set and E is a binary relation on V. In a digraph $\langle V, E\rangle$, the elements of V are called *vertices*, and the elements of E are called the *edges* of the digraph. The *out-degree* of a vertex v is the number of edges (v, u) $(u \in V)$, and the *in-degree* of v is the number of edges (u, v) $(u \in V)$. A digraph $\langle V', E'\rangle$ is a *subgraph* of a digraph $\langle V, E\rangle$ if $V' \subseteq V$ and $E' \subseteq E$.

In a digraph $\langle V, E\rangle$, a *path* from a vertex u to a vertex u' is a sequence $v_0, v_1, \ldots, v_k$ of vertices such that $u = v_0$ and $u' = v_k$, and $(v_{i-1}, v_i) \in E$ for $1 \leq i \leq k$. If there is a path from a vertex u to a vertex v, then we say that v is *reachable from* u. If V' is a subset of V, a path from u to v whose vertices belong to V' is a *path* from u to v *in* V'. If there exists a path from u to v in V' , v is *reachable from* u *in* V'.

A *rooted tree* is a digraph with a vertex of in-degree 0, called the *root*, such that every vertex different from the root has in-degree 1 and is reachable from the root. In a rooted tree, a vertex of out-degree 0 is called a *leaf*.

A *phylogenetic tree* (or *phylogeny*) for a set of taxa is a finite rooted binary tree $\langle V, E\rangle$ along with two finite sets I and S and a function f from $L \times I$ to S, where L is the set of leaves of the tree. The set L represents the given taxonomic units, whereas the set V describes their ancestral units and the set E describes the genetic relationships between them. The elements of I are usually positive integers ("indices") that represent, intuitively, qualitative characters, and elements of S are possible states of these characters. The function f "labels" every leaf v by mapping every index i to the state $f(v, i)$ of the corresponding character in that taxonomic unit.

For instance, Fig. 1 illustrates a phylogeny with $I = \{1, 2\}$ and $S = \{0, 1\}$; $f(v, i)$ is represented by the ith member of the tuple labeling the leaf v.

A character $i \in I$ is *compatible* with a phylogeny (V, E, L, I, S, f) if there exists a function $g : V \times \{i\} \rightarrow S$ such that

(C1) For every leaf v of the phylogeny, $g(v, i) = f(v, i)$;

(C2) For every $s \in S$, if the set

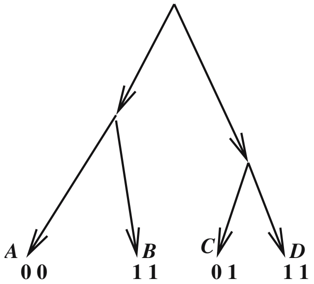
$$V_{is} = \{x \in V : \; g(x, i) = s\}$$

is nonempty, then the digraph $\langle V, E\rangle$ has a subgraph with the set V_{is} of vertices that is a rooted tree.

A character is *incompatible* with a phylogeny if it is not compatible with that phylogeny.

Note in the definition above that, for each character state s, the vertices in V that are mapped to s by such a function g, form a tree. Such a tree provides an explanation as to why the leaves mapped to the character state s by f have the same state at character i: most probably these leaves have inherited the state s from their common ancestor. Note also that, for each character state s, such a function g models an evolution without backmutation or parallel evolution; this conforms with the intuition behind the concept of compatibility mentioned in the introduction.

Fig. 1 A phylogeny for the languages A, B, C, D



For instance, Character 2 is compatible with the phylogeny of Fig. 1: take g to be a function that maps every internal vertex to 1, and every leaf x to $f(x)$. The vertices labeled 1 by g form a tree; the vertices labeled 0 by g also form a tree. On the other hand, Character 1 is incompatible: there is no way of labeling the internal vertices of the tree so that the vertices labeled 1 form a tree and that the vertices labeled 0 form a tree.

The computational problem we are interested in is as follows: Given the sets L, I, S, and the function f, build a phylogeny (V, E, L, I, S, f) with the maximum number of compatible characters. This problem is called the *maximum compatibility problem*. It is NP-hard even when the characters are binary [8].[5]

To solve the maximum compatibility problem, we consider the following problem: given sets L, I, S, a function f from $L \times I$ to S, and a nonnegative integer n, decide the existence of a phylogeny (V, E, L, I, S, f) with at most n incompatible characters.

We describe this decision problem as a logic program (Section 5) whose answer sets correspond to such phylogenies. That is, the program has an answer set iff there is such a phylogeny.

3 Answer Sets

The method we introduce in this paper is based on the answer set semantics of logic programs [21]. In the following, we define the concept of an answer set for the kind of programs considered in this paper.[6]

We begin with a set of propositional symbols, called *atoms*. *Elementary formulas* are atoms and the 0-place connectives $\bot$ and $\top$. *Formulas* are built from elementary formulas by using the unary connective *not* (negation as failure) and the binary connectives , (conjunction) and ; (disjunction). A *rule* is an expression of the form

$$\mathit{Head} \leftarrow \mathit{Body} \tag{1}$$

where *Head* is an atom or $\bot$, and *Body* is a formula.[7] If $\mathit{Head} = \bot$, we will drop the head; rules with the head $\bot$ are called *constraints*. A *program* is a set of rules.

We define when a set X of atoms *satisfies* a formula F (symbolically, $X \models F$) recursively, as follows:

- For elementary F, $X \models F$ if $F \in X$ or $F = \top$,
- $X \models \mathit{not}\ F$ if $X \not\models F$,
- $X \models (F, G)$ if $X \models F$ and $X \models G$,
- $X \models (F; G)$ if $X \models F$ or $X \models G$.

[5] A slight modification of the problem is the *perfect phylogeny problem*, where the goal is to build a "perfect" phylogeny, that is, a phylogeny (V, E, L, I, S, f) according to which every character in I is compatible. This problem is NP-hard [2].

[6] Answer sets are defined for programs of a more general form that may contain classical negation $\neg$ and disjunction [22] and nested expressions [29] in heads of rules as well.

[7] In [29], the syntax of rules is more general: the head may be an arbitrary formula, in particular a disjunction.

A set X of atoms is *closed under* a program Π if, for every rule (1) in Π, $Head \in X$ whenever $X \models Body$.

Let us first define the answer set for a program Π that does not contain negation as failure. We say that X is an *answer set* for Π if X is minimal among the sets of atoms closed under Π. For instance, the set $\{p\}$ is the answer set for the program consisting of the single rule

$$p \leftarrow . \tag{2}$$

Now consider a program Π that may contain negation as failure. The *reduct* Π^X of a program Π relative to a set X of atoms is obtained from Π by replacing every maximal occurrence of a formula of the form *not* F in Π (that is, every occurrence of *not* F that is not in the scope of another *not*) with $\perp$ if $X \models F$, and with $\top$ otherwise. A set X of atoms is an *answer set* for Π if it is the answer set for the reduct Π^X. For instance, the reduct of the program

$$p \leftarrow \mathit{not}\ \mathit{not}\ p \tag{3}$$

relative to $\{p\}$ is also (2). Since $\{p\}$ is the answer set for (2), $\{p\}$ is an answer set for program (3). Similarly, $\{\}$ is an answer set for program (3) as well.

4 Answer Set Programming

The idea of answer set programming (ASP) [28, 33, 40] is to represent a computational problem as a logic program whose answer sets correspond to the solutions of the problem and to find the answer sets for that program by using an answer set solver.

When we represent a problem in answer set programming, two kinds of rules play a major role: those that "generate" many answer sets corresponding to "possible solutions" and those that can be used to "weed out" the answer sets that do not correspond to solutions. For instance, rules (3) are of the former kind: they generate the answer sets $\{p\}$ and $\{\}$. Constraints are of the latter kind. For instance, adding the constraint

$$\leftarrow p$$

to a program, such as (3), eliminates the answer sets for the program that contain p.

In answer set programming, one can use special constructs of the form

$$\{A_1, \ldots, A_n\}^c \tag{4}$$

and of the form

$$l \leq \{A_1, \ldots, A_m\} \leq u \tag{5}$$

in programs, where each A_i is an atom; the nonnegative integers l and u (the "lower bound" and the "upper bound") are optional [40]. (See Lifschitz (unpublished draft), [17] for more information on these constructs.) Programs using these constructs can be viewed as abbreviations for programs in the sense of the previous section, from [18]. For instance, the following program

$$\{p\}^c \leftarrow$$

stands for program (3).

Expression (4) describes subsets of $\{A_1, \ldots, A_n\}$. Such expressions can be used in heads of rules to generate many answer sets. For instance, the answer sets for the program

$$\{p, q, r\}^c \leftarrow \tag{6}$$

are arbitrary subsets of $\{p, q, r\}$.

Expression (5) describes the subsets of the set $\{A_1, \ldots, A_m\}$ whose cardinalities are at least l and at most u. Such expressions can be used in constraints to eliminate some answer sets. For instance, adding the constraint

$$\leftarrow 2 \leq \{p, q, r\}$$

to program (6) eliminates the answer sets for (6) whose cardinalities are at least 2. Adding the constraint

$$\leftarrow \mathit{not}\ (1 \leq \{p, q, r\}) \tag{7}$$

to program (6) eliminates the answer sets for (6) whose cardinalities are not at least 1. Adding the constraint

$$\leftarrow \mathit{not}\ (\{p, q, r\} \leq 1) \tag{8}$$

to program (6) eliminates the answer sets for (6) whose cardinalities are not at most 1.

In the following, the rules

$$\begin{array}{l} \{A_1, \ldots, A_m\}^c \leftarrow \mathit{Body} \\ \leftarrow \mathit{not}\ (l \leq \{A_1, \ldots, A_m\}) \\ \leftarrow \mathit{not}\ (\{A_1, \ldots, A_m\} \leq u) \end{array}$$

will be abbreviated as

$$l \leq \{A_1, \ldots, A_m\}^c \leq u \leftarrow \mathit{Body}.$$

For instance, rules (6)–(8) can be written as

$$1 \leq \{p, q, r\}^c \leq 1 \leftarrow$$

whose answer sets are the singleton subsets of $\{p, q, r\}$.

5 Describing the Problem as a Logic Program

We formalize the problem of phylogeny reconstruction for a set of taxa (as described in Section 2) as a logic program. The inputs to this problem are

- a set L of leaves $0, \ldots, k$ ($k > 0$), representing a set of taxa,
- a set I of (qualitative) characters,
- a set S of (character) states,
- a function f mapping every leaf, for every character, to a state, and
- a nonnegative integer n.

The output is a phylogeny (V, E, L, I, S, f) for L with at most n incompatible characters, if such a phylogeny exists.

The logic program describing the problem has two parts. In the first part, rooted binary trees whose leaves represent the given taxa are generated. In the second part, the rooted binary trees with more than n incompatible characters are eliminated.

Part 1 In this part we generate rooted binary trees with the leaves L. In order to determine whether a character is compatible with a given phylogeny, the names of nonleaf vertices in $V \setminus L$ are not important. What is important for compatibility is the way in which the vertices are connected. (We illustrated compatibility and phylogenies using Fig. 1, which did not even show the names of internal vertices.) Therefore, we may choose an arbitrary set of names for vertices in $V \setminus L$. Even with a fixed set of vertex and leaf names, however, many trees are isomorphic to each other. For example, Fig. 2 shows a rooted binary tree with vertices $V = \{0, \dots, 6\}$, and we may change the names of the internal vertices to get five other isomorphic trees. Since we are interested only in how vertices are connected in the rooted binary trees, we need a way to represent all such isomorphic trees by a single tree. For that we need the following definition.

An *ordered binary tree* $\langle V, E\rangle$ with the leaves $L = \{0, \dots, k\}$ is a rooted binary tree with the leaves L such that

(O1) $V = \{0, \dots, 2k\}$,
(O2) For every edge $(x, y) \in E$, $x > y$, and
(O3) For any two internal vertices x and x_1, $x > x_1$ iff the maximum of the children of x is greater than the maximum of the children of x_1.

For example, Fig. 2 shows an ordered binary tree. The following proposition states that no other ordered binary trees are isomorphic to it.

Proposition 1 *For any rooted binary tree X with the leaves $L = \{0, \dots, k\}$, there is a unique ordered binary tree with the leaves L that is isomorphic to X.*

Proposition 1 allows us to represent isomorphic rooted binary trees canonically with ordered binary trees. We generate ordered binary trees as follows.

From condition (O1), we take $V = \{0, \dots, 2k\}$. (In the rest of the paper, whenever we refer to a set X of edges as a tree, we mean the tree $\langle V, X\rangle$.) Suppose that the edges (x, y) of the ordered binary tree, that is, elements of E, are described by atoms of the form $edge(x, y)$. We describe E as follows.

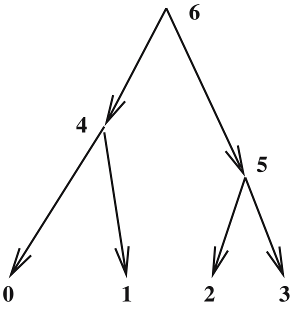


Fig. 2 An ordered binary tree with leaves $L = 0, 1, 2, 3$

First the sets of atoms of the form *edge*(x, y) are "generated" by the rule

$$2 \leq \{edge(x, y) : y \in V, x > y\}^c \leq 2 \leftarrow \quad (x \in V \setminus L). \tag{9}$$

Each set describes a digraph where there is an edge from every internal vertex to two other vertices such that condition (O2) holds. Note that, because of the numbering of the internal vertices above, the in-degree of Vertex $2k$ is 0. Therefore, Vertex $2k$ is the root of the tree.

Then these generated sets of edges are "tested" with some constraints expressing that the set describes a tree – (a) the set describes a connected digraph, and (b) the digraph is acyclic – and with some constraints expressing that this tree is ordered, that is, condition (O3) holds.

To describe (a) and (b), we "define" the reachability of a vertex y from vertex x in $\langle V, E\rangle$:[8]

$$\begin{aligned} &reachable(x, y) \leftarrow edge(x, y) \quad (x, y \in V)\\ &reachable(x, y) \leftarrow edge(x, z), reachable(z, y) \quad (x, y, z \in V). \end{aligned} \tag{10}$$

For (a), we make sure that every vertex different from the root is reachable from the root by the constraint

$$\leftarrow not\ reachable(2k, x) \quad (x \in V \setminus \{2k\}). \tag{11}$$

For (b), we make sure that no vertex is reachable (over a path $v_0, \ldots, v_k$ $(k \geq 1)$) from itself:

$$\leftarrow reachable(x, x) \quad (x \in V). \tag{12}$$

For condition (O3), we first "define" $max_{\text{child}}(x, y)$ ("Child y of vertex x is larger than the sister of y")

$$max_{\text{child}}(x, y) \leftarrow edge(x, y), edge(x, y_1) \quad (x, y, y_1 \in V, y > y_1) \tag{13}$$

and make sure that a vertex x is larger than another vertex x_1 if the largest child of x is larger than that of x_1:

$$\leftarrow max_{\text{child}}(x, y), max_{\text{child}}(x_1, y_1) \quad (x, x_1, y, y_1 \in V, y > y_1, x < x_1). \tag{14}$$

Part 2 In this part we eliminate the rooted binary trees, generated by Part 1 above, with more than n incompatible characters. To do so, we need the following definition and proposition.

Let (V, E, L, I, S, f) be a phylogeny. A character $i \in I$ is *g-incompatible* with (V, E, L, I, S, f) for a function $g : V \times \{i\} \to S$ if condition (C1) holds but condition (C2) does not. For a function $g : V \times I \to S$, we will denote by I_g the set of characters $i \in I$ that are g-incompatible with (V, E, L, I, S, f) for $g|_{V\times\{i\}}$.

[8] Since we need the concept of reachability to express that every vertex different from the root is reachable from the root and that no vertex is reachable (over a path $v_0, \ldots, v_k$ $(k \geq 1)$) from itself, we define the *reachable* relation as the irreflexive transitive closure of the relation *edge*.

Proposition 2 *Let (V, E, L, I, S, f) be a phylogeny. Then the following conditions are equivalent:*

- *There exists a function $g : V \times I \to S$ such that $g|_{L\times I} = f$ and $|I_g| \leq n$.*
- *The number of characters incompatible with the phylogeny (V, E, L, I, S, f) is at most n.*

Proposition 2 allows us to identify the ordered binary trees $\langle V, E\rangle$, generated by Part 1, with more than n incompatible characters, by checking whether the number of g-incompatible characters for some function is greater than n. We describe g-incompatible characters as follows.

According to condition (C1), g coincides with f, where the latter is defined:

$$g(x, i, s) \leftarrow \quad (x \in L,\ f(x, i) = s, i \in I, s \in S). \tag{15}$$

The internal vertices are labeled by exactly one state for each character by the rule

$$1 \leq \{g(x, i, s) : s \in S\}^c \leq 1 \leftarrow \quad (x \in V \setminus L, i \in I). \tag{16}$$

By Proposition 1 of [12], condition (C2) can be equivalently expressed as follows:

(C2)′ For every $s \in S$, if the set

$$V_{is} = \{x \in V :\ g(x, i) = s\}$$

is nonempty, then there is a vertex $r_{is} \in V_{is}$ such that every other vertex in V_{is} is reachable from r_{is} in V_{is}.

Note that, among the vertices in V_{is}, vertex r_{is} is the closest to the root of $\langle V, E\rangle$, that is, r_{is} is not reachable from any other vertex in V_{is} in $\langle V, E\rangle$. Therefore, instead of describing that condition (C2) does not hold, we can describe that condition (C2)′ does not hold.

First, for each character i and for each state s such that V_{is} is not empty, we pick a root x from V_{is} by the rule

$$\{root_{is}(x, i, s)\}^c \leftarrow g(x, i, s) \quad (x \in V, i \in I, s \in S). \tag{17}$$

We make sure that exactly one root is picked by the constraints

$$\leftarrow root_{is}(x, i, s), root_{is}(y, i, s) \quad (x, y \in V, x \neq y, i \in I, s \in S) \tag{18}$$

$$\leftarrow \{root_{is}(x, i, s) : x \in V\} \leq 0,\ g(y, i, s) \quad (y \in V, i \in I, s \in S) \tag{19}$$

and that this root is not reachable from any other vertex y in V_{is} by the constraint

$$\leftarrow root_{is}(x, i, s), g(y, i, s), reachable(y, x) \quad (x, y \in V, x \neq y, i \in I, s \in S). \tag{20}$$

After defining the reachability of a vertex in V_{is} from the root

$$reachable_{is}(x, i, s) \leftarrow root_{is}(x, i, s) \quad (x \in V, i \in I, s \in S) \tag{21}$$

$$reachable_{is}(x, i, s) \leftarrow g(x, i, s), reachable_{is}(z, i, s), edge(z, x) \\ (x, z \in V, i \in I, s \in S), \tag{22}$$

we define the g-incompatibility of a character i for g:

$$incompatible(i) \leftarrow g(x,i,s), not\ reachable_{is}(x,i,s) \qquad (x \in V, i \in I, s \in S). \tag{23}$$

We make sure that there are at most n such g-incompatible characters for some function g by the constraint

$$\leftarrow n+1 \le \{incompatible(i) : i \in I\}. \tag{24}$$

To find a phylogeny with the minimum number of incompatible characters (and thus, by Proposition 2, the minimum number of g-incompatible characters for some function g), we need to find the minimum n such that the program above has an answer set.

Note that, for the minimum n, constraints (19) and (20) are redundant and thus can be dropped from the program above. In our experiments, we drop just constraint (19) for a faster computation.

6 Correctness Properties of the Program

Let Π_1 be the program consisting of rules (9)–(14). The following proposition shows that the answer sets for Π_1 describe ordered binary trees.

Let us denote by E_k the set of all atoms of the form $edge(x,y)$ such that $0 \le y < x \le 2k$. We will identify every edge (x,y) with the atom $edge(x,y)$.

Proposition 3 *For a set X of edges, X is an ordered binary tree with the set $L = \{0,\ldots,k\}$ of leaves iff X can be represented in the form $Z \cap E_k$ for some answer set Z for Π_1. Furthermore, every ordered binary tree with the set L of leaves can be represented in this form in only one way.*

Let Π_2 be the program consisting of rules (15)–(24). Given one of the ordered binary trees $\langle V, E\rangle$ generated by Part 1, a function f labeling the leaves of the tree, and a nonnegative integer n, program Π_2 checks whether the phylogeny (V, E, L, I, S, f) has at most n g-incompatible characters for some function $g : V \times I \rightarrow S$.

Indeed, let us consider the program Π_2' obtained from Π_2 by replacing constraint (20) with

$$\leftarrow root_{is}(x,i,s), g(y,i,s) \qquad (x, y \in V, x \neq y, x \text{is reachable from } y \text{in } \langle V, E\rangle, i \in I, s \in S) \tag{25}$$

and by replacing rule (22) with

$$reachable_{is}(x,i,s) \leftarrow g(x,i,s), reachable_{is}(z,i,s) \qquad (x, z \in V, (z,x) \in E, i \in I, s \in S). \tag{26}$$

Intuitively, Π_2' is the partial evaluation of Π_2 with respect to the tree $\langle V, E\rangle$ described by Π_1.

Proposition 4 *A phylogeny (V, E, L, I, S, f) has at most n g-incompatible characters for some function $g : V \times I \to S$ iff Π'_2 has an answer set.*

Let Π be the program consisting of rules (9)–(24). The following theorem shows that Π correctly describes the maximum compatibility problem stated as a decision problem.

Correctness Theorem for the Phylogency Program *For a given input (L, I, S, f, n), and for a set E of edges that is a rooted binary tree with the leaves L, E describes a phylogeny (V, E, L, I, S, f) with at most n incompatible characters iff E is isomorphic to the ordered binary tree $Z \cap E_k$ for some answer set Z for Π. Furthermore, for every rooted binary tree X with the leaves L, there is only one ordered binary tree isomorphic to X.*

The proofs of Propositions 3 and 4 and the correctness theorem are based on the splitting set theorem and use the method proposed in [14]. They are presented in the Appendix.

7 Useful Heuristics

We can use the answer set solver CMODELS with the phylogeny program described above to solve small instances of the maximum compatibility problem. Larger datasets, such as the Indo-European dataset (Section 10), require the use of some heuristics.

7.1 Informative Characters

Sometimes the maximum compatibility problem for a given input (L, I, S, f, n) can be simplified by making the set I of characters smaller. In particular, we can identify the characters that would be compatible with any phylogeny constructed for the given taxa. For instance, if every taxonomic unit is mapped to a different state at the same character, that is, the character does not have any "essential" state,[9] then we do not need to consider this character in the computation. Similarly, if every taxonomic unit is mapped to the same state at the same character, then the character has only one essential state, and that character can be eliminated. Therefore, we can consider just the characters with at least two essential states. Such a character will be called *informative* because it is incompatible for some phylogeny. With this kind of preprocessing, for instance, for Indo-European language groups, we have found that, out of 275 characters, 21 are informative. After identifying the set of informative characters, we consider I to be this set in the phylogeny program.

[9] Let (V, E, L, I, S, f) be a phylogeny, with $f : L \times I \to S$. A state $s \in S$ is *essential* with respect to a character $j \in I$ if there exist two different leaves l_1 and l_2 in L such that $f(l_1, j) = f(l_2, j) = s$.

7.2 Partial Labelings

As described in [12], we can use partial labelings of vertices considering essential states instead of a total one. To do so, we need to replace rule (16) by the rule

$$\{g(x,i,s) : s \in ES_i\}^c \leq 1 \leftarrow \quad (x \in V \setminus L, i \in I) \tag{27}$$

where ES_i denotes the set of essential states for the informative character i. This heuristic, for instance, improves the computation time by a factor of 3 for the Indo-European languages.

By the definition of a (partial) perfect network in [12], a character i is compatible with respect to a phylogeny (V, E, L, I, S, f) iff there is a partial mapping g from $V \times \{i\}$ to S such that $(V, E, \emptyset, g)$ is a partial perfect network built on the phylogeny $(V, E, L, \{i\}, S, f|_{L\times\{i\}})$. Then, Propositions 4 and 5 from [12] ensure that no solution is lost when the heuristics above are used in the reconstruction of a phylogeny with the maximum number of compatible characters.

7.3 Propagating Labels Up

By condition (C2), every nonempty V_{is} forms a tree in $\langle V, E\rangle$. In each such tree, for every pair of sisters x and y, such that $x, y \in V_{is}$, x and y are labeled for character i in the same way as their parent is labeled. Therefore, to make the computation more efficient, while labeling the internal vertices of the rooted binary tree in Part 2, we can propagate common labels up.

For that, we modify the phylogeny program as follows. First we introduce the atom $g'(x,i,s)$ expressing that vertex x is mapped to state s at character i as a result of propagation of states up the tree.

$$\begin{aligned} g'(x,i,s) &\leftarrow (x \in V,\ f(x,i) = s, i \in I, s \in ES_i) \\ g'(y,i,s) &\leftarrow g'(x,i,s), g'(x',i,s) \\ &\qquad\qquad (\langle y,x\rangle, \langle y,x'\rangle \in E, x < x', i \in I, s \in ES_i) \end{aligned} \tag{28}$$

Then, for each informative character i, we identify the vertices x that are labeled by this process of propagation.

$$marked(x,i) \leftarrow g'(x,i,s) \quad (x \in V, i \in I, s \in ES_i) \tag{29}$$

After that we replace rules (15) with the rules

$$g(x,i,s) \leftarrow g'(x,i,s) \quad (x \in V, i \in I, s \in ES_i) \tag{30}$$

and replace rules (27) with the rules

$$\begin{aligned} \{g(x,i,s) : s \in ES_i\}^c \leq 1 &\leftarrow not\ marked(x,i) \\ &\qquad (x \in V \setminus L, i \in I). \end{aligned} \tag{31}$$

Finally, we consider ES_i instead of S in rules (17)–(23).

With these modifications, for instance, the computation time is improved by a factor of 2 for the *Alcataenia* species.

8 Computation and Evaluation of Phylogenetic Trees

We have applied the computational method described above to three sets of taxa: Chinese dialects, Indo-European languages, and *Alcataenia* (a tapeworm genus) species. Our experiments with these taxa are described in the following three sections.

To compute phylogenies, we have used the answer set solver CMODELS with the programs describing a set of taxa, preprocessing of the taxa, and reconstruction of a phylogeny. CMODELS transforms them into a propositional theory [30] and calls a SAT solver to compute the models of this theory. By the following proposition, these models are identical to the answer sets for the given programs.[10]

Proposition 5 *For a given input* (L, I, S, f, n)*, program* Π *is tight on every set of atoms closed under* Π.

In our experiments, we have used CMODELS (Version 3.55) with the SAT solver ZCHAFF (Version 2004.11.15) [35]. All problem instances used in these experiments are available at http://people.sabanciuniv.edu/~esraerdem/ASP-benchmarks/cladistics.html. Performance from our experiments is summarized in Table 1. All CPU times are in seconds, for a PC with a 733 MHz Intel Pentium III processor and 256 MB RAM, running SuSE Linux (Version 8.1). The computation time includes the time spent by CMODELS for transforming the input programs into a propositional theory and the time spent by ZCHAFF for finding a satisfying interpretation. The instance size is described in terms of the number of atoms and the number of clauses contained in the propositional theory obtained by CMODELS from that instance.

For each taxa and a given integer n, we have computed all phylogenies with at most n incompatible characters, iteratively with a script as follows: at iteration i, compute the ith phylogeny with the input program using CMODELS and then add to the input program a constraint that prevents generation of the answer sets that describe the ith phylogeny.

We can observe from Table 1 that the computation of a phylogeny for *Alcataenia* takes more time compared to the computation of a phylogeny for Chinese dialects and for Indo-European languages; this is due to the lack of domain-specific information about *Alcataenia*.

With these problem instances, we have also tried the answer set solver SMODELS (Version 2.32), and observed that CMODELS performs better in terms of computation time (e.g., a phylogeny for Chinese dialects cannot be computed by SMODELS in several minutes). In the following, we present the computed trees in the Newick format, where the sister subtrees are enclosed by parentheses. For instance, the tree of Fig. 1 can be represented in the Newick format as ((A, B), (C, D)).

We compare the computed phylogenetic trees with respect to three criteria. First, we identify the phylogenies that are plausible. For the Chinese dialects and Indo-European languages, the plausibility of phylogenies depends on the linguistics and archaeological evidence; for *Alcataenia*, the plausibility of the phylogeny we compute depends on the knowledge of host phylogeny (e.g., phylogeny of the seabird family

[10] See [11] for more information on tight programs.

Table 1 Experimental results in terms of computation time (in seconds) and input size

Taxa	CPU time	# of atoms	# of clauses
Chinese dialects	< 1	6,885	27,270
Indo-European languages	< 2	8,480	30,635
Alcataenia species	< 121	8,747	36,385

Alcidae), chronology of the fossil record, and biogeographical evidence. Since our method is based on maximum compatibility, the second criterion is the number of incompatible characters: the more the number of compatible characters, the better the trees are. As pointed out in Section 1, we view reconstructing phylogenies as the first step of reconstructing the evolutionary history of a set of taxonomic units. The second step is then, to obtain a perfect (temporal) phylogenetic network from the reconstructed phylogeny by adding some lateral edges, in the sense of [12, 13, 36]. Therefore, the third criterion is the minimum number of lateral edges (denoting contacts such as borrowings) required to turn the phylogeny into a phylogenetic network.

We also compare these trees to the ones computed by a maximum parsimony method. Usually, in order to compare a set of trees with another set, "consensus trees" are used. A consensus tree "summarizes" a set of trees by retaining components that occur sufficiently often. We have used the program CONSENSE, available with PHYLIP [16], to find consensus trees.

9 Computing Phylogenetic Trees for Chinese Dialects

The computational method described above was used to reconstruct a phylogeny for the Chinese dialects Xiang, Gan, Wu, Mandarin, Hakka, Min, and Yue. We have used the dataset, originally gathered by Xu Tongqiang and processed by Wang Feng, described in [34]. In this dataset, there are 15 lexical characters, and they are all informative. Each character has two to five states. For some characters, their states are presented in Table 2. After the inessential states are eliminated as explained in Section 7, each character has two essential states.

Table 2 The character states of some informative characters for seven Chinese dialects: Xiang, Gan, Wu, Mandarin, Hakka, Min, and Yue

Character	Xiang	Gan	Wu	Mandarin	Hakka	Min	Yue
'eat'	1	1	1	1	2	2	2
'egg'	1	1	1	3	2	2	1
'eye'	1	1	1	1	2	2	1
'feather'	1	2	2	1	2	1	2
'give'	1	1	2	3	4	5	2
'grease'	1	2	1	3	2	2	2
'know'	1	1	1	2	2	2	2
'say'	1	3	2	2	1	1	1
'what'	3	1	4	1	2	1	2

With this dataset, we have computed 33 phylogenies with six incompatible characters, and we have found out that there is no phylogeny with fewer than six incompatible characters. These phylogenies are presented in Table 3 in the Newick format. One feature common to all these 33 phylogenies is that characters “feather” and “what” are not compatible with respect to any of them.

The subgrouping of the Chinese dialects is not yet established. However, many specialists agree that there is a Northern group and a Southern group. That is, for the dialects we chose in our study, we would expect a (Wu, Mandarin, Gan, Xiang) Northern grouping and a (Hakka, Min) Southern grouping. (It is not clear which group Yue belongs to.) Phylogenies 15, 18, 23, 24, and 27 are more plausible than

Table 3 The 33 phylogenies computed for Chinese dialects Xiang, Gan, Wu, Mandarin, Hakka, Min, and Yue, using CMODELS

	Phylogenies	*m*
1	(Mandarin, (Wu, (Xiang, (Gan, (Yue, (Hakka, Min))))))	3
2	(Xiang, (Gan, ((Wu, Mandarin), (Yue, (Hakka, Min)))))	3
3	(Mandarin, ((Wu, (Xiang, Gan)), (Yue, (Hakka, Min))))	3
4	((Wu, (Xiang, Gan)), (Mandarin, (Yue, (Hakka, Min))))	3
5	(Hakka, (Min, (Yue, (Mandarin, (Wu, (Xiang, Gan))))))	3
6	(Min, (Hakka, (Yue, (Mandarin, (Wu, (Xiang, Gan))))))	3
7	(Min, (Hakka, (Yue, (Gan, (Xiang, (Wu, Mandarin))))))	2
8	(Hakka, (Min, (Yue, (Gan, (Xiang, (Wu, Mandarin))))))	2
9	(Xiang, (Gan, (Wu, (Mandarin, (Yue, (Hakka, Min))))))	3
10	((Xiang, Gan), ((Wu, Mandarin), (Yue, (Hakka, Min))))	3
11	(Gan, (Xiang, (Wu, (Mandarin, (Yue, (Hakka, Min))))))	3
12	(Hakka, (Min, (Yue, ((Xiang, Gan), (Wu, Mandarin)))))	3
13	(Gan, ((Xiang, (Wu, Mandarin)), (Yue, (Hakka, Min))))	2
14	(Wu, ((Xiang, Gan), (Mandarin, (Yue, (Hakka, Min)))))	3
*15	((Hakka, Min), (Yue, (Gan, (Xiang, (Wu, Mandarin)))))	2
16	((Xiang, Gan), (Wu, (Mandarin, (Yue, (Hakka, Min)))))	3
17	(Wu, (Mandarin, (Xiang, (Gan, (Yue, (Hakka, Min))))))	3
*18	((Yue, (Hakka, Min)), (Mandarin, (Wu, (Xiang, Gan))))	3
19	(Xiang, ((Wu, Mandarin), (Gan, (Yue, (Hakka, Min)))))	2
20	((Wu, Mandarin), (Xiang, (Gan, (Yue, (Hakka, Min)))))	2
21	(Wu, (Mandarin, ((Xiang, Gan), (Yue, (Hakka, Min)))))	3
22	((Wu, Mandarin), ((Xiang, Gan), (Yue, (Hakka, Min))))	3
*23	((Hakka, Min), (Yue, ((Xiang, Gan), (Wu, Mandarin))))	3
*24	((Yue, (Hakka, Min)), (Gan, (Xiang, (Wu, Mandarin))))	2
25	(Gan, (Xiang, ((Wu, Mandarin), (Yue, (Hakka, Min)))))	3
26	((Xiang, (Wu, Mandarin)), (Gan, (Yue, (Hakka, Min))))	2
*27	((Hakka, Min), (Yue, (Mandarin, (Wu, (Xiang, Gan)))))	3
28	(Yue, ((Hakka, Min), (Mandarin, (Wu, (Xiang, Gan)))))	3
29	(Yue, ((Hakka, Min), (Gan, (Xiang, (Wu, Mandarin)))))	2
30	(Mandarin, (Wu, ((Xiang, Gan), (Yue, (Hakka, Min)))))	3
31	(((Xiang, Gan), (Wu, Mandarin)), (Yue, (Hakka, Min)))	3
32	(Yue, ((Hakka, Min), ((Xiang, Gan), (Wu, Mandarin))))	3
33	(Min, (Hakka, (Yue, ((Xiang, Gan), (Wu, Mandarin)))))	3

The ones marked with the symbol * are more plausible from the viewpoint of historical linguistics. Each of these trees has six incompatible characters and requires *m* lateral edges to turn into a perfect phylogenetic network.

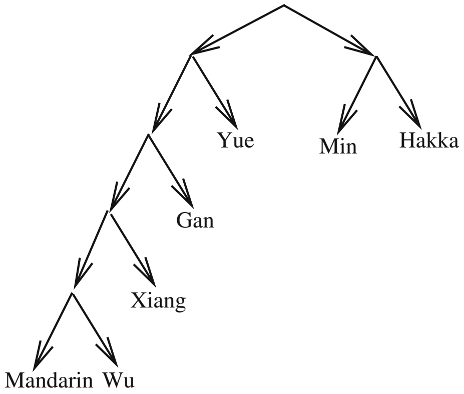


Fig. 3 A phylogeny computed for Chinese dialects, using CMODELS

the other phylogenies with respect to this hypothesis.[11] One of these plausible trees, Phylogeny 15, is presented in Fig. 3. Among these five phylogenies, Phylogenies 15 and 24 require at least two lateral edges (representing borrowings between Gan and Wu, and between (Mandarin, Wu) and Min) to turn into a perfect phylogenetic network; the others require at least three edges.

With the dataset above, we have constructed the five most parsimonious phylogenies using the phylogeny reconstruction program PARS. They are presented in Table 4. We observe that none of these phylogenies is consistent with the hypothesis about the grouping of Northern and Southern Chinese dialects.

Now let us compare the 33 trees of Table 3 with the 55 trees of [34] computed by a method based on the maximum parsimony criterion of [19].

Using the program CONSENSE, we have computed the majority-consensus tree for our 33 phylogenies:

((Yue, (Hakka, Min)), ((Gan, Xiang), (Wu, Mandarin))).

Both this tree and the majority-consensus tree for the 55 most parsimonious trees of [34]

((Yue, (Hakka, Min)), (Wu, Mandarin, Gan, Xiang))

are consistent with the more conventional hypothesis above, grouping Yue with the Southern dialects.

All of the 33 phylogenies we have computed using CMODELS correspond to the trees of Types I–III in [34]. Each of the remaining 22 trees (of Types IV and V) of [34] has seven incompatible characters, but they have the same degree of parsimony as the other 33 trees. This highlights the difference between a maximum parsimony method and a maximum compatibility method.

[11] Based on recent evidence [1], Yue may well be the outgroup of all the other languages; then Phylogenies 28, 29, and 32 would be plausible, too (but perhaps not as plausible as Phylogenies 15, 18, 23, 24, and 27).

Table 4 Five most parsimonious phylogenies for Chinese dialects, computed using PARS

	Phylogeny
1	((Mandarin,Wu),((Yue,(Min,Hakka)),Gan),Xiang)
2	((((Yue,(Min,Hakka)),Mandarin),Wu),Gan,Xiang)
3	(((Yue,(Min,Hakka)),(Mandarin,Wu)),Gan,Xiang)
4	(Mandarin,(Wu,((Yue,(Min,Hakka)),Gan)),Xiang)
5	(Mandarin,(((Yue,(Min,Hakka)),Wu),Gan),Xiang)

10 Computing Phylogenetic Trees for Indo-European Languages

We have applied the computational method described above to reconstruct a phylogeny for the Indo-European languages Hittite, Luvian, Lycian, Tocharian A, Tocharian B, Vedic, Avestan, Old Persian, Classical Armenian, Ancient Greek, Latin, Oscan, Umbrian, Gothic, Old Norse, Old English, Old High German, Old Irish, Welsh, Old Church Slavonic, Old Prussian, Lithuanian, Latvian, and Albanian. We have used the dataset assembled by Don Ringe and Ann Taylor, with the advice of other specialist colleagues. This dataset is described in [38].

There are 282 informative characters in this dataset. Out of 282 characters, 22 are phonological characters encoding regular sound changes that have occurred in the prehistory of various languages, 12 are morphological characters encoding details of inflection (or, in one case, word formation), and 248 are lexical characters defined by meanings on a basic word list. For each character, there are 2–22 states. Some of the character states for some Indo-European languages are shown in Table 5.

To compute phylogenetic trees, we have considered as units the language groups Balto–Slavic, Italo–Celtic, Greco–Armenian, Anatolian, Tocharian, Indo–Iranian, Germanic, and the language Albanian; the seven language groups are shown in Table 6.

For each language group, we have obtained the character states by propagating the character states for languages up, similar to the preprocessing of [12]. For instance, the languages Ancient Greek and Classical Armenian have State 2 at Character "father." These two languages have the same state because, most probably, they have inherited this state from their parent. Therefore, we map proto-Greco–Armenian

Table 5 The character states of some informative characters for six Indo-European languages: Ancient Greek, Old Church Slavonic, Old English, Old High German, Latin, and Old Persian

Character	Ancient Greek	Old Church Slavonic	Old English	Old High German	Latin	Old Persian
'arm'	3	6	8	8	10	13
'beard'	2	5	5	5	5	10
'child'	3	8	10	18	12	15
'father'	2	1	2	2	2	2
'free'	3	8	10	10	3	14
'laugh'	2	7	9	9	11	14
'pour'	3	6	14	14	14	9
'tear'	2	4	2	2	2	7

Table 6 Seven Indo-European language groups

Proto-Language	Subtree
proto-Anatolian	(Hittite,(Luvian,Lycian))
proto-Tocharian	(Tocharian A,Tocharian B)
proto-Italo–Celtic	(((Oscan,Umbrian),Latin),(Old Irish,Welsh))
proto-Germanic	(((Old English,Old High German),Old Norse),Gothic)
proto-Greco–Armenian	(Ancient Greek,Classical Armenian)
proto-Balto–Slavic	(((Lithuanian,Latvian),Old Prussian),Old Church Slavonic)
proto-Indo–Iranian	((Old Persian,Avestan),Vedic)

to State 2 at Character 'father.' After propagating character states up, we have found out that grouping Baltic and Slavic makes one character incompatible, and grouping Italic and Celtic makes six characters incompatible. (For the purposes of this experiment we accept the Italo–Celtic subgroup as found in [38] largely on the basis of phonological and morphological characters.) Other groupings do not make any character incompatible. Therefore, we have not considered these seven characters while computing a phylogenetic tree, since we already know that they would be incompatible with every phylogeny.

Then, we have identified the characters that would be compatible with every phylogeny built for these seven language groups and the language Albanian. For instance, Character "child" is compatible with any phylogeny because it does not have any essential state: every language group is mapped to a different state. By eliminating such characters, as explained in Section 7, we have found out that, out of $282 - 7 = 275$ characters, 21 characters are informative. Out of those 21, two are phonological (P2 and P3) and one is morphological (M5). Each character has 2–3 essential states.

While computing phylogenetic trees for the seven language groups and the language Albanian, we have ensured that each tree satisfies the following domain-specific constraints:

1. Anatolian is the outgroup for all the other subgroups; within the residue, Tocharian is the outgroup.
2. Within the residue of that, Italo–Celtic, and possibly Albanian are outgroups, but not necessarily as a single clade.
3. Albanian cannot be a sister of Indo–Iranian or Balto–Slavic.

The domain-specific information above can be formalized as constraints. For instance, we can express that Anatolian is the outgroup for all the other subgroups by the constraint

$$\leftarrow \textit{not edge}(2k, 6)$$

where $2k$ is the root of the phylogeny and 6 denotes proto-Anatolian.

Another piece of domain-specific information is about the phonological and morphological characters. The phonological and morphological innovations (except the phonological characters P2 and P3) considered in the dataset are too unlikely to have spread from language to language, and independent parallel innovation is practically excluded. (The only qualification involves the phonological characters P2

Table 7 Abbreviations for some Indo-European (proto-)languages

(Proto-)Language	Abbreviation
proto-Anatolian	AN
proto-Tocharian	TO
proto-Italo–Celtic	IC
proto-Germanic	GE
proto-Greco–Armenian	GA
proto-Balto–Slavic	BS
proto-Indo–Iranian	IIR
Albanian	AL

and P3, because it has been claimed that those sound changes might have spread through an already differentiating dialect continuum.) Therefore, while computing phylogenetic trees, we have also ensured that all morphological characters and all phonological characters, except P2 and P3, are compatible with them. This is achieved by adding to the program the constraint

$$\leftarrow incompatible(i) \quad (i \in I \cap MP)$$

where MP is the set of all morphological and phonological characters except P2 and P3.

With 21 informative characters, each with two to three essential states, we have computed 45 phylogenetic trees for the seven language groups above and the language Albanian. Phylogenies 1–45 can be divided into three groups: 1–6 in Group 1, 7–39 in Group 2, and 40–45 in Group 3. These trees are shown, in the Newick format, in Tables 8, 9, and 10. In these tables, abbreviations are used for proto-Balto–Slavic, proto-Italo–Celtic, proto-Greco–Armenian, proto-Anatolian, proto-Tocharian, proto-Indo–Iranian, proto-Germanic, and the language Albanian, as shown in Table 7.

In Group 1, the trees are of the form

(AN, (TO, (AL, (IC, (a tree formed for GE, GA, BS, IIR))))).

In this group the least likely tree is Phylogeny 3 because it puts BS and GA together; Phylogeny 4 is also a little odd because it puts GE and GA together. (There seems to be no good evidence of shared innovations between GA and those subtrees

Table 8 Phylogenies 1–6 computed for Indo-European languages, using CMODELS

	Phylogeny	n	m
*1	(AN, (TO, (AL, (IC, (GE, (GA, (IIR, BS)))))))	17	3
*2	(AN, (TO, (AL, (IC, (GE, (BS, (IIR, GA)))))))	18	3
3	(AN, (TO, (AL, (IC, (GE, (IIR, (BS, GA)))))))	19	
4	(AN, (TO, (AL, (IC, ((IIR, BS), (GA, GE))))))	20	
*5	(AN, (TO, (AL, (IC, ((IIR, GA), (BS, GE))))))	20	>3
*6	(AN, (TO, (AL, (IC, (GA, (GE, (IIR, BS)))))))	20	3

The ones marked with the symbol * are plausible from the viewpoint of historical linguistics. Each phylogeny has n incompatible characters, and each plausible phylogeny requires m lateral edges to turn into a perfect phylogenetic network.

Table 9 Phylogenies 7–39 computed for Indo-European languages, using CMODELS

	Phylogeny	*n*	*m*
*7	(AN, (TO, (IC, ((GE, AL), (GA, (IIR, BS))))))	16	3
*8	(AN, (TO, (IC, ((GE, AL), (BS, (IIR, GA))))))	17	3
*9	(AN, (TO, (IC, (GE, (AL, (GA, (IIR, BS)))))))	17	3
*10	(AN, (TO, (IC, (AL, (GE, (GA, (IIR, BS)))))))	17	3
*11	(AN, (TO, (IC, (GE, (GA, (AL, (IIR, BS)))))))	17	3
*12	(AN, (TO, (IC, (GE, ((IIR, BS), (GA, AL))))))	17	3
*13	(AN, (TO, (IC, (AL, (GE, (BS, (IIR, GA)))))))	18	3
14	(AN, (TO, (IC, ((GE, AL), (IIR, (BS, GA))))))	18	
*15	(AN, (TO, (IC, (GE, (BS, (IIR, (GA, AL)))))))	18	> 3
*16	(AN, (TO, (IC, (GE, (AL, (BS, (IIR, GA)))))))	18	3
*17	(AN, (TO, (IC, (GE, (BS, (AL, (IIR, GA)))))))	18	> 3
18	(AN, (TO, (IC, (AL, (GE, (IIR, (BS, GA)))))))	19	
*19	(AN, (TO, (IC, ((IIR, GA), (BS, (GE, AL))))))	19	> 3
*20	(AN, (TO, (IC, (GE, (IIR, (BS, (GA, AL)))))))	19	> 3
21	(AN, (TO, (IC, (GE, (AL, (IIR, (BS, GA)))))))	19	
22	(AN, (TO, (IC, ((IIR, BS), (GA, (GE, AL))))))	19	
*23	(AN, (TO, (IC, (GA, ((IIR, BS), (GE, AL))))))	19	3
24	(AN, (TO, (IC, (GE, (IIR, (AL, (BS, GA)))))))	19	
*25	(AN, (TO, (IC, (AL, ((IIR, GA), (BS, GE))))))	20	> 3
*26	(AN, (TO, (IC, ((GA, AL), (GE, (IIR, BS))))))	20	3
*27	(AN, (TO, (IC, ((BS, GE), (AL, (IIR, GA))))))	20	> 3
*28	(AN, (TO, (IC, ((IIR, GA), (AL, (BS, GE))))))	20	> 3
*29	(AN, (TO, (IC, (GA, (GE, (AL, (IIR, BS)))))))	20	3
30	(AN, (TO, (IC, (IIR, (GA, (BS, (GE, AL)))))))	20	
*31	(AN, (TO, (IC, (GA, (IIR, (BS, (GE, AL)))))))	20	3
*32	(AN, (TO, (IC, ((GA, GE), (AL, (IIR, BS))))))	20	3
33	(AN, (TO, (IC, (IIR, ((BS, GA), (GE, AL))))))	20	
*34	(AN, (TO, (IC, ((BS, GE), (IIR, (GA, AL))))))	20	> 3
*35	(AN, (TO, (IC, (AL, ((IIR, BS), (GA, GE))))))	20	3
*36	(AN, (TO, (IC, ((IIR, BS), (AL, (GA, GE))))))	20	3
*37	(AN, (TO, (IC, (AL, (GA, (GE, (IIR, BS)))))))	20	3
*38	(AN, (TO, (IC, (GA, (AL, (GE, (IIR, BS)))))))	20	3
*39	(AN, (TO, (IC, ((IIR, BS), (GE, (GA, AL))))))	20	3

The ones marked with the symbol * are plausible from the point of view of historical linguistics. Each phylogeny has *n* incompatible characters, and each plausible phylogeny requires *m* lateral edges to turn into a perfect phylogenetic network.

that are not also shared by other subtrees.) In this group, Phylogeny 1 is marginally better than the others.

In Group 2, the trees are of the form

(AN, (TO, (IC, (a tree formed for GE, GA, BS, IIR, AL)))).

In this group, most of the trees look reasonable; again, the least likely are the ones that put GA and BS together (namely, Phylogenies 14, 18, 21, 24, and 33). Also a little odd are Phylogeny 22 (which puts GA and (GE, AL) together) and Phylogeny 30 (which puts GA, BS, and (GE, AL) in a clade together against IIR). In this group, Phylogeny 7, shown in Fig. 4, is marginally better than any of the other 44.

Table 10 Phylogenies 40–45 computed for Indo-European languages, using CMODELS

	Phylogeny	*n*	*m*
*40	(AN, (TO, ((GE, (GA, (IIR, BS))), (AL, IC))))	17	3
*41	(AN, (TO, ((GE, (BS, (IIR, GA))), (AL, IC))))	18	3
42	(AN, (TO, ((GE, (IIR, (BS, GA))), (AL, IC))))	19	
*43	(AN, (TO, ((GA, (GE, (IIR, BS))), (AL, IC))))	20	3
*44	(AN, (TO, (((IIR, GA), (BS, GE)), (AL, IC))))	20	> 3
45	(AN, (TO, (((IIR, BS), (GA, GE)), (AL, IC))))	20	

The ones marked with the symbol * are plausible from the viewpoint of historical linguistics. Each phylogeny has *n* incompatible characters, and each plausible phylogeny requires *m* lateral edges to turn into a perfect phylogenetic network.

In Group 3, the trees are of the form

(AN, (TO, ((AL, IC), (a tree formed for GE, GA, BS, IIR)))).

In this Group 3, it is the same story: Phylogeny 42 is not very plausible because it puts BS and GA together, and Phylogeny 45 is odd because it puts GA and GE together. Phylogeny 40 is marginally better than the other trees in this group, strictly by the numbers.

In summary, out of the 45 phylogenies computed by using CMODELS, 34 are identified by Don Ringe as plausible from the viewpoint of historical linguistics. Figure 4 shows the most plausible one with 16 incompatible characters. This phylogeny is identical to the phylogeny presented in [38], which was computed with a greedy heuristic using the Perfect Phylogeny software and was used in [12, 13, 36] to build a perfect phylogenetic network for Indo-European.

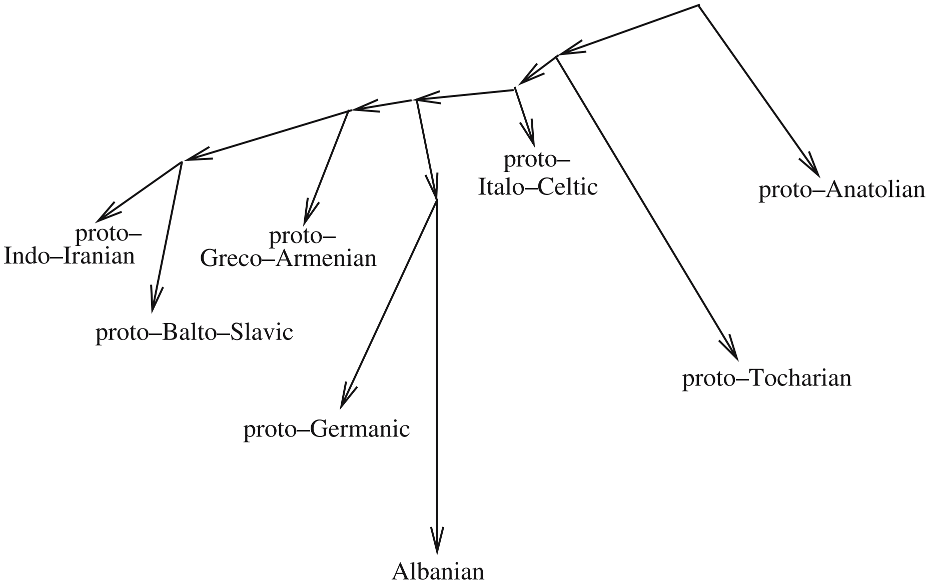


Fig. 4 A phylogeny computed for Indo-European languages, using CMODELS

Table 11 Phylogenies a–c of [37] computed for Indo-European languages Baltic (BA), Slavic (SL), Indic (IN), Iranian (IR), Germanic (GE), Italic (IT), Albanian (AL), Celtic (CE), Greek (GR), Armenian (AR), Hittite (HI), with the dataset of [9], using PAUP

	Phylogeny
1	(((((BA,SL),(IN,IR)),(((GE,IT),AL),CE)),GR),AR),HI)
2	(((((BA,SL),(IN,IR)),((GE,IT,AL),CE)),GR),AR),HI)
3	((((((BA,SL),((GE,IT),CE)),(GR,AR)),IN),IR),AL),HI)

With the same Indo-European dataset obtained after preprocessing (with 21 informative characters, each with two to three essential states), we have also computed a most parsimonious phylogeny using the computational tool PARS:

$$(\mathrm{AN,TO,(GA,((AL,((IC,GE),BS)),IIR)))}\ .$$

Some other most parsimonious phylogenies constructed for Indo-European languages are due to [37], where the authors use PAUP [41] with the dataset Isidore Dyen [9] to generate phylogenies, whose consensus trees are presented in Table 11. None of these most parsimonious trees is consistent with the domain-specific information described above, and thus none is plausible from the viewpoint of historical linguistics. On the other hand, we note that Dyen's dataset is not very reliable since it is a purely lexical database from modern languages. (With this data, one cannot always separate potential contact phenomena from real inheritance.)

11 Computing Phylogenetic Trees for *Alcataenia* Species

With the computational method presented above, we can also infer phylogenies for some species, based on some morphological features. Here we have considered 9 species of *Alcataenia* – a tapeworm genus whose species live in alcid birds (puffins and their relatives): *A. larina*, *A. fraterculae*, *A. atlantiensis*, *A. cerorhincae*, *A. pygmaeus*, *A. armillaris*, *A. longicervica*, *A. meinertzhageni*, *A. campylacantha*. We have used the dataset described in [27].

In this dataset, there are 15 characters, each with two to three states. For some characters, their states are presented in Table 12. After preprocessing, we are left with 10 informative characters, each with two essential states.

Table 12 Character states of some characters for five *Alcataenia* species: *A. longicervica* (LO), *A. cerorhincae* (CE), *A. pygmaeus* (PY), *A. meinertzhageni* (ME), *A. campylacantha* (CA)

Character	LO	CE	PY	ME	CA
genital atrium	1	1	2	1	1
uterus	1	1	1	1	1
size of hooks	1	0	1	2	2
position in host	1	0	1	1	0
position of hooks	1	0	0	2	1
length of neck	2	0	0	0	0

Table 13 Abbreviations for Alcataenia species

Species	Abbreviation
A. Larina	LA
A. fraterculae	FR
A. atlantiensis	AT
A. cerorhincae	CE
A. pygmaeus	PY
A. armillaris	AR
A. longicervica	LO
A. meinertzhageni	ME
A. campylacantha	CA

According to [27], the outgroup for all *Alcataenia* species is *A. larina*. We have expressed this domain-specific information by the constraint

$$\leftarrow \textit{not edge}(2k, 0)$$

where $2k$ is the root of the phylogeny, and 0 denotes *A. larina*.

With the dataset obtained after preprocessing, we have found out that, for *Alcataenia*, no phylogeny has fewer than five incompatible characters. Then we have computed 18 phylogenies, with five incompatible characters, for *Alcataenia*. Abbreviations are used for Alcataenia species as shown in Table 13. The trees are shown, in the Newick format, in Table 14. One of these phylogenies is presented in Fig. 5.

For the plausibility of the phylogenies for *Alcataenia*, we consider the phylogenies of its host *Alcidae* (a seabird family) and the geographical distributions of *Alcidae* (see Table 15). This information is summarized in Table 3 of [27]. For instance, according to host and geographic distributions over the time, diversification of

Table 14 Phylogenies 1–18 computed for *Alcataenia* species, using CMODELS

The ones marked with the symbol * are plausible from the point of view of coevolution of *Alcataenia-Alcidae*, and historical biogeography. Each phylogeny has five incompatible characters, and each plausible phylogeny requires three lateral edges to turn into a perfect phylogenetic network

	Phylogeny
1	(((((((LO,ME),CA),AR),CE),(PY,AT)),FR),LA)
*2	((((((((CA,ME),LO),AR),PY),AT),CE),FR),LA)
3	(((((((CA,ME),LO),AR),PY),(AT,CE)),FR),LA)
4	((((((((CA,ME),LO),AR),PY),CE),AT),FR),LA)
5	((((((LO,ME),CA),AR)((PY,CE),AT)),FR),LA)
6	(((((((LO,ME),CA),AR),(AT,CE)),PY),FR),LA)
7	((((((LO,ME),CA),AR)((PY,AT),PY)),FR),LA)
8	((((((((LO,ME),CA),AR),AT),PY),CE),FR),LA)
9	((((((((LO,ME),CA),AR),CE),PY),AT),FR),LA)
10	((((((LO,ME),CA),AR),((AT,PY),CE)),FR),LA)
11	(((((((LO,ME),CA),AR),PY),(AT,CE)),FR),LA)
12	(((((((LO,ME),CA),AR),AT),(PY,CE)),FR),LA)
13	(((((((LO,ME),CA),AR),(AT,PY)),CE),FR),LA)
14	((((((((LO,ME),CA),AR),CE),AT),PY),FR),LA)
15	((((((((LO,ME),CA),AR),PY),CE),AT),FR),LA)
*16	((((((((LO,ME),CA),AR),PY),AT),CE),FR),LA)
17	((((((((LO,ME),CA),AR),AT),CE),PY),FR),LA)
18	(((((((LO,ME),CA),AR),(CE,PY)),AT),FR),LA)

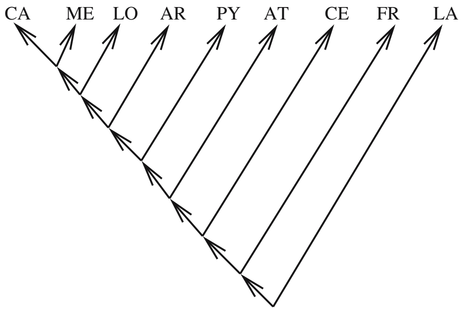


Fig. 5 A plausible phylogeny computed for *Alcataenia* species, using CMODELS

Alcataenia is associated with sequential colonization of puffins (parasitized by *A. fraterculae* and *A. cerorhincae*), razorbills (parasitized by *A. atlantiensis*), auklets (parasitized by *A. pygmaeus*), and murres (parasitized by *A. armillaris*, *A. longicervica*, and *A. meinertzhageni*). This pattern of sequential colonization is supported by the phylogeny of *Alcidae* in [7]. Out of the 18 trees we have computed, only two are consistent with this pattern. One of them, Phylogeny 2 is shown in Fig. 5. These trees are also plausible from the viewpoint of historical biogeography of *Alcataenia* in *Alcidae*, summarized in [27]. Each plausible tree needs three lateral edges to turn into a perfect phylogenetic network.

On the other hand, from the viewpoint of historical biogeography of *Alcataenia* in *Alcidae*, the story has begun with the fluctuations of the sea level (e.g., 100 m reduction) during glacial maxima. These fluctuations have led to the separation of North Pacific basin and Arctic basin. It is conjectured that the origin of *Alcataenia* in *Laridae* (gull family, another host of *A. larina*) was in the North Atlantic (3–3.5 million years ago). Subsequently, range expansion occurred through the Arctic basin and resulted in the development of early holarctic distributions for hosts and parasites (2.5–3 million years ago).[12] Initial entry to the North Pacific basin through the Bering Strait occurred soon after the submergence of Beringia. *Alcataenia* species diversified, following colonization of puffins (parasitized by *A. fraterculae* and *A. cerorhincae*), through sequential colonization and radiation in auklets (parasitized by *A. pygmaeus*), murres (parasitized by *A. armillaris*, *A. longicervica*, and *A. meinertzhageni*), and guillemots (parasitized by *A. campylacantha*) (starting less than 2.5 million years ago). Secondary holarctic ranges were attained later by *Alcataenia* species among murres and guillemots since less than 1.5 million years ago.

Note that the evolutionary and biogeographical histories of *Alcataenia* are congruent.

With the *Alcataenia* dataset described above, we have computed a most parsimonious tree using PARS.

((((((CA,ME),LO),AR),PY),CE,AT),FR,LA)

[12] The holarctic region consists of all the nontropical parts of Europe and Asia, Africa north of the Sahara, and North America south to the Mexican desert region.

Table 15 Host and geographic distribution for nine *Alcataenia* species among *Alcidae*

Species	Host	Geographic range
A. fraterculae	Puffins	North Pacific basin
A. atlantiensis	Razorbills	Eastern Atlantic basin
A. cerorhincae	Auklets	North Pacific basin
A. pygmaeus	Auklets	North Pacific basin
A. armillaris	Murres	Holarctic Region
A. longicervica	Murres	North Pacific basin
A. meinertzhageni	Murres	Holarctic Region
A. campylacantha	Guillemots	North Pacific basin, Holarctic Region, Okhotsk Sea

This tree is very similar to the phylogeny of Fig. 5, and to the most parsimonious phylogeny computed for the *Alcataenia* species above (except *A. atlantiensis*) by Eric Hoberg [26, Fig. 1]:

(((((((CA,ME),LO),AR),PY),CE),FR),LA) .

According to [26, 27], a more plausible phylogeny for *Alcataenia* is the variation of the phylogeny of Fig. 5 where *A. armillaris* and *A. longicervica* are sisters. We can express that *A. armillaris* and *A. longicervica* are sisters by the constraint

$$\leftarrow \mathit{not\ sister}(2,4)$$

where 2 and 4 denote *A. armillaris* and *A. longicervica*, respectively. By adding this constraint to the problem description, we have computed three phylogenies, each with six incompatible characters,

(((((((CA,ME),(AR,LO)),PY),AT),CE),FR),LA)
(((((((CA,ME),(AR,LO)),PY),CE),AT),FR),LA)
((((((CA,ME),(AR,LO)),PY),(CE,AT)),FR),LA)

whose strict consensus tree is

(((((CA,ME),(LO,AR)),PY),AT,CE,FR),LA) .

This consensus tree is identical to the one presented in Fig. 5 of [27]. It is not the most parsimonious tree.

12 Conclusion

We have described how to use answer set programming to generate conjectures about the phylogenies of a set of taxa based on the compatibility of characters. Using this method with the answer set solver CMODELS, we have computed phylogenies for seven Chinese dialects and for 24 Indo-European languages. Some of these trees are plausible from the viewpoint of historical linguistics. We have also computed phylogenies for nine *Alcataenia* species and identified some as more plausible from the viewpoint of coevolution and historical biogeography.

Some of the plausible phylogenies we have computed (e.g., the ones computed for Indo-European) using CMODELS are different from the ones computed using other software, like PARS of PHYLIP, based on maximum parsimony. These results show that the availability of our computational method based on maximum compatibility can be useful for generating conjectures that cannot be found by other computational tools.

One software program that can compute phylogenies for a set of taxa based on the maximum compatibility criterion is CLIQUE (available with PHYLIP), which is applicable only to sets of taxa where a taxonomic unit is mapped to state 0 or state 1 for each character. Another one is the Perfect Phylogeny software of [38], which can compute a phylogeny with the maximum number of compatible characters only when all characters are compatible. Our method is applicable to sets of taxa (like the ones we have experimented with) where a taxonomic unit can be mapped to multiple states. Also, it guarantees to find a tree with the maximum number of compatible characters, if one exists, when all characters may not be compatible. In this sense, our method is more general than the existing ones that compute trees based on maximum compatibility.

Another advantage of our method over the existing ones mentioned above is due to answer set programming. Its declarative representation formalism allows us to easily include in the program domain specific information, and thus to prevent the reconstruction of some phylogenetic trees that are not plausible. Moreover, well-studied properties of programs in this formalism allow us to easily prove that the maximum compatibility problem is correctly described as a decision problem by the phylogeny program.

Acknowledgements We have had useful discussions with Vladimir Lifschitz on the formalization of the problem and with Wang Feng on the plausibility of phylogenies for Chinese dialects. Eric P. Hoberg, Luay Nakhleh, William S.-Y. Wang, and Tandy Warnow have pointed out or supplied relevant references. Anonymous reviewers have helped us improve the presentation of the paper with their questions and suggestions. Dan Brooks was supported by an NSERC Discovery Grant to DRB. Don Ringe was supported by NSF under grant BCS 03-12911. Selim T. Erdoğan was partially supported by the National Science Foundation under Grant IIS-0412907. James Minett was supported in part by grants 1224/02H and 1127/04H awarded by the Research Grants Council of Hong Kong to the Chinese University of Hong Kong. Esra Erdem was supported in part by FWF under project P16536-N04 while she visited Vienna University of Technology, which was made possible by Thomas Eiter; part of this work was done while she visited the University of Toronto, which was made possible by Hector Levesque and Ray Reiter.

Appendix

Proofs of Theorems

We use Proposition 2 and the splitting set theorem of [14], Proposition 12(iii) and Theorem 1 of [17], and Theorem 2 and Propositions 4 and 5 of [11] in our proofs, with programs containing expressions like $\{\ldots\}^c$, like $l\{\ldots\}^c l$, or like $l\{\ldots\}$. This is possible because of the equivalent transformations of [18, Section 4.2].[13]

[13] The propositions and theorems from [14], [11], and [17] are stated in a form slightly simpler than their originals because we don't use $\neg$ in the programs in this paper.

Proposition 2 of [14] *For any program Π and formula F, a set X of atoms is an answer set for $\Pi \cup \{\leftarrow F\}$ iff X is an answer set for Π and does not satisfy F.*

The following definitions are needed for stating the splitting set theorem of [14].

A *splitting set* for a program Π is any set U of atoms such that, for every rule $r \in \Pi$, if the head contains any atoms from U, then all of the atoms in r are in U. The *bottom* of Π relative to U, denoted by $b_U(\Pi)$, is the set of rules $r \in \Pi$ such that all of the atoms in r belong to U. The *top* of Π relative to U is the set $\Pi \setminus b_U(\Pi)$.

The function e_U defined below represents the process of "partial evaluation" of a formula. Consider two sets of atoms U, X and a formula F. For each occurrence of an atom A in F such that $A \in U$, if $A \in X$ replace L with $\top$; otherwise replace L with $\bot$. The new formula obtained will be denoted by $e_U(F, X)$. For a program Π, we will denote by $e_U(\Pi, X)$ the program obtained by replacing each rule $F \leftarrow G$ of Π by $e_U(F, X) \leftarrow e_U(G, X)$.

The Splitting Set Theorem of [14] *Let U be a splitting set for a program Π. A set of atoms is an answer set for Π iff it can be written as $X \cup Y$, where X is an answer set for $b_U(\Pi)$ and Y is an answer set for $e_U(\Pi \setminus b_U(\Pi), X)$.*

Proposition 12(iii) of [17] *Let $A_1, \ldots, A_n$ be pairwise distinct atoms, let l and u be nonnegative integers, and let X be a set of atoms. X is an answer set for $l \leq \{A_1, \ldots, A_n\}^c \leq u$ iff $X \subseteq \{A_1, \ldots, A_n\}$ and $l \leq |X| \leq u$.*

The following definition is needed for stating Theorem 1 of [17].

An atom A is a *head atom* of a program Π if it occurs in the head of at least one rule in Π.

Theorem 1 of [17] *Any answer set for Π is a subset of the set of head atoms of Π.*

In the statements of Proposition 4, Theorem 2, and Proposition 5 of [11], C is a nonempty set of symbols ("object constants"), p and tc are functions from $C \times C$ to the set of atoms such that all atoms $p(x, y)$ and $tc(x, y)$ are pairwise distinct, and Def denotes the definition of the transitive closure tc of a binary predicate p:

$$\begin{array}{l} tc(x, y) \leftarrow p(x, y) \\ tc(x, y) \leftarrow p(x, v), tc(v, y). \end{array}$$

Proposition 4 of [11] *Let Π be a program that does not contain atoms of the form $tc(x, y)$ in the heads of rules. If X is an answer set for $\Pi \cup Def$, then*

$$\{\langle x, y\rangle : tc(x, y) \in X\}$$

is the transitive closure of

$$\{\langle x, y\rangle : p(x, y) \in X\}.$$

The following definitions are needed for stating Theorem 2 of [11].

For any program Π and any set X of atoms, we say about atoms $P, P' \in X$ that P is a *parent* of P' relative to Π and X if there is a rule (1) in Π such that

- $X \models Body$,
- P occurs in $Body$, but not in the scope of negation as failure, and
- $P' = Head$.

For any program Π and any set X of atoms, we say about atoms $P, P' \in X$ that P' is an *ancestor* of P relative to Π and X if there exists a finite sequence of atoms $P_1, \ldots, P_n \in X$ $(n > 1)$ such that $P = P_1$, $P' = P_n$ and for every i $(1 \leq i < n)$, P_{i+1} is a parent of P_i relative to Π and X. In other words, the ancestor relation is the transitive closure of the parent relation.

Theorem 2 of [11] *Let Π be a program that does not contain atoms of the form $tc(x, y)$ in the heads of rules. For any set X of atoms, if*

(i) *Π is tight on X,*
(ii) *$\{\langle x, y\rangle : p(y, x) \in X\}$ is well-founded, and*
(iii) *No atom of the form $tc(x, y)$ is an ancestor of an atom of the form $p(x, y)$ relative to Π and X,*

then $\Pi \cup Def$ is tight on X.

Proposition 5 of [11] *If Π contains constraint*

$$\leftarrow tc(x, x)$$

and C is finite then, for every set X of atoms closed under $\Pi \cup Def$, set $\{\langle x, y\rangle : p(y, x) \in X\}$ is well-founded.

Proof of Proposition 1

Proposition 1 *A rooted binary tree with the leaves $L = \{0, \ldots, k\}$ is uniquely represented by an ordered binary tree with the leaves L.*

We define the height of a vertex v in a tree $\langle V, E\rangle$ as follows.

$$height(v) = \begin{cases} 0 & v \text{ is a leaf} \\ 1 + max_{(v,u)\in E} height(u) & \text{otherwise} \end{cases}$$

The height of a tree is $max_{v\in V} height(v)$.

Lemma 1 *For an ordered binary tree with the leaves $L = \{0, \ldots, k\}$, the vertices with the same height are consecutive numbers.*

Proof of Proposition 1 The proof follows from the following three facts. First, every internal vertex is larger than its children (from (O2)). Second, by Lemma 1, the vertices with the same height are labeled with consecutive numbers. Third, the vertices at some height are labeled in only one way (from (O3)). Therefore, by (O1), the tree is labeled in only one way. □

Proof of Lemma 1 For an ordered binary tree with the leaves $L = \{0, \ldots, k\}$ and with the height H, we want to show that, for every h $(0 \leq h \leq H)$, the vertices with the height h are consecutive numbers.

The proof is by induction on h. The vertices with $h = 0$ are the leaves $L = \{0, \ldots, k\}$. To prove the lemma for vertices with height $h+1$ $(h \geq 0)$, suppose that

(IH) For all j $(0 \leq j \leq h < H)$, the vertices with the height j are consecutive numbers.

If there is only one vertex with the height $h + 1$, then the vertices with the height $h + 1$ are consecutive numbers. Suppose that there are at least two vertices with the height $h + 1$. Let m be the smallest vertex of height $h + 1$ and $m + w$ the largest $(m > k, w > 0)$. Take any vertex m' $(m < m' \leq m + w)$ such that $\mathit{height}(m') = h + 1$. By the induction hypothesis (IH), the definition of *height* of a vertex, and thus every internal vertex, is larger than its children (from (O2)),

(*) For all j $(0 \leq j \leq h < H)$, every vertex with height j is smaller than vertices with heights greater than j.

Since $\mathit{height}(m) = h + 1$ and $m \leq m' - 1$, $\mathit{height}(m' - 1)$ cannot be smaller than $h + 1$ (i.e., $h + 1 \leq \mathit{height}(m' - 1)$); otherwise, it contradicts (*). On the other hand, $\mathit{height}(m' - 1)$ cannot be greater than $h + 1$ (i.e., $\mathit{height}(m' - 1) \leq h + 1$); otherwise, i.e., $\mathit{height}(m' - 1) > h + 1$, by (O2) and the definition of *height*, $m' - 1$ is an ancestor of a vertex m'' $(m' - 1 > m'')$ with the height $h + 1$. By (*), m'' is greater than the vertices with heights less than $h + 1$, thus the children of the vertices with the height $h + 1$. Then, by (O3), $m' - 1$ is greater than any vertex with the height $h + 1$. This contradicts that $\mathit{height}(m') = h + 1$. Then, $\mathit{height}(m' - 1) = h + 1 = \mathit{height}(m')$. Therefore, if there are at least two vertices with the height $h + 1$, then the vertices with the height $h + 1$ are consecutive numbers. □

Proof of Proposition 2

Proposition 2 *Let (V, E, L, I, S, f) be a phylogeny. Then there exists a function $g : V \times I \rightarrow S$ such that $g|_{L\times I} = f$ and $|I_g| \leq n$ iff the number of characters incompatible with the phylogeny (V, E, L, I, S, f) is at most n.*

Proof Let (V, E, L, I, S, f) be a phylogeny. Let IN $(IN \subseteq I)$ denote the set of characters incompatible with this phylogeny. Recall that, for a function $g : V \times I \rightarrow S$, I_g denotes the set of characters $i \in I$ that are g-incompatible with (V, E, L, I, S, f) for $g|_{V\times\{i\}}$. Recall also that a character $i \in I$ is incompatible with (V, E, L, I, S, f) iff i is g-incompatible with (V, E, L, I, S, f) for every function $g : V \times \{i\} \rightarrow S$ such that condition (C1) holds. *Left-to-right.* Let $g : V \times I \rightarrow S$ be a function such that $g|_{L\times I} = f$. By the definition of incompatibility, $IN \subseteq I_g$. Therefore, if $|I_g| \leq n$, then $|IN| \leq n$. *Right-to-left.* Suppose that $|IN| \leq n$. Take a function $g : V \times I \rightarrow S$ such that $g|_{L\times I} = f$, and for which $|I_g|$ is minimum. Such a function exists by the definition of incompatibility. Since $|I_g|$ is minimum, for every character i in I_g, there is no function $g : V \times \{i\} \rightarrow S$ such that conditions (C1) and (C2) hold. That is, I_g is the set of the characters incompatible with (V, E, L, I, S, f). Thus, $|I_g| \leq n$. □

Proof of Proposition 3

Proposition 3 *For a set X of edges, X is an ordered binary tree with the set $L = \{0, \dots, k\}$ of leaves iff X can be represented in the form $Z \cap E_k$ for some answer set Z for Π_1. Furthermore, every ordered binary tree with the set L of leaves can be represented in this form in only one way.*

Recall that E_k denotes the set of all atoms of the form $edge(x, y)$ such that $0 \leq y < x \leq 2k$.

Given a set X of edges, we will denote by R_X the set of atoms $reachable(x, y)$ for all vertices x and y such that there is a nonempty path from x to y over edges in X.

Given a set X of edges, we will denote by M_X the set of atoms $max_Y(x, y)$ for all vertices x and y such that $(x, y) \in X$ and y is the largest of the vertices y', where $(x, y') \in X$.

Let Π_{1_1} be the program consisting of rules (9) and (10) of Π_1. Let Π_{1_2} be the program $\Pi_{1_1} \cup$ (11) $\cup$ (12). Let Π_{1_3} be the program $\Pi_{1_2} \cup$ (13).

Lemma 2 *For a set Z of atoms, Z is an answer set for Π_{1_1} iff $Z = X \cup R_X$ for some $X \subseteq E_k$ such that X is an answer set for (9).*

Lemma 3 *For a set X of edges (x, y) $(x, y \in V = \{0, \dots, 2k\}, y < x)$, X is a rooted binary tree with the set $L = \{0, \dots, k\}$ of leaves iff X can be represented in the form $Z \cap E_k$ for some answer set Z for Π_{1_2}. Furthermore, Z has the form $X \cup R_X$.*

Lemma 4 *For a set Z of atoms, Z is an answer set for Π_{1_3} iff $Z = X \cup R_X \cup M_X$ for some $X \subseteq E_k$ such that $X \cup R_X$ is an answer set for Π_{1_2}.*

Proof of Proposition 3 Let Z be a set of atoms. By Proposition 2 of [14], Z is an answer set for Π_1 iff

(1) Z is an answer set for Π_{1_3} and
(2) Z does not violate constraints (14).
From (1), by Lemma 4, it follows that $Z = X \cup R_X \cup M_X$ such that $X \subseteq E_k$ and $X \cup R_X$ is an answer set for Π_{1_2}. Then, from Lemma 3, $(X \cup R_X) \cap E_k = Z \cap E_k$ is a rooted binary tree with the set L of leaves. From the definition of E_k, conditions (O1) and (O2) are satisfied by X. From (2), M_X does not satisfy the formula

$$max_{\text{child}}(x, y), max_{\text{child}}(x_1, y_1)$$

for all vertices x, x_1, y, y_1 covered by edges in X such that $y > y_1$ and $x < x_1$. That is, condition (O3) is satisfied by X. Therefore, $Z \cap E_k = X$ is an ordered binary tree with the set L of leaves.
Suppose there are two answer sets Z, Z' such that $Z \cap E_k = Z' \cap E_k = Y$ represents an ordered binary tree with the set L of leaves. By Proposition 2 of [14],
(3) Z and Z' are answer sets for Π_{1_3}.

Then, by (3) and Lemma 4,

$$Z = X \cup R_X \cup M_X, \quad Z' = X' \cup R_{X'} \cup M_{X'}$$

for some $X, X' \subseteq E_k$ such that $X \cup R_X$ and $X' \cup R_{X'}$ are answer sets for Π_{1_2}. Then, $Y = X = X'$, and, by the definition of R_X and M_X, $Z = Z'$. □

Proof of Lemma 2 The proof is similar to the proof of the lemma in [14]. Take E_k as a splitting set for Π_{1_1}. The bottom $b_{E_k}(\Pi_{1_1})$ is (9), and the top $\Pi_{1_1} \setminus b_{E_k}(\Pi_{1_1})$ is (10).

The answer sets for $b_{E_k}(\Pi_{1_1})$ are subsets of E_k such that, for every vertex $x \in V \setminus L$,

$$|\{edge(x, y) : y \in V, edge(x, y) \in E_k\}| = 2$$

from Proposition 12(iii) of [17]. For any subset X of E_k, the program $e_{E_k}(\Pi_{1_1} \setminus b_{E_k}(\Pi_{1_1}), X)$ is

$$\begin{array}{ll} reachable(x, y) \leftarrow \top & ((x, y) \in E_k) \\ reachable(x, y) \leftarrow \bot & ((x, y) \notin E_k, x, y \in V) \\ reachable(x, y) \leftarrow \top, reachable(z, y) & ((x, z) \in E_k, y \in V) \\ reachable(x, y) \leftarrow \bot, reachable(z, y) & ((x, z) \notin E_k, x, y, z \in V), \end{array}$$

which can be rewritten as

$$\begin{array}{ll} reachable(x, y) \leftarrow \top & ((x, y) \in E_k) \\ reachable(x, y) \leftarrow \top, reachable(z, y) & ((x, z) \in E_k, y \in V). \end{array} \tag{32}$$

The only answer set for this program is the smallest set that satisfies it, which is R_X. By the splitting set theorem of [14], it follows that a set of atoms is an answer set for Π_{1_1} iff it has the form $X \cup R_X$ for some $X \subseteq E_k$ that is an answer set for (9). □

Proof of Lemma 3 Let X be a set of edges (x, y) $(x, y \in V, y < x)$. Note that the in-degree of the vertex $2k$ is 0. Then X is a rooted binary tree with the root $2k$ and the set L of leaves iff

(a) every vertex x $(0 \leq x < 2k)$ is reachable from $2k$ over the edges in X,
(b) X is acyclic, and
(c) the out-degree of each vertex in L is 0, and the out-degree of each vertex in $V \setminus L$ is 2.

Condition (a) can be expressed by saying that R_X does not satisfy the formulas

$$not\ reachable(2k, x) \tag{33}$$

for all vertices $x \in \{0, \ldots, 2k - 1\}$.

Condition (b) can be expressed by saying that R_X does not satisfy the formulas

$$reachable(x, x) \tag{34}$$

for all vertices $x \in \{0, \ldots, 2k\}$.

Condition (c) can be expressed by saying that X is an answer set for (9)

$$2 \leq \{edge(x, y) : y \in V\}^c \leq 2 \leftarrow$$

for all $x \in V \setminus L$, due Proposition 12(iii) of [17].

By Proposition 2 of [14], the condition

$$X \text{ has the form } Z \cap E_k \text{ for some answer set } Z \text{ for } \Pi_{1_2}$$

is equivalent to

$$X \text{ has the form } Z \cap E_k \text{ for some answer set } Z \text{ for } \Pi_{1_1}$$
$$\text{that does not satisfy formulas (33) and (34)}$$

because Π_{1_2} is the union of Π_{1_1} with the constraints (11) and (12), the bodies of which are (33) and (34), respectively. By Lemma 2, Z has the form $X' \cup R_{X'}$ for some $X' \subseteq E_k$, and the condition above is equivalent to

$$X \text{ has the form } (X' \cup R_{X'}) \cap E_k \text{ for some } X' \subseteq E_k$$
$$\text{such that } X' \text{ is an answer set for (9), and}$$
$$X' \cup R_{X'} \text{ does not satisfy formulas (33) and (34).}$$

Since $(X' \cup R_{X'}) \cap E_k = X'$, $X = X'$, and by the definition of R_X, $R_X = R_{X'}$. Then Z has the form $X \cup R_X$, and the condition above is further equivalent to the condition

$$X \cup R_X \text{ does not satisfy formulas (33) and (34), and}$$
$$X \text{ is an answer set for (9) .}$$

This is equivalent to saying that X is a binary tree with the leaves $0, \ldots, k$ and the root $2k$, that is, conditions (a)–(c) are satisfied. □

Proof of Lemma 4 The proof is similar to the proof of Lemma 2. Take the set $E_k \cup R_{E_k}$ as a splitting set for Π_{1_3}. The bottom $b_{E_k \cup R_{E_k}}(\Pi_{1_3})$ is Π_{1_2} and the top $\Pi_{1_3} \setminus b_{E_k \cup R_{E_k}}(\Pi_{1_3})$ is (13).

The answer sets for $b_{E_k \cup R_{E_k}}(\Pi_{1_3})$, by Proposition 2 of [14], are also answer sets for Π_{1_1}. Then, by Lemma 2, the answer sets for $b_{E_k \cup R_{E_k}}(\Pi_{1_3})$ are of the form $X \cup R_X$ where $X \subseteq E_k$ and X is an answer set for (9). For any such answer set $X \cup R_X$, the program $e_{E_k \cup R_{E_k}}(\Pi_{1_3} \setminus b_{E_k \cup R_{E_k}}(\Pi_{1_3}), X \cup R_X)$ is

$$max_{child}(x, y) \leftarrow \top \quad ((x, y), (x, y_1) \in X, y > y_1).$$

The only answer set for the program above is the smallest set that satisfies it, which is M_X, since, due to Lemma 3, X is a binary tree. By the splitting set theorem of [14], it follows that a set of atoms is an answer set for Π_{1_3} iff it has the form $(X \cup R_X) \cup M_X$ for some $X \subseteq E_k$ such that $X \cup R_X$ is an answer set for Π_{1_2}. □

Proof of Proposition 4

Proposition 4 *A phylogeny (V, E, L, I, S, f) has at most n g-incompatible characters for some function $g : V \times I \to S$ iff Π'_2 has an answer set.*

The proof of Proposition 4 is similar to that of Proposition 3 in that a series of applications of the splitting set theorem is used.

Let (V, E, L, I, S, f) be a phylogeny. Let W be the set of atoms of the form $g(x, i, s)$ $(x \in V, i \in I, s \in S)$. Let O be the set of atoms of the form $root_{is}(x, j, z)$ $(x \in V, j \in I, z \in S)$.

For some $X \subseteq W$, describing a function g mapping $V \times I$ to S such that $g|_{L \times I} = f$, let O_X denote the set consisting of all subsets of O, each containing, for every $j \in I$ and $z \in S$ such that $V_{jz} = \{x : g(x, j, z) \in X\}$ is not empty, exactly one atom $root_{is}(x, j, z)$ for some $x \in V_{jz}$ such that x is not reachable from any other vertex $y \in V_{jz}$ in $\langle V, E\rangle$.

For some $Y \in O_X$, by $R_{X,Y}$ we will denote the set of atoms of the form $reachable_{is}(x, j, z)$ $(x \in V,\ j \in I,\ z \in S)$ satisfying the following condition: If there is some $r \in V_{jz}$ such that $root_{is}(r, j, z) \in Y$, then there is a path from r to x in V_{jz}.

For some $Y \in O_X$, we will denote by $I_{X,Y}$ the set of atoms of the form $incompatible(j)$ $(j \in I)$ satisfying the following condition: for some $x \in V$ and for some $z \in S$ such that $g(x, j, z) \in X$, $reachable_{is}(x, j, z)$ is not in $R_{X,Y}$.

Let Π'_{2_1} be the program consisting of rules (15)–(17) of Π'_2. Let Π'_{2_2} be the program consisting of Π'_{2_1} and rules (18), (19), and (25). Let Π'_{2_3} be the program consisting of Π'_{2_2} and rules (21) and (26). Let Π'_{2_4} be the program consisting of Π'_{2_3} and rules (23).

Lemma 5 *Let (V, E, L, I, S, f) be a phylogeny, and let Z be a set of atoms. Then Z is an answer set for Π'_{2_1} iff $Z = X \cup Y$ for some $X \subseteq W$ and for some $Y \subseteq O$ such that X is an answer set for* (15) $\cup$ (16). *Furthermore, X describes a function g mapping $V \times I$ to S such that $g|_{L\times I} = f$.*

Lemma 6 *Let (V, E, L, I, S, f) be a phylogeny. Let Z be a set of atoms. Then Z is an answer set for Π'_{2_2} iff $Z = X \cup Y$ for some $X \subseteq W$ and for some $Y \in O_X$ such that $X \cup Y$ is an answer set for Π'_{2_1}.*

Lemma 7 *Let (V, E, L, I, S, f) be a phylogeny. Let Z be a set of atoms. Then Z is an answer set for Π'_{2_3} iff $Z = X \cup Y \cup R_{X,Y}$ for some $X \subseteq W$ and for some $Y \in O_X$ such that $X \cup Y$ is an answer set for Π'_{2_2}.*

Lemma 8 *Let (V, E, L, I, S, f) be a phylogeny. Let Z be a set of atoms. Then Z is an answer set for Π'_{2_4} iff $Z = X \cup Y \cup R_{X,Y} \cup I_{X,Y}$ for some $X \subseteq W$, for some $Y \in O_X$ such that $X \cup Y \cup R_{X,Y}$ is an answer set for Π'_{2_3}.*

Proof of Proposition 4 Let (V, E, L, I, S, f) be a phylogeny. Let us identify a function g mapping $V \times I$ to S such that $g|_{L\times I} = f$, with a subset X of W such that $g(x, i) = s$ iff $g(x, i, s) \in X$. Let us also identify the g-incompatible characters for a function X, that is, characters for which condition (C2) (or, from Proposition 1 of [12], equivalently condition (C2)′) does not hold relative to function X, with a set of atoms of the form $incompatible(j)$ $(j \in I)$.

Suppose that the phylogeny (V, E, L, I, S, f) has at most n g-incompatible characters for a function $X \subseteq W$ mapping $V \times I$ to S such that $g|_{L\times I} = f$. For each character j and for some state z such that V_{jz} is not empty, let us pick a vertex $r_{jz} \in V_{jz}$ closest to the root of $\langle V, E\rangle$, i.e., r_{jz} is not reachable from any other vertex $y \in V_{jz}$ in $\langle V, E\rangle$. A character $i \in I$ is g-incompatible with (V, E, L, I, S, f) for X iff some vertex other than r_{jz} in V_{jz} is not reachable from r_{jz}. Let us identify vertices r_{jz} by the atoms $root_{is}(r_{jz}, j, z)$. Then the set of vertices r_{jz} can be described by an element

Y of O_X. Then, the set $I_{X,Y}$ describes the set of all g-incompatible characters for X. Since $|I_{X,Y}| \leq n$, $I_{X,Y}$ does not satisfy the formula

$$n + 1 \leq \{incompatible(i) : i \in I\}. \tag{35}$$

Let Z be a set of atoms. By Proposition 2 of [14], the condition

$$\Pi'_2 \text{ has an answer set } Z$$

is equivalent to

$$\Pi'_{2_4} \text{ has an answer set } Z \text{ such that } Z \text{ does not satisfy formula (35)}$$

because Π'_2 is the union of Π'_{2_4} with the constraint (24), the body of which is (35). By Lemmas 6, 7, and 8, this condition is equivalent to

$$\begin{gathered} Z = X \cup Y \cup R_{X,Y} \cup I_{X,Y} \text{ for some } X \subseteq W, \text{ for some } Y \in O_X \\ \text{such that } X \cup Y \text{ is an answer set for } \Pi'_{2_1}, \text{ and} \\ \text{that } I_{X,Y} \text{ does not satisfy formula (35)} \end{gathered}$$

and by Lemma 5, it follows that

$$\begin{gathered} Z = X \cup Y \cup R_{X,Y} \cup I_{X,Y} \text{ for some } X \subseteq W, \text{ for some } Y \in O_X \\ \text{such that } X \text{ is an answer set for (15)} \cup \text{(16)}, \\ \text{that } X \text{ describes a function } g \text{ mapping } V \times I \text{ to } S \text{ such that } g|_{L\times I} = f, \text{ and} \\ \text{that } I_{X,Y} \text{ does not satisfy formula (35)} . \end{gathered}$$

This is equivalent to saying that, for the phylogeny (V, E, L, I, S, f) and for some function g, the number of g-incompatible characters is at most n. □

Proof of Lemma 5 Take W as a splitting set for Π'_{2_1}. The bottom $b_W(\Pi'_{2_1})$ is (15) ∪ (16), and the top $\Pi'_{2_1} \setminus b_W(\Pi'_{2_1})$ is (17).

The answer sets for $b_W(\Pi'_{2_1})$ are subsets X of W such that, for every vertex x and for every character i, there is exactly one state s such that $g(x, i, s) \in X$. (Furthermore, for any $g(x, i, s) \in X$, if $x \in L$, then $s = f(x, i)$.) For any such subset X, the program $e_W(\Pi'_{2_1} \setminus b_W(\Pi'_{2_1}), X)$ is

$$\{root_{is}(x, i, s)\}^c \leftarrow \top \quad (g(x, i, s) \in X).$$

Every answer set for this program is a subset Y of O. By the splitting set theorem of [14], it follows that a set of atoms is an answer set for Π'_{2_1} iff it has the form $X \cup Y$ for some $X \subseteq W$ and for some $Y \subseteq O$ such that X is an answer set for (15) ∪ (16). □

Proof of Lemma 6 Let (V, E, L, I, S, f) be a phylogeny, and let $X \subseteq W$ be a function mapping $V \times I$ to S such that $g|_{L\times I} = f$.

Let Y be an element of O_X. Then, for each character $i \in S$ and for each state $s \in S$ such that $V_{is} \neq \emptyset$,

$$|\{x : root_{is}(x, i, s) \in Y\}| = 1 \tag{36}$$

and

$$\begin{gathered}\text{for every element } root_{is}(x, i, s)\text{of } Y,\\ \text{for every vertex } m \neq x\text{from which } x\text{is reachable }, m \notin V_{is}.\end{gathered} \tag{37}$$

Equivalently, instead of (36) and (37), we can say that $X \cup Y$ does not satisfy the formulas

$$root_{is}(x, i, s), root_{is}(y, i, s) \tag{38}$$

for all vertices $x, y \in V$ where $x \neq y$,

$$\{root_{is}(x, i, s) : x \in V\}\, 0,\ \ g(y, i, s) \tag{39}$$

for all vertices $y \in V$, and the formulas

$$root_{is}(x, i, s), g(y, i, s) \tag{40}$$

for all vertices $x, y \in V$, $x \neq y$, where x is reachable from y.

By Proposition 2 of [14], the condition

$$Z \text{ is an answer set for } \Pi'_{2_2}$$

is equivalent to

$$\begin{gathered}Z \text{ is an answer set for } \Pi'_{2_1}\text{, and}\\ Z \text{ does not satisfy formulas (38)–(40)}\end{gathered}$$

because Π'_{2_2} is the union of Π'_{2_1} with the constraints (18), (19), and (25), the bodies of which are (38)–(40) respectively. By Lemma 5, this condition is equivalent to

$$\begin{gathered}Z = X \cup Y \text{ for some } X \subseteq W \text{ and for some } Y \subseteq O \text{ such that}\\ X \text{ is an answer set for (15)} \cup \text{(16)},\\ X \text{ describes a function } g \text{ mapping } V \times I \text{ to } S \text{ such that } g|_{L\times I}= f\text{, and}\\ Z \text{ does not satisfy formulas (38)–(40)},\end{gathered}$$

which is, by the definition of O_X, further equivalent to saying that

$$\begin{gathered}Z = X \cup Y \text{ for some } X \subseteq W \text{ and for some } Y \in O_X \text{ such that}\\ X \cup Y \text{ is an answer set for } \Pi'_{2_1}.\end{gathered}$$

□

Proof of Lemma 7 Take $W \cup O$ as a splitting set for Π'_{2_3}. The bottom $b_{W\cup O}(\Pi'_{2_3})$ is Π'_{2_2}, and the top $\Pi'_{2_3} \setminus b_{W\cup O}(\Pi'_{2_3})$ is (21) ∪ (26).

From Lemma 6, answer sets A for Π'_{2_2} are of the form $X \cup Y$ for some $X \subseteq W$, for some $Y \in O_X$. For any answer set A, the program $e_{W \cup O}(\Pi'_{2_3} \setminus b_{W \cup O}(\Pi'_{2_3}), A)$ is

$$\begin{aligned} reachable_{is}(x, i, s) &\leftarrow \top \quad (root_{is}(x, i, s) \in A) \\ reachable_{is}(x, i, s) &\leftarrow reachable_{is}(z, i, s) \\ &\qquad (x, z \in V, (z, x) \in E, g(x, i, s) \in A). \end{aligned}$$

The answer set for this program is $R_{X,Y}$.

Then, by the splitting set theorem of [14], it follows that a set of atoms is an answer set for Π'_{2_3} iff it has the form $X \cup Y \cup R_{X,Y}$ for some $X \subseteq W$ and for some $Y \in O_X$ such that $X \cup Y$ is an answer set for Π'_{2_2}. □

Proof of Lemma 8 Let Q be the set of atoms of the form $reachable_{is}(x, j, z)$ for $x \in V$, $j \in I$ and $z \in S$. Take $W \cup O \cup Q$ as a splitting set for Π'_{2_4}. The bottom $b_{W \cup O \cup Q}(\Pi'_{2_4})$ is Π'_{2_3}, and the top $\Pi'_{2_4} \setminus b_{W \cup O \cup Q}(\Pi'_{2_4})$ is (23).

By Lemma 7, answer sets A for Π'_{2_3} are of the form $X \cup Y \cup R_{X,Y}$ for some $X \subseteq W$ and for some $Y \in O_X$. For any such answer set A, the program $e_{W \cup O \cup Q}(\Pi'_{2_4} \setminus b_{W \cup O \cup Q}(\Pi'_{2_4}), A)$ is

$$incompatible(i) \leftarrow \top \quad (g(x, i, s) \in A, reachable_{is}(x, i, s) \notin A).$$

The answer set for this program is $I_{X,Y}$.

By the splitting set theorem of [14], it follows that a set of atoms is an answer set for Π'_{2_4} iff it has the form $X \cup Y \cup R_{X,Y} \cup I_{X,Y}$ for some $X \subseteq W$ and for some $Y \in O_X$ such that $X \cup Y \cup R_X$ is an answer set for Π'_{2_3}. □

Proof of the Correctness Theorem for the Phylogeny Program

Correctness Theorem for the Phylogeny Program *For a given input (L, I, S, f, n), and for a set E of edges that is a rooted binary tree with the leaves L, E describes a phylogeny (V, E, L, I, S, f) with at most n incompatible characters iff E can be represented by the ordered binary tree $Z \cap E_k$ for some answer set Z for Π. Furthermore, for every rooted binary tree X with the leaves L, there is only one ordered binary tree isomorphic to X.*

Proof Take $E_k \cup R_{E_k} \cup M_{E_k}$ as a splitting set for Π. The bottom $b_{E_k \cup R_{E_k} \cup M_{E_k}}(\Pi)$ is Π_1, and the top $\Pi \setminus b_{E_k \cup R_{E_k} \cup M_{E_k}}(\Pi)$ is Π_2.

By Proposition 3, every answer set X for $b_{E_k \cup R_{E_k} \cup M_{E_k}}(\Pi)$ describes an ordered binary tree with the set L of leaves and thus, by Proposition 1, uniquely represents a rooted binary tree E with the set L of leaves. For any such answer set X, the program

$$e_{E_k \cup R_{E_k} \cup M_{E_k}} \left(\Pi \setminus b_{E_k \cup R_{E_k} \cup M_{E_k}}(\Pi), X\right)$$

is program Π'_2. By the splitting set theorem of [14], it follows that a set Z of atoms is an answer set for Π iff it has the form $X \cup Y$ such that X is an answer set for Π_1 and Y is an answer set for Π'_2. Note also that $Z \cap E_k = X \cap E_k$, since $Y \cap E_k = \emptyset$ from Theorem 1 of [17].

By Proposition 4, Π'_2 has an answer set iff the phylogeny described by E has at most n g-incompatible characters for some function $g : V \times I \rightarrow S$. By Proposition 2,

Π'_2 has an answer set iff phylogeny described by E has at most n incompatible characters.

Therefore, Π has an answer set Z iff $Z \cap E_k$ represents a rooted binary tree E with the set L of leaves such that the phylogeny (V, E, L, I, S, f) has at most n incompatible characters □

Proof of Proposition 5

Proposition 5 *For a given input* (L, I, S, f, n)*, program* Π *is tight on every set of atoms closed under* Π.

Proof For a set Y of atoms closed under Π:

(1) $(\Pi \setminus ((21) \cup (22))) \setminus (10)$ is absolutely tight;
(2) *reachable* is the transitive closure of *edge* (Proposition 4 of [11]);
(3) the set $\{\langle x, y\rangle : edge(y, x) \in Y\}$ is well-founded (Proposition 5 of [11], with finite V, (2) above, and constraint (12) in Π);
(4) none of the atoms of the form $reachable(x, y)$ is an ancestor of an atom of the form $edge(x, y)$ relative to $(\Pi \setminus ((21) \cup (22))) \setminus (10)$ and Y;
(5) $\Pi \setminus ((21) \cup (22))$ is tight on Y (Theorem 2 of [11], with (1)–(4) above), that is, the parent relation relative to $\Pi \setminus ((21) \cup (22))$ and Y is well-founded;
(6) none of the atoms of the form $reachable_{is}(x, i, s)$ is an ancestor of an atom of the form $edge(x, y)$, an atom of the form $root_{is}$, or an atom of the form $g(x, i, s)$ relative to $(\Pi \setminus ((21) \cup (22)))$ and Y;
(7) no atom $reachable_{is}(x, i, s) \in Y$ is an ancestor of itself relative to Π and Y (Y is closed under Π, and (3) above); and
(8) the parent relation relative to Π and Y is well-founded ((5)–(7) above), that is, Π is tight on Y. □

References

1. Ben Hamed, M.: Neighbour-nets portray the Chinese dialect continuum and the linguistic legacy of China's demic history. Proc. R. Soc. Lond., B **272**(1567), 1015–1022 (2005)
2. Bodlaender, H.L., Fellows, M.R., Hallett, M.T., Wareham, H.T., Warnow, T.J.: The hardness of perfect phylogeny, feasible register assignment and other problems on thin colored graphs. Theor. Comp. Sci. **244**(1–2), 167–188 (2000)
3. Brooks, D.R., Erdem, E., Minett, J.W., Ringe, D.: Character-based cladistics and answer set programming. In: Proceedings of the Seventh International Symposium on Practical Aspects of Declarative Languages (PADL), pp. 37–51 (2005)
4. Brooks, D.R., Mayden, R.L., McLennan, D.A.: Phylogeny and biodiversity: conserving our evolutionary legacy. Trends Ecol. Evol. **7**, 55–59 (1992)
5. Brooks, D.R., McLennan, D.A.: Phylogeny, Ecology, and Behavior: A Research Program in Comparative Biology. University of Chicago Press, Chicago, IL (1991)
6. Camin, J.H., Sokal, R.R.: A method for deducing branching sequences in phylogeny. Evolution **19**, 311–326 (1965)
7. Chandler, R.M.: Phylogenetic analysis of the alcids. Ph.D. thesis, University of Kansas (1990)
8. Day, W.H.E., Sankoff, D.: Computational complexity of inferring phylogenies by compatibility. Syst. Zool. **35**(2), 224–229 (1986)
9. Dyen, I., Kruskal, J.B., Black, P.: An Indo-European classification: a lexicostatistical experiment. Trans. Am. Philos. Soc. **82**, 1–132 (1992)

10. Edwards, A.W.F., Cavalli-Sforza, L.L.: Reconstruction of evolutionary trees. In: Phenetic and Phylogenetic Classification, pp. 67–76 (1964)
11. Erdem, E., Lifschitz, V.: Tight logic programs. Theor. Pract. Log. Prog. **3**(4–5), 499–518 (2003)
12. Erdem, E., Lifschitz, V., Nakhleh, L., Ringe, D.: Reconstructing the evolutionary history of Indo-European languages using answer set programming. In: Proceedings of the Fifth International Symposium on Practical Aspects of Declarative Languages (PADL), pp. 160–176 (2003)
13. Erdem, E., Lifschitz, V., Ringe, D.: Temporal phylogenetic networks and logic programming. Theory Pract. Log. Program. **6**(5), 539–558 (2006)
14. Erdoğan, S.T., Lifschitz, V.: Definitions in answer set programming. In: Proceedings of the Seventh International Conference on Logic Programming and Nonmonotonic Reasoning, pp. 114–126 (2004)
15. Felsenstein, J.: Numerical methods for inferring evolutionary trees. Q. Rev. Biol. **57**, 379–404 (1982)
16. Felsenstein, J.: PHYLIP (phylogeny inference package) version 3.6 (2004) (Distributed by the author)
17. Ferraris, P., Lifschitz, V.: Mathematical foundations of answer set programming. In: We Will Show Them! Essays in Honour of Dov Gabbay, vol. 1, College Publications, pp. 615–664 (2005)
18. Ferraris, P., Lifschitz, V.: Weight constraints as nested expressions. Theor. Pract. Log. Prog. **5**, 45–74 (2005)
19. Fitch, W.M.: Toward defining the course of evolution: minimum change for a specific tree topology. Syst. Zool. **20**(4), 406–416 (1971)
20. Foulds, L.R., Graham, R.L.: The Steiner tree problem in phylogeny is NP-complete. Adv. Appl. Math. **3**, 43–49 (1982)
21. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Kowalski, R., Bowen, K. (eds.) Logic Programming: Proceedings of the Fifth International Conference and Symposium, pp. 1070–1080 (1988)
22. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Gener. Comput. **9**, 365–385 (1991)
23. Hennig, W.: Grundzuege einer Theorie der phylogenetischen Systematik. Deutscher Zentralverlag (1950)
24. Hennig, W.: Phylogenetic systematics. Annu. Rev. Entomol. **10**, 97–116 (1965)
25. Hennig, W.: Phylogenetic Systematics. University of Illinois Press (1966) Translated from Grundzuege einer Theorie der phylogenetischen Systematik (1950) by D.D. Davis and R. Zangerl
26. Hoberg, E.P.: Evolution and historical biogeography of a parasite-host assemblage: *Alcataenia* spp. (Cyclophyllidea: Dilepididae) in Alcidae (Chradriiformes). Can. J. Zool. **64**, 2576–2589 (1986)
27. Hoberg, E.P.: Congruent and synchronic patterns in biogeography and speciation among seabirds, pinnipeds, and cestodes. J. Parasitol. **78**(4), 601–615 (1992)
28. Lifschitz, V.: Answer set programming and plan generation. Artif. Intell. **138**, 39–54 (2002)
29. Lifschitz, V., Tang, L.R., Turner, H.: Nested expressions in logic programs. Ann. Math. Artif. Intell. **25**, 369–389 (1999)
30. Lloyd, J.: Foundations of Logic Programming. Springer, Berlin Heidelberg New York (1984)
31. Mair, V.H. (ed.): The Bronze Age and Early Iron Age Peoples of Eastern Central Asia. Institute for the Study of Man, Washington (1998)
32. Mallory, J.P.: In Search of the Indo-Europeans. Thames and Hudson, London (1989)
33. Marek, V., Truszczyński, M.: Stable models and an alternative logic programming paradigm. In: The Logic Programming Paradigm: A 25-Year Perspective, pp. 375–398. Springer, Berlin Heidelberg New York (1999)
34. Minett, J.W., Wang, W.S.-Y.: On detecting borrowing: distance-based and character-based approaches. Diachronica **20**(2), 289–330 (2003)
35. Moskewicz, M.W., Madigan, C.F., Zhao, Y., Zhang, L., Malik, S.: Chaff: engineering an efficient SAT solver. In: Proceedings of Design Automation Conference (DAC2001) (2001)
36. Nakhleh, L., Ringe, D., Warnow, T.: Perfect phylogenetic networks: a new methodology for reconstructing the evolutionary history of natural languages. Language **81**(2), 382–420 (2005)
37. Rexova, K., Frynta, D., Zrzavý, J.: Cladistic analysis of languages: Indo-European classification based on lexicostatistical data. Cladistics **19**, 120–127 (2003)
38. Ringe, D., Warnow, T., Taylor, A.: Indo-European and computational cladistics. Trans. Philol. Soc. **100**(1), 59–129 (2002)

39. Roberts, R.G., Jones, R.M., Smith, M.A.: Thermoluminescence dating of a 50,000-year-old human occupation site in Northern Australia. Science **345**, 153–156 (1990)
40. Simons, P., Niemelä, I., Soininen, T.: Extending and implementing the stable model semantics. Artif. Intell. **138**, 181–234 (2002)
41. Swofford, D.L.: PAUP*: phylogenetic analysis under parsimony (and other methods), version 4.0. Sinauer Associates, Sunderland, MA (2003)
42. White, J.P., O'Connell, J.F.: A Prehistory of Australia, New Guinea, and Sahul. Academic, San Diego, CA (1982)