THE COMPLEXITY OF ROOTED PHYLOGENY PROBLEMS

MANUEL BODIRSKY AND JENS K. MUELLER

CNRS/LIX, École Polytechnique, Palaiseau, France
e-mail address: bodirsky@lix.polytechnique.fr

Friedrich-Schiller-University, Jena, Germany
e-mail address: jkm@informatik.uni-jena.de

Abstract. Several computational problems in phylogenetic reconstruction can be formulated as restrictions of the following general problem: given a formula in conjunctive normal form where the literals are *rooted triples*, is there a rooted binary tree that satisfies the formula? If the formulas do not contain disjunctions, the problem becomes the famous rooted triple consistency problem, which can be solved in polynomial time by an algorithm of Aho, Sagiv, Szymanski, and Ullman. If the clauses in the formulas are restricted to disjunctions of negated triples, Ng, Steel, and Wormald showed that the problem remains NP-complete. We systematically study the computational complexity of the problem for all such restrictions of the clauses in the input formula. For certain restricted disjunctions of triples we present an algorithm that has sub-quadratic running time and is asymptotically as fast as the fastest known algorithm for the rooted triple consistency problem. We also show that any restriction of the general rooted phylogeny problem that does not fall into our tractable class is NP-complete, using known results about the complexity of Boolean constraint satisfaction problems. Finally, we present a pebble game argument that shows that the rooted triple consistency problem (and also all generalizations studied in this paper) cannot be solved by Datalog.

1. Introduction

Rooted phylogeny problems are fundamental computational problems for phylogenetic reconstruction in computational biology, and more generally in areas dealing with large amounts of data about rooted trees. Given a collection of *partial information* about a rooted tree, we would like to know whether there exists a single rooted tree that *explains* the data. A concrete example of a computational problem in this context is the *rooted*

2000 ACM Subject Classification: Theory of Computation, Mathematical Logic, logic and constraint programming, model theory.

Key words and phrases: Constraint Satisfaction Problems, Phylogenetic Reconstruction, Computational Complexity, Datalog, ω-categorical structures.

A preliminary version of this work appeared in the Proceedings of the 13th International Conference on Database Theory (ICDT), 2010.

The first author has received funding from the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013 Grant Agreement no. 257039).

LOGICAL METHODS
IN COMPUTER SCIENCE
DOI:10.2168/LMCS-???

triple consistency problem. We are given a set V of variables, and a set of triples $ab|c$ with $a, b, c \in V$, and we would like to know whether there exists a rooted tree T with leaf set V such that for each of the given triples $ab|c$ the youngest common ancestor of a and b in this tree is below the youngest common ancestor of a and c (if such a tree exists, we say that the instance is satisfiable).

The rooted triple consistency problem has an interesting history. The first polynomial time algorithm for the problem was discovered by Aho, Sagiv, Szymanski, and Ullman [ASSU81], motivated by problems in database theory. This algorithm was later rediscovered for phylogenetic analysis [Ste92]. Henzinger, King, and Warnow [HKW96] showed how to use decremental graph connectivity algorithms to improve the quadratic runtime $O(mn)$ of the algorithm by Aho et al. to a deterministic algorithm with runtime $O(m\sqrt{n})$.

Dekker [Dek86] asked the question whether there is a finite set of 'rules' that allows to infer a triple $ab|c$ from another given set of triples Φ if all trees satisfying Φ also satisfy $ab|c$. This question was answered negatively by Bryant and Steel [BS95]. Dekker's 'rules' have a very natural interpretation in terms of Datalog programs. Datalog as an algorithmic tool for rooted phylogeny problems is more powerful than Dekker's rules. We say that a Datalog program *solves* the rooted triple consistency problem if it derives a distinguished 0-ary predicate *false* on a given set of triples if and only if the instance of the rooted triple consistency problem is not satisfiable. One of the results of this paper is the proof that there is no Datalog program that solves the rooted triple consistency problem.

Datalog inexpressibility results are known to be very difficult to obtain, and the few existing results often exhibit interesting combinatorics [KV95, ASY91, FV99, Gro94, BK10]. The tool we apply to show our result, the existential pebble game, originates in finite model theory, and was successfully applied to finite domain constraint satisfaction [KV98]. A recent generalization of the intimate connection between Datalog and the existential pebble game to a broad class of infinite domain constraint satisfaction problems [BD08] allows us to apply the game to study the expressive power of Datalog for the rooted triple consistency problem.

There are several other important rooted phylogeny problems One is the *subtree avoidance problem*, introduced by [NSW00], or the *forbidden triple problem* [Bry97]; both are NP-hard. It turns out that all of those problems and many other rooted phylogeny problems can be conveniently put into a common framework, which we introduce in this paper.

A *rooted triple formula* is a formula Φ in conjunctive normal form where all literals are of the form $ab|c$. It turns out that the problems mentioned above and many other rooted phylogeny problems (we provide more examples in Section 2) can be formalized as the satisfiability problem for a given rooted triple formula Φ where the set of clauses that might be used in Φ is (syntactically) restricted. If $\mathcal{C}$ is a class of clauses, and the input is confined to rooted triple formulas with clauses from $\mathcal{C}$, we call the corresponding computational problem the *rooted phylogeny problem for clauses from* $\mathcal{C}$.

In this paper, we determine for all classes of clauses $\mathcal{C}$ the computational complexity of the rooted phylogeny problem for clauses from $\mathcal{C}$. In all cases, the corresponding computational problem is either in P or NP-complete. In our proof of the complexity classification we apply known results from Boolean constraint satisfaction. The rooted phylogeny problem is closely related to a corresponding *split problem* (defined in Section 4), which is a Boolean constraint satisfaction problem where we are looking for a *non-trivial* solution, i.e., a solution where at least one variable is set to true and at least one variable is set to false. The complexity of Boolean split problems has been classified in [CKS01]. If $\mathcal{C}$ is such that

the corresponding split problem can be solved efficiently, our algorithmic results in Section 4 show that the rooted phylogeny problem for clauses from $\mathcal{C}$ can be solved in polynomial time. Conversely, we present a general reduction that shows that if the split problem is NP-hard, then the rooted phylogeny problem for $\mathcal{C}$ is NP-hard as well.

2. Phylogeny Problems

We fix some standard terminology concerning rooted trees. Let T be a tree (i.e., an undirected, acyclic, and connected graph) with a distinguished vertex r, the *root* of T. The vertices with exactly one neighbor in T are called *leaves*. The vertices of T are denoted by $V(T)$, and the leaves of T by $L(T) \subseteq V(T)$. For $u, v \in V(T)$, we say that u *lies below* v if the path from u to r passes through v. We say that u *lies strictly below* v if u lies below v and $u \neq v$. The *youngest common ancestor (yca)* of two vertices $u, v \in V(T)$ is the node w such that both u and v lie below w and w has maximal distance from r. Note that the yca, viewed as a binary operation, is commutative and associative, and hence there is a canonical definition of the yca of a set of elements $u_1, \ldots, u_k$. The tree T is called *binary* if the root has two neighbors, and every other vertex has either three neighbors or one neighbor. A neighbor u of a vertex v is called a *child* of v (and v is called the *parent* of u) in T if the distance of u from the root is strictly larger than the distance of v from the root. We write $uv|w$ (or say that $uv|w$ *holds* in T) if u, v, w are distinct leaves of T and $yca(u, v)$ lies strictly below $yca(u, w)$ in T. Note that for distinct leaves u, v, w of any binary tree T, exactly one of the triples $uv|w$, $uw|v$, and $vw|u$ holds in T.

Definition 2.1. A *rooted triple formula* is a (quantifier-free) conjunction of *clauses* (also called *triple clauses*) where each clause is a disjunction of literals of the form $xy|z$.

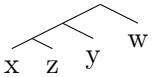
Example 2.2. An example of a triple clause is $xz|y \vee yz|x$; it will also be denoted by $xy \nmid z$. Another example of a triple clause is $xy|z_1 \vee xy|z_2$.

The following notion is used frequently in later sections. If Φ is a formula, and S is a subset of the variables of Φ, then $\Phi[S]$ denotes the conjunction of all those clauses in Φ that only contain variables from S.

Definition 2.3. A rooted triple formula Φ is *satisfiable* if there exists a rooted binary tree T and a mapping α from the variables of Φ to the leaves of T such that in every clause at least one literal is satisfied. A literal $xy|z$ is *satisfied* by (T, α) if $\alpha(x), \alpha(y), \alpha(z)$ are distinct and if $yca(\alpha(x), \alpha(y))$ lies strictly below $yca(\alpha(x), \alpha(z))$ in T. The pair (T, α) is then called a *solution* to Φ.

We would like to remark that a rooted triple formula Φ is satisfiable if and only if there exists a rooted binary tree T and an *injective* mapping α from the variables of Φ to the leaves of T such that the formula evaluates under α to true.

Example 2.4. Let $\Phi = xz|y \vee yz|x \wedge xy|w$ be a rooted triple formula with variables $V = \{w, x, z, y\}$. Then the tree T



together with the identity mapping on V is a solution to Φ.

A fundamental problem in phylogenetic reconstruction is the rooted triple consistency problem [HKW96, BS95, Ste92, ASSU81]. This problem can be stated conveniently in terms of rooted triple formulas.

Problem 2.5 (Rooted-Triple-Consistency)**.**
INSTANCE: A rooted triple formula Φ without disjunction.
QUESTION: Is Φ satisfiable?

The following NP-complete problem was introduced and studied in an equivalent formulation by Ng, Steel, and Wormald [NSW00].

Problem 2.6 (Subtree-Avoidance-Problem)**.**
INSTANCE: A rooted triple formula Φ where each clause is of the form $x_1y_1 \nmid z_1 \vee \cdots \vee x_ky_k \nmid z_k$. (As in Example 2.2, $xy \nmid z$ stands for $xz|y \vee yz|x$.)
QUESTION: Is Φ satisfiable?

Also the following problem is NP-hard; it has been studied in [Bry97].

Problem 2.7 (Forbidden-Triple-Consistency)**.**
INSTANCE: A rooted triple formula Φ where each clause is of the form $xy \nmid z$.
QUESTION: Is Φ satisfiable?

More generally, if $\mathcal{C}$ is a class of triple clauses, the rooted phylogeny problem for clauses from $\mathcal{C}$ is the following computational problem.

Problem 2.8 (Rooted-Phylogeny for clauses from $\mathcal{C}$)**.**
INSTANCE: A rooted triple formula Φ where each clause can be obtained from clauses in $\mathcal{C}$ by substitution of variables.
QUESTION: Is Φ satisfiable?

All of these problems belong to NP. A given solution (T, α) can be verified in polynomial time using the following deterministic algorithm. For each literal of each clause of Φ check whether the literal is satisfied. If there is at least one literal per clause satisfied by (T, α), then the given solution is valid else it is invalid. A literal $ab|c$ is satisfied if $\alpha(a)$, $\alpha(b)$, and $\alpha(c)$ are distinct and if $v_1 = yca(\alpha(a), \alpha(b))$ lies strictly below $v_2 = yca(\alpha(a), \alpha(c))$ (recalling definition 2.3). Determining the youngest common ancestor of two vertices is straightforward using a bottom-up search for each vertex. Another search is then used to check if v_1 lies strictly below v_2.

Note that the rooted triple consistency problem, the subtree avoidance problem, and the forbidden triple consistency problem are examples of rooted phylogeny problems, by appropriately choosing the class $\mathcal{C}$. For example, for the rooted triple consistency problem we choose $\mathcal{C} = \{xy|z\}$. The subtree avoidance problem is the rooted phylogeny problem for the class $\mathcal{C}$ that contains for each k the clause $x_1y_1 \nmid z_1 \vee \cdots \vee x_ky_k \nmid z_k$.

Constraint Satisfaction Problems. Many phylogeny problems can be viewed as infinite domain *constraint satisfaction problems* (CSPs), which are defined as follows. Let Γ be a structure[1] with a finite relational signature τ. A first-order formula over τ is called *primitive positive* if it is of the form $\exists x_1, \ldots, x_n. \psi_1 \wedge \cdots \wedge \psi_m$ where $\psi_1, \ldots, \psi_m$ are atomic formulas over τ, i.e., of the form $x = y$ or $R(x_1, \ldots, x_k)$ for a k-ary $R \in \tau$. Then the *constraint satisfaction problem for* Γ, denoted by CSP(Γ), is the computational problem to decide whether a given primitive positive sentence (i.e., a primitive positive formula without free variables) is true in Γ. The sentence Φ is also called an *instance* of CSP(Γ), and the clauses of Φ are also called the *constraints* of Φ. We cannot give a full introduction to constraint satisfaction and to constraint satisfaction on infinite domains, but point the reader to [BJK05, Bod08]. Here, we only specify an infinite structure Δ that can be used to describe the rooted triple consistency problem as a constraint satisfaction problem. It will then be straightforward to see that all rooted phylogeny problems for clauses from a finite class $\mathcal{C}$ can be formulated as infinite domain CSPs as well.

The signature of Δ is $\{|\}$ where $|$ is a ternary relation symbol. The domain of Δ is $\mathbb{N} \to \{0, 1\}$, i.e., the set of all infinite binary strings (hence, the domain of Δ is uncountable). For two elements f, g of Δ, let $\mathrm{lcp}(f, g)$ be the set $\{1, \ldots, n\}$ where n is the largest natural number i such that $f(j) = g(j)$ for all $j \in \{1, \ldots, i\}$; if no such i exists, we set $\mathrm{lcp}(f, g) := \emptyset$, and if $f = g$, we set $\mathrm{lcp}(f, g) := \mathbb{N}$. The ternary relation $fg|h$ in Δ holds on elements f, g, h of Δ if they are pairwise distinct and $|\mathrm{lcp}(f, g)| > |\mathrm{lcp}(f, h)|$.

The following lemma shows that instances of the rooted triple consistency problem can be viewed as primitive positive formulas over the signature $\{|\}$.

Proposition 2.9. *A rooted triple formula $\Phi(x_1, \ldots, x_n)$ is satisfiable if and only if the sentence $\exists x_1, \ldots, x_n. \Phi(x_1, \ldots, x_n)$ is true in Δ.*

Proof. Suppose that $\exists x_1, \ldots, x_n. \Phi(x_1, \ldots, x_n)$ is true in Δ, and let $f_1, \ldots, f_n : \mathbb{N} \to \{0, 1\}$ be witnesses for $x_1, \ldots, x_n$ that satisfy Φ in Δ. We define a finite rooted tree T as follows. The vertex set of T consists of the restrictions of f_i to $\mathrm{lcp}(f_i, f_j)$ for all $1 \leq i, j \leq n$ (we do not require i and j to be distinct). Vertex g is above vertex g' in T if g' extends g; it is clear that this describes T uniquely. Note that $f_1, \ldots, f_n$ are exactly the leaves of T, and that T is binary. Let α be the map that sends x_i to f_i. Then (T, α) satisfies Φ.

Conversely, let (T, α) be a solution to Φ. For each vertex v of T that is not a leaf, let $l(v)$ and $r(v)$ be the two neighbors of v in T that have larger distance from the root than v. Let h be the length of the path $r = p_1, \ldots, p_h = \alpha(x_i)$ from the root r to $\alpha(x_i)$ in T. Define $f_i : \mathbb{N} \to \{0, 1\}$ by setting $f_i(j) = 0$ if $p_{j+1} = l(p_j), 1 \leq j < h$, and $f_i(j) = 1$ otherwise. Clearly, the elements $f_1, \ldots, f_n$ of Δ show that $\exists x_1, \ldots, x_n. \Phi(x_1, \ldots, x_n)$ is true in Δ. □

This shows that the rooted triple consistency problem is indeed a constraint satisfaction problem. A refined version of this observation will be useful in Section 3 to apply known techniques for proving Datalog inexpressibility of the rooted triple consistency problem.

A triple clause is called *trivial* if the clause is satisfied by any injective mapping from the variables into the leaves of any rooted tree, or if it is unsatisfiable (and otherwise we call the clause *non-trivial*). The following lemma (Lemma 2.10) shows that the rooted triple consistency problem is among the simplest rooted phylogeny problems, that is, for every class $\mathcal{C}$ that contains a non-trivial triple clause the rooted phylogeny problem for $\mathcal{C}$ can simulate the rooted triple consistency problem in a simple way.

[1] We follow standard terminology in logic, see e.g. [Hod97].

Lemma 2.10. *Let $\phi(x_1, \dots, x_k)$ be a non-trivial triple clause. Then there are variables $y^1_1, \dots, y^1_k, \dots, y^l_1, \dots, y^l_k \in \{a, b, c\}$ such that $\bigwedge_{i=1..l} \phi(y^i_1, \dots, y^i_k)$ is logically equivalent to $ab|c$.*

Proof. First observe that if $k = 3$ and if $\phi(x_1, x_2, x_3)$ contains only one literal then renaming its variables is trivial. Second, if $\phi(x_1, x_2, x_3) = x_1x_3|x_2 \vee x_2x_3|x_1$, then $\phi(a, c, b) \wedge \phi(b, c, a)$ is logically equivalent to $ab|c$. If $\phi(x_1, x_2, x_3)$ contains three or more literals, then due to its non-triviality there can only be at most two distinct literals. Thus, we fall back to one of the already shown cases and the claim follows for all clauses with exactly three variables.

If $k > 3$, then non-triviality of ϕ implies that $\phi(x_1, \dots, x_k)$ can be written as $x_{i_1}x_{i_2}|x_{i_3} \vee \phi'(x_1, \dots, x_k)$ for distinct variables $x_{i_1}, x_{i_2}, x_{i_3}$ such that ϕ' does not imply $x_{i_1}x_{i_2} \nmid x_{i_3}$, or as $x_{i_1}x_{i_2} \nmid x_{i_3} \vee \phi'(x_1, \dots, x_k)$ for distinct variables $x_{i_1}, x_{i_2}, x_{i_3}$ such that ϕ' does not imply $x_{i_1}x_{i_2}|x_{i_3}$. In both cases we can falsify all literals in ϕ' that contain a variable x_{i_4} distinct from $x_{i_1}, x_{i_2}, x_{i_3}$ by making x_{i_4} equal to some other variable in this literal. The claim then follows from the case $k = 3$. □

This implies that the Datalog inexpressibility result for the rooted triple consistency problem we present in the next section applies to all the rooted phylogeny problems for clauses from $\mathcal{C}$ that contain a non-trivial clause.

3. Datalog

Datalog is an important algorithmic concept originating both in logic programming and in database theory [AHV95, EF99, Imm98]. Feder and Vardi [FV99] observed that Datalog programs can be used to formalize efficient constraint propagation algorithms used in Artificial Intelligence [All83, Mon74, Dec92, Mac77]. Such algorithms have also been studied for the phylogenetic reconstruction problem. Dekker [Dek86] studied *rules* that infer rooted triples from given sets of rooted triples, and asked whether there exists a set of rules such that a rooted triple can be derived by these rules from a set of rooted triples Φ if and only if it is logically implied by Φ. This question was answered negatively by Bryant and Steel [BS95].

In this section, we show the stronger result that the rooted triple consistency problem cannot be solved by Datalog. This is a considerable strengthening of this previous result by Bryant and Steel, since we can use Datalog programs not only to infer rooted triples that are implied by other rooted triples, but rather might use Datalog rules to infer an arbitrary number of relations (aka *IDBs*) of arbitrary arity to solve the problem. Moreover, we only require that the Datalog program derives false if and only if the instance is unsatisfiable. In particular, we do not require that the Datalog program derives every rooted triple that is logically implied by the instance (which is required for the question posed by Dekker). Finally, as already announced in the conference version of this paper, we show that the proof technique extends to other constraint formalisms for reasoning about trees.

In our proof, we use a pebble-game that was introduced to describe the expressive power of Datalog [KV95] and which was later used to study Datalog as a tool for finite domain constraint satisfaction problems [FV99]. The correspondence between Datalog and pebble games extends to infinite domain constraint satisfaction problems for countably infinite ω-categorical structures. A countably infinite structure is called *ω-categorical* if its first-order theory[2] has exactly one countable model up to isomorphism. It can be seen (e.g.

[2]The *first-order theory* of a structure is the set of first-order sentences that are true in the structure.

using the theorem of Ryll-Nardzewski, see [Hod97]) that the structure Δ introduced in Section 2 is, unfortunately, *not* ω-categorical. However, there are several ways of defining an ω-categorical structure Λ (described also in [Cam90]) which has the same constraint satisfaction problem.

We exactly follow the axiomatic approach to define such a structure Λ given in [AN98]. A ternary relation C is said to be a *C-relation* on a set L if for all $a, b, c, d \in L$ the following conditions hold:

(C1) $C(a; b, c) \rightarrow C(a; c, b)$;
(C2) $C(a; b, c) \rightarrow \neg C(b; a, c)$;
(C3) $C(a; b, c) \rightarrow C(a; d, c) \vee C(d; b, c)$;
(C4) $a \neq b \rightarrow C(a; b, b)$.

A C-relation is called *dense* if it satisfies

(C7) $C(a; b, c) \rightarrow \exists e.\ (C(e; b, c) \wedge C(a; b, e))$.

The structure $(L; C)$ is also called a *C-set*.

A structure Γ is called *k-transitive* if for any two k-tuples $(a_1, \ldots, a_k)$ and $(b_1, \ldots, b_k)$ of distinct elements of Γ there is an automorphism[3] of Γ that maps a_i to b_i for all $i \leq k$. A structure Γ is said to be *relatively k-transitive* if for every partial isomorphism f between induced substructures of Γ of size k there exists an automorphism of Γ that extends f. Note that a relatively 3-transitive C-set is necessarily 2-transitive.

Theorem 3.1 (Theorem 14.7 in [AN98])**.** *Let $(L; C)$ be a relatively 3-transitive C-set. Then $(L; C)$ is ω-categorical.*

Theorem 11.2 and 11.3 in [AN98] show how to construct such a C-relation from a *semi-linear order*[4] that is *dense*, *normal*, and *branches everywhere* (all these concepts are defined in [AN98]). Such a semi-linear order is explicitly constructed in Section 5 of [AN98].

In fact, there is, up to isomorphism, a unique relatively 3-transitive countable C-set which

- is *uniform with branching number* 2, that is, if for all $a, b, c \in L$ we have $C(a; b, c) \vee C(b; c, a) \vee C(c; a, b)$,
- is dense, and
- satisfies $\neg C(a; a, a)$ for all $a \in L$.

(See the comments in [AN98] after the statement of Theorem 14.7; the condition that $\neg C(a; a, a)$ for all (equivalently, for some) $a \in L$ has been forgotten there, but is necessary to obtain uniqueness.)

In the following, let Λ be the structure whose domain is the domain of the dense C-set that is uniform with branching number 2; the signature of Λ is not the C-relation, but the relation $xy|z$ defined from the C-relation by

$$xy|z \;\Leftrightarrow\; C(z; x, y) \wedge x \neq y \wedge y \neq z \wedge x \neq z\ .$$

It is clear that Λ has the same automorphism group than the dense C-set from which it is defined, and in particular, it is also ω-categorical (since by the Theorem of Ryll-Nardzewski, ω-categoricity of a structure only depends on the automorphism group of Γ, see [Hod97]). Note that the relation $|$ of Λ satisfies $(C1)$, $(C2)$, $(C3)$, but not $(C4)$.

[3] An automorphism of a structure Γ is an isomorphism between Γ and itself.

[4] A poset is connected if for any two a, b there exists a c such that $a \leq c$ and $b \leq c$, or $a \geq c$ and $b \geq c$. A connected poset is called *semi-linear* if for every point, the set of all points above it is linearly ordered.

The following observation has already been made in [Bod08], but without proof, so we provide a proof here.

Proposition 3.2. *A rooted triple formula $\Phi(x_1, \ldots, x_n)$ is satisfiable if and only if the sentence $\exists x_1, \ldots, x_n.\ \Phi(x_1, \ldots, x_n)$ is true in Λ.*

Proof. Suppose that there are $a_1, \ldots, a_n$ such that $\Phi(a_1, \ldots, a_n)$ is true in Λ. We first define a binary relation $\preceq$ on the set of all pairs (a, b) with $a, b \in \{a_1, \ldots, a_n\}$. We set $(a, b) \preceq (c, d)$ if $\neg cd|a \wedge \neg cd|b$, and define $R := \{(u, v) \mid u \preceq v \wedge v \preceq u\}$.

Lemma 3.3 (Lemma 12.1 in [AN98])**.** *The relation $\preceq$ is a preorder, and hence R is an equivalence relation.*

Also the following is taken from [AN98]; but to avoid extensive references into the proofs there, we give a self-contained presentation here. We claim that the poset $\preceq /R$ that is induced by $\preceq$ in the natural way on the equivalence classes of R is semi-linear. To see this, let $(a_1, a_2), (b_1, b_2), (c_1, c_2)$ be such that $(a_1, a_2) \preceq (b_1, b_2)$ and $(a_1, a_2) \preceq (c_1, c_2)$. We have to show that (b_1, b_2) and (c_1, c_2) are comparable in $\preceq$. If $(b_1, b_2) \not\preceq (c_1, c_2)$, then $c_1c_2|b_1$ or $c_1c_2|b_2$. Suppose in the following that $c_1c_2|b_1$; the case $c_1c_2|b_2$ is analogous. Since $(a_1, a_2) \preceq (c_1, c_2)$ we have in particular $\neg c_1c_2|a_1$ in Λ. Recall that the relation $|$ satisfies $(C3)$, which can be equivalently written as $\forall a, b, c.\ (C(a; b, c) \wedge \neg C(d; b, c)) \rightarrow C(a; d, c)$, so we find that $a_1c_1|b_1$. By (C2) we have $\neg a_1b_1|c_1$. Since $(a_1, a_2) \preceq (b_1, b_2)$ we have $\neg b_1b_2|a_1$. Axiom (C3) can also be written as $\forall a, b, c.\ (\neg C(a; d, c) \wedge \neg C(d; b, c)) \rightarrow \neg C(a; b, c)$, and thus $\neg b_1b_2|c_1$. Similarly, $\neg b_1b_2|c_2$. Therefore, $(c_1, c_2) \preceq (b_1, b_2)$, which is what we had to show.

Next, note that when (d_1, d_2) and (e_1, e_2) are incomparable with respect to $\preceq$, then (d_1, e_1) is an upper bound for (d_1, d_2) and (e_1, e_2), that is, $(d_1, d_2) \preceq (d_1, e_1)$ and $(e_1, e_2) \preceq (d_1, e_1)$. It follows that $\preceq /R$ is indeed a semi-linear order with a smallest element r, and there exists a tree T on the equivalence classes of R such that p lies below q in T if for all (equivalently, for some) $(a, b) \in p$ and $(c, d) \in q$ we have $(c, d) \preceq (a, b)$. Let α be the map that sends x_i to the equivalence class of (a_i, a_i); it is straightforward to verify that (T, α) satisfies Φ.

Conversely, let (T, α) be a solution to Φ. We now determine elements $a_1, \ldots, a_n$ from Λ, and prove by induction on i that $\alpha(x_r)\alpha(x_s)|\alpha(x_t)$ in T if and only if $a_ra_s|a_t$ in Λ, for all $r, s, t \leq i$. This is trivial for $n = i = 1$, and for $n = i = 2$ we can choose arbitrary distinct elements a_1 and a_2 from Λ. Now suppose we have already found elements $a_1, \ldots, a_i$ of Λ, for $2 \leq i < n$, that satisfy the inductive hypothesis. Let v be the vertex in T that has the maximal distance from the root of T such that there is an $j \leq i$ where both $\alpha(x_j)$ and $\alpha(x_{i+1})$ lie strictly below v.

First consider the case that v is the root of T. Then we can choose $k, l \in \{1, \ldots, i\}$ such that $v = yca(\alpha(x_k), \alpha(x_l))$. Let a be an element of Λ that is distinct from a_k and a_l, and by the properties of Λ ($xy|z$ is uniform with branching number 2) we have that $a_ka_l|a$, $a_ka|a_l$ or $aa_k|a_l$ holds. In the first case, we set a_{i+1} to a. In the second case, by relative 3-transitivity of Λ there exists an automorphism β of Λ that maps a_k to a_l and that fixes a. In this case we set a_{i+1} to $\beta(a_l)$. In the third case we proceed similar to the second. In all three cases we have $a_pa_q|a_{i+1}$ for all $p, q \leq i$, which proves the inductive step.

Next, consider the case that v is not the root of T. In this case, there must be an $m \leq i$ such that $\alpha(x_j)\alpha(x_{i+1})|\alpha(x_m)$; choose m such that the distance between the root and $yca(\alpha(x_j), \alpha(x_m))$ is maximal. When j is the only index of size at most i such that $\alpha(x_j)$ lies below v in T, then density of Λ (axiom C7 in the special case that $b = c$) implies

that there is an a such that $a_j a|a_m$. We can then set a_{i+1} to a. Otherwise, there are $j', j'' \leq i$ such that $\alpha(x_{j'})\alpha(x_{j''})|\alpha(x_{i+1})$; choose j', j'' such that the distance between v and $yca(\alpha(x_{j'}), \alpha(x_{j''}))$ is minimal. Again we apply density (axiom C7) and conclude that there is an a such that $a_{j'}a_{j''}|a$ and $a_{j'}a|a_m$. We can then set a_{i+1} to a. □

The Existential Pebble Game. The fact that Λ is ω-categorical allows us to use the existential k-pebble game to establish the Datalog lower bound for the rooted triple consistency problem [BD08].

The existential k-pebble game (for a structure Γ) is played by the players *Spoiler* and *Duplicator* on an instance Φ of CSP(Γ) and Γ. Each player has k pebbles, $p_1, \ldots, p_k$ for Spoiler and $q_1, \ldots, q_k$ for Duplicator; we say that that q_i *corresponds* to p_i. Spoiler places his pebbles on the variables of Φ, Duplicator her pebbles on elements of Γ. Initially, none of the pebbles is placed. In each round of the game Spoiler picks some of his pebbles. If some of these pebbles are already placed on Φ, then Spoiler removes them from Φ, and Duplicator responds by removing the corresponding pebbles from Γ. Duplicator looses if at some point of the game

- there is a clause $R(x_1, \ldots, x_k)$ in Φ such that $x_1, \ldots, x_k$ are pebbled by $p_{j_1}, \ldots, p_{j_k}$, and
- the corresponding pebbles $q_{j_1}, \ldots, q_{j_k}$ of Duplicator are placed on elements $b_1, \ldots, b_k$ in Γ such that $R(b_1, \ldots, b_k)$ does *not* hold in Γ.

Duplicator wins if the game continues forever. We will make use of the following theorem from [BD08].

Theorem 3.4 (Theorem 5 in [BD08])**.** *Let Γ be an ω-categorical (or finite) structure. Then there is no Datalog program that solves CSP(Γ) if and only if for every k there exists an unsatisfiable instance Φ_k of CSP(Γ) such that Duplicator wins the existential k-pebble game on Φ_k and Γ.*

Our Method. The *incidence graph* $G(\Phi)$ of an instance Φ of CSP(Γ) is the (undirected, simple) bipartite graph whose vertex set is the disjoint union of the variables of Φ and the clauses of Φ. An edge joins a variable a and a clause ϕ of Φ when a appears in ϕ. A *leaf* of Φ is a variable that has degree one in $G(\Phi)$. An instance has *girth* k if the shortest cycle of its incidence graph has $2k$ edges.

Lemma 3.5. *Let Γ be an l-transitive (for $l \geq 1$) ω-categorical (or finite) structure with relations of arity at most $l+1$. Suppose that for every k there exists an unsatisfiable instance Φ_k of girth at least k where every constraint has an injective satisfying assignment. Then CSP(Γ) cannot be solved by Datalog.*

We will see examples for $l = 1$ and for $l = 2$ in this paper. Note that by 1-transitivity, every unary relation in Γ either denotes the empty set or the full domain of Γ. Since Φ_k only contains satisfiable constraints, all unary constraints in Φ_k are satisfied by every mapping to Γ. So we make in the following the assumption that Φ_k does *not* contain unary constraints.

In the proof we use the following concept, inspired by a Datalog inexpressibility result that was established for temporal reasoning [BK10].

Definition 3.6. Let Φ be an instance of girth at least $k+1$. Then a subset S of at least 2 and at most k variables of Φ is called *dominated* if $G_S := G(\Phi[S])$ is connected (and hence a tree), and if all but at most one of the leaves of G_S are pebbled.

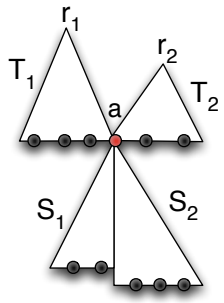


Figure 1: A situation in the proof of Lemma 3.5: Spoiler just pebbled a, Duplicator is next.

The notion of dominated sets allows us to specify a winning strategy for Duplicator for the existential k-pebble game.

Proof of Lemma 3.5. To apply Theorem 3.4, we have to prove that Duplicator wins the existential k-pebble game on Φ_k and Γ.

Suppose that in the course of the game, u is an unpebbled leaf of a dominated set S with pebbled leaves $a_1, \ldots, a_l$, and let $b_1, \ldots, b_l$ be the corresponding responses of Duplicator. Duplicator will play in such a way that $b_1, \ldots, b_l$ are pairwise distinct. Moreover, Duplicator always maintains the following invariant. *Whenever Spoiler places a pebble on a_{l+1}, Duplicator can play a value b_{l+1} from Γ such that the mapping that assigns a_i to b_i for $1 \leq i \leq l+1$ can be extended to all of S such that this extension is a satisfying assignment for $\Phi_k[S]$.*

The invariant is satisfied at the beginning of the game: when spoiler places a pebble on a_1, Duplicator can play any value b_1, which is a legal move by our assumption that Φ_k does not contain unary constraints.

Suppose that during the game Spoiler pebbles a variable a. Let $S_1, \ldots, S_p$ be the dominated sets where a is the unpebbled leaf before Spoiler puts his pebble on a. (If there is no such dominated set, then $p = 0$.) Let $T_1, \ldots, T_q$ be the newly created dominated sets after Spoiler put his pebble on a. Note that since each T_i has not been a dominated set before Spoiler put his pebble on a, it must contain one unpebbled leaf distinct from a, which we denote by r_i. For an illustration, see Figure 1.

We have to show that under the assumption that Duplicator in her previous moves has always maintained the invariant, she will be able to make a move that again fulfills the invariant. If $p > 0$, then the union S of the sets $S_1, \ldots, S_p$ was itself a dominated set already before Spoiler played on a, since G_S is clearly connected (all the S_i share the vertex a) and no unpebbled leaves can be created by taking a union of dominated sets. The next move of Duplicator is the value b from the invariant applied to S. This preserves the invariant, since for every $i \leq q$, the set $T_i \cup S$ has been a dominated set already before Spoiler played on a: because T_i and S share the vertex a, the graph $G_{S \cup T_i}$ is connected, and since a is not a leaf in $G_{S \cup T_i}$, the only unpebbled leaf of $G_{S \cup T_i}$ is r_i. Therefore, α can be extended to all of T_i.

If $p = 0$, Duplicator plays an arbitrary element b in Γ. We prove by induction on the size of T_i that α can be extended to T_i such that $\alpha(a) = b$. We can assume that only leaves in G_{T_i} are pebbled (otherwise, since G_{T_i} is a tree, the task reduces to proving the statement

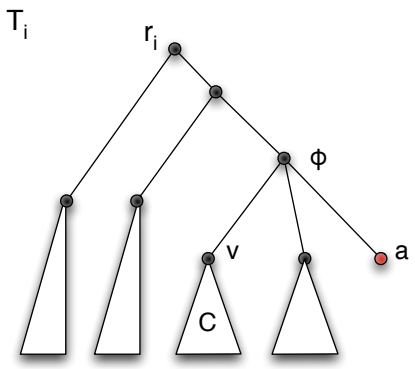


Figure 2: A situation in the proof of Lemma 3.5: extending α to all of T_i.

for proper subsets of T_i). Consider a clause ϕ of $\Phi_k[T_i]$ that contains a, and let V be the variables of ϕ. This clause must be unique: otherwise, the graph obtained from $G(\Phi_k)$ by removing the vertex a has at least two components. Only one of those components can contain r_i; the other component must then be a dominated set where all leaves are pebbled, a contradiction to the assumption that $p = 0$. Now consider the graph H obtained from G_{T_i} by removing the vertex that corresponds to ϕ. See Figure 2.

If one of the connected components of H, say C, forms a dominated set, then the unique variable v in $C \cap V$ (uniqueness again follows from the fact that G_{T_i} is a tree) is the unique unpebbled leaf of C, and by the invariant of Duplicator's strategy α can be extended to α' that is defined on all of C such that it satisfies $\Phi_k[C]$. Hence, by removing the pebbles from C and adding a pebble on v, with $\alpha'(v)$ the corresponding response of Duplicator, we can apply the inductive assumption to $T_i \setminus C \cup \{v\}$ to find an extension of α that is a satisfying assignment for $\Phi_k[T_i]$ and maps a to b.

Otherwise, all variables in V except for the variable that lies in the connected component of r_i in H are pebbled. By our assumption on the signature, the clause ϕ contains at most l pebbled variables (including a). Also by assumption there exists an injective mapping $\beta : V \to \Gamma$ that satisfies ϕ. Since Γ is l-transitive, there is an automorphism γ of Γ that maps $\beta(a)$ to b and that sends $\beta(w)$ to $\alpha(w)$, for $w \in T_i \setminus \{v\}$. Then we extend α to v by $\alpha(v) := \gamma(\beta(v))$; the extension clearly satisfies ϕ. Now we repeat the argument with v in place of a, and $\alpha(v)$ in place of b, and are done by inductive assumption. □

Application to the Rooted Phylogeny Problem. We now turn back to the rooted triple consistency problem, CSP(Λ). The structure Λ is 2-transitive and the only relation has arity three, and hence we can apply Lemma 3.5 to prove that CSP(Λ) cannot be solved by Datalog.

To construct an unsatisfiable girth k instance Φ_k for CSP(Λ), let G be a cubic graph of girth at least k that has a Hamiltonian cycle. Such a graph exists; see e.g. the comments after the proof of Theorem 3.2 in [Big98]. Note that G must have an even number of vertices. Let $H = (v_1, v_2, \ldots, v_n)$ be the Hamilton cycle of G. For any vertex a of G, let $r(a)$ be the vertex that precedes a on H, $s(a)$ the vertex that follows a on H, and $t(a)$ the third remaining neighbor of a in G.

We now define Φ_k. The vertices of G will be the variables of Φ_k. Then

$$\Phi_k := \bigwedge_{a \in V(G)} r(a)s(a)|t(a) \; .$$

Consider the graph on the variables of Φ_k that has an edge ab when Φ_k contains a triple clause $ab|c$ for some variable c of Φ_k. This graph is connected, since it actually equals the Hamilton cycle H of G. Hence, a condition due to Aho et al. [ASSU81] implies that Φ_k is unsatisfiable for all $k \geq 1$. This can also be seen by Lemma 4.3 in Section 4. It is clear that every triple clause of Φ_k has an injective satisfying assignment. So the only remaining condition to apply Lemma 3.5 is the verification that $G(\Phi_k)$ has girth k. But this is obvious since any cycle of length $2l < 2k$ in the incidence graph $G(\Phi_k)$ would give rise to a cycle of length $l < k$ in G, in contradiction to G having girth k.

Corollary 3.7. *There is no Datalog program that solves the rooted triple consistency problem.*

Other Applications of the Technique. Our technique to show Datalog inexpressibility can be adapted to show that the following (closely related) problems cannot be solved by Datalog as well.

- Satisfiability of branching time constraints [BJ03];
- The network consistency problem of the left-linear-point algebra [Due05, Hir97];
- Cornell's tree description logic [Cor94, BK07];

All these three problems contain the following computational problem as a special case.

Problem 3.8 (Tree-Description-Consistency)**.**
INSTANCE: A finite structure $(V; <, ||)$ where $<$ and $||$ are binary relations.
QUESTION: Is there a rooted tree T and $\alpha : V \to V(T)$ such that if $x < y$ then $\alpha(y)$ lies strictly below $\alpha(x)$ in T, and if $x||y$ then neither $\alpha(x)$ lies below $\alpha(y)$ nor $\alpha(y)$ lies below $\alpha(x)$ in T?

To again apply Lemma 3.5, we first have to show that Tree-Description-Consistency can be formulated as a CSP for a transitive ω-categorical structure $\Omega = (D; <, ||)$; this has already been observed in [BN06]. This time, it is more convenient to directly construct Ω. The domain D consists of the set of all non-empty finite sequences of rational numbers. For $a = (q_1, q_2, \ldots, q_n), b = (q'_1, q'_2, \ldots, q'_m)$, $n \leq m$, we write $a < b$ if one of the following conditions holds:

- a is a proper initial subsequence of b, i.e., $n < m$ and $q_i = q'_i$ for $1 \leq i \leq n$;
- $q_i = q'_i$ for $1 \leq i < n$, and $q_n < q'_n$.

The relation $||$ is the set of all unordered pairs of distinct elements that are incomparable with respect to $<$. A proof that Ω is indeed 1-transitive and ω-categorical can be found in [AN98] (Section 5). Since the signature is binary, we can again apply Lemma 3.5, and have to find unsatisfiable instances of arbitrarily high girth.

Here we use the fact that Tree-Description-Consistency can simulate the rooted triple consistency problem by a simple reduction [BK07]. We construct Ψ_k from Φ_k by replacing each triple clause of Φ_k of the form $xy|z$ by the three conjuncts $u_{xyz}||z$, $u_{xyz} < x$, and $u_{xyz} < y$, where u_{xyz} is a newly introduced variable. It can be shown (see [BK07]) that this transformation preserves (un-)satisfiability, and thus Ψ_k is unsatisfiable as well. Moreover,

the transformation is such that the girth of Ψ_k is not smaller than the girth of Φ_k. Finally, it is clear that every conjunct in Ψ_k has an injective satisfying assignment. Hence, Lemma 3.5 applies, and CSP(Ω) cannot be solved by Datalog.

4. The Algorithm

In this section we show that the rooted phylogeny problem can be solved in polynomial time if all clauses come from the following class $\mathcal{T}$, defined as follows.

Definition 4.1. A disjunction $\psi := x_1y_1|z_1 \vee \cdots \vee x_py_p|z_p$ is called *tame* if it is trivial or if $\{x_i, y_i\} = \{x_j, y_j\}$ for all $1 \leq i, j \leq p$. The set of all tame clauses is denoted by $\mathcal{T}$.

The algorithm we present in this section builds on previous algorithmic results about the rooted triple consistency problem, most notably [ASSU81, HKW96]. One of the central ideas for the polynomial-time algorithm for the rooted triple consistency problem in [ASSU81] is to associate a certain undirected graph to an instance of the rooted triple consistency problem. We generalize this idea to tame clauses as follows.

Definition 4.2. Let Φ be an instance of the rooted triple consistency problem with tame clauses. Then $F_\Phi := (V, E)$ is the graph where the vertex set V is the set of variables of Φ, and where E contains an edge $\{x, y\}$ iff Φ contains a non-trivial clause $xy|z_1 \vee \cdots \vee xy|z_p$ for $p \geq 1$.

The following provides a sufficient (but not a necessary) condition for unsatisfiability of rooted triple formulas with tame clauses.

Lemma 4.3. *Let Φ be an instance of the rooted phylogeny problem with tame clauses. If F_Φ is connected then Φ is unsatisfiable.*

Proof. Let V be the set of variables in Φ. Suppose that there is a solution (T, α) for Φ. Let r be the yca of $\alpha(V)$ in T (where $\alpha(V)$ is the set of all leaves in the image of V under α). It cannot be that all vertices in $\alpha(V)$ lie below the same child of r in T, since otherwise the child would have been above $r = yca(\alpha(V))$, which is impossible. Since the graph F_Φ is connected, there is an edge $\{x, y\}$ in F_Φ such that $\alpha(x)$ and $\alpha(y)$ lie below different children of r in T. Hence, there are $z_1, \ldots, z_p \in V$ and a clause $xy|z_1 \vee \cdots \vee xy|z_p$ in Φ. By assumption, the yca of $\alpha(x)$ and $\alpha(y)$, which is r, lies strictly below the yca of $\alpha(x)$ and $\alpha(z_i)$ for some $1 \leq i \leq p$, a contradiction to the choice of r. □

To see that the condition is not necessary consider the following example.

Example 4.4. The rooted triple formula $\Phi = (ab|c \wedge bc|a \wedge ab|d)$ is unsatisfiable since the first two literals cannot simultaneously be satisfied. But the graph F_Φ is disconnected; it has the two components $\{a, b, c\}$ and $\{d\}$.

Theorem 4.5. *The algorithm Solve in Figure 3 determines whether a given instance Φ of the rooted phylogeny problem for tame clauses is satisfiable. When m is the number of triples in all clauses, and n is the number of variables of Φ, then the algorithm can be implemented to run in time $O(m \log^2 n)$.*

Proof. If Φ is the empty conjunction, then Φ is clearly satisfiable, and so the answer of the algorithm is correct in this case. The algorithm first computes a connected component S of F_Φ (we discuss details of this step in the paragraph about the running time of the algorithm); if $S = V$, i.e., if F_Φ is connected, then Lemma 4.3 implies that Φ is unsatisfiable.

```
Solve(Φ)
Input: A rooted triple formula Φ with variables V and clauses from 𝒯.
Output: true if Φ is satisfiable, false otherwise.

If Φ is the empty conjunction then return 'true'
If F_Φ is connected
    return 'false'
else
    Let S be the vertices of a connected component of F_Φ
    If Solve(Φ[S]) is false or Solve(Φ[V \ S]) is false return 'false'
    else return 'true'
    end if
end if
```

Figure 3: The algorithm for the rooted phylogeny problem for tame clauses.

Otherwise, we execute the algorithm recursively on $\Phi[S]$ and on $\Phi[V \setminus S]$. If any of these recursive calls reports an inconsistency, then Φ is clearly unsatisfiable as well: since if there was a solution (T, α) to Φ, then $(T, \alpha|_V)$ would be a solution to $\Phi[V]$. Otherwise, we inductively assume that the algorithm correctly asserts the existence of a solution (T_1, α_1) of $\Phi[S]$ and of a solution (T_2, α_2) of $\Phi[V \setminus S]$.

Let T be the tree obtained by creating a new vertex r, linking the roots of T_1 and T_2 below r, and making r the root of T. Let α be the mapping that maps x to $\alpha_i(x)$ if $x \in L(T_i)$, for $i \in \{1, 2\}$. We claim that (T, α) is a solution to Φ, i.e., we have to show that in every clause ψ of Φ at least one literal is satisfied. If $\psi = (xy|z_1 \vee \cdots \vee xy|z_p)$, then x and y are in the same subtree T_i of T, since they are connected by an edge in F_Φ. If all variables of ψ lie completely inside S or completely inside $V \setminus S$, we are done by inductive assumption, because (T_1, α_1) is a solution for $\Phi[S]$ and (T_2, α_2) is a solution for $\Phi[V \setminus S]$. Otherwise, there must be a j, $1 \leq j \leq p$, such that z_j lies in a different component than x and y. But in this case the yca of $\alpha(x)$ and $\alpha(y)$ lies strictly below r, which is the yca of $\alpha(x)$ and $\alpha(z_j)$. Hence, the literal $xy|z_j$ in ψ is satisfied. This concludes the correctness proof of the algorithm shown in Figure 3.

We still have to show how this procedure can be implemented such that the running time is in $O(m \log^2 n)$. There are amortized sub linear algorithms for testing connectivity in undirected graphs while removing the edges of the graph. This was used to speed-up the algorithm for the rooted triple consistency problem [HKW96]. At present, the fastest known algorithm for this purpose appears to be the deterministic decremental graph connectivity algorithm of Holm, de Lichtenberg, and Thorup [THdL98], which has a query time in $O(\log n / \log \log n)$, and an update time in $O(\log^2 n)$. We can use the same approach as in [HKW96] and obtain an $O(m \log^2 n)$ bound for the worst-case running time of our algorithm. □

5. Complexity Classification

This section is devoted to the proof of the following result.

Theorem 5.1. *Let $\mathcal{C}$ be a set of rooted triple clauses that contains clauses that are not tame (Definition 4.1). Then the rooted phylogeny problem for clauses from $\mathcal{C}$ is NP-complete.*

Our proof of Theorem 5.1 consists of two parts. In the first part, we show that if $\mathcal{C}$ is not a subset of $\mathcal{T}$, then a certain Boolean split problem associated to $\mathcal{C}$ (defined below) is NP-hard. In the second part we show that this Boolean split problem reduces to the rooted phylogeny problem for $\mathcal{C}$.

Definition 5.2 (split formula for Φ)**.** Let Φ be a rooted triple formula. Then the *split formula* for Φ is the Boolean formula obtained from Φ by replacing positive literals of the form $xy|z$ by $(x \leftrightarrow y) \wedge (z \vee \neg z)$.

The purpose of the tautological second conjunct $z \vee z$ is to introduce the variable z, which would otherwise not appear in the formula; this becomes relevant in the following. If $\mathcal{C}$ is a class of triple clauses, we define $B(\mathcal{C})$ to be the set of split formulas for clauses from $\mathcal{C}$. A solution to a propositional formula is called *non-trivial* if at least one variable is set to true and at least one variable is set to false. The *split problem* for a set of Boolean formulas $\mathcal{B}$ is the problem to decide whether a given conjunction of formulas obtained from formulas in $\mathcal{B}$ by variable substitution has a non-trivial solution.

We will show that if $\mathcal{C}$ is a class of triple clauses that is not a subclass of $\mathcal{T}$, then there exists a finite subset $\mathcal{C}'$ of $\mathcal{C}$ such that the split problem for $B(\mathcal{C}')$ is NP-complete. In the proof of this statement we use the following result, which follows from Theorem 6.12 in [CKS01], and is due to [CH97]. The notion of Horn, dual Horn, affine, and bijunctive Boolean formulas are standard and introduced in detail in [CKS01]. Bijunctive formulas are also known as 2-CNF formulas.

Theorem 5.3 (of [CH97])**.** *Let $\mathcal{B}$ be a set of Boolean formulas. Then the split problem for $\mathcal{B}$ is in P if all formulas in $\mathcal{B}$ are from one of the following types: Horn, dual Horn, affine, bijunctive. In all other cases, $\mathcal{B}$ contains a finite subset $\mathcal{B}'$ such that the split problem for $\mathcal{B}'$ is NP-complete.*

We say that a Boolean formula ψ is *preserved* by an operation $f : \{0,1\}^k \to \{0,1\}$ if for all satisfying assignments $\alpha_1, \ldots, \alpha_k$ of ψ the mapping defined by $x \mapsto f(\alpha_1(x), \ldots, \alpha_k(x))$ is also a satisfying assignment for ψ.

Proposition 5.4. *If $\mathcal{C}$ is not a subclass of $\mathcal{T}$, then $B(\mathcal{C})$ is neither Horn, dual Horn, affine, nor bijunctive.*

Proof. Let ϕ be a clause from $\mathcal{C} \setminus \mathcal{T}$. By construction the split formula ψ for ϕ is preserved by $x \mapsto \neg x$ and is also preserved by constant operations (both trivial assignments satisfy ψ). Moreover, it is known (and follows from [Pos41]) that every Boolean formula that is preserved by $\neg$, contains the constants, and is either Horn, dual Horn, affine, or bijunctive must also be preserved by the operation xor defined as $(x, y) \mapsto (x + y \mod 2)$. So it suffices to show that ψ cannot be preserved by xor.

Because ϕ is not from $\mathcal{T}$ and in particular non-trivial, there is a tree T and an injective mapping from the variables V of ϕ to the leaves of T such that (T, α) is not a solution to ϕ. Moreover, since the clause ϕ is not tame, it must contain triples $ab|c$ and $uv|z$ where $\{a, b\} \neq \{u, v\}$.

Consider the assignment β that maps $x \in V$ to 0 if $\alpha(x)$ is below the first child of the yca of $\alpha(V)$ in T, and that maps x to 1 otherwise (which child is selected as the first child is not important in the proof). By construction, the assignment β does not satisfy the split

formula for ψ, since ϕ is not satisfied by (T, α). Observe that the assignment β_1 that is obtained from β by negating the value assigned to a is a satisfying assignment for ψ, since it satisfies the disjunct $((a \leftrightarrow b) \wedge (c \vee \neg c))$ of ψ. The assignment β_2 that is constant 0 except for the variable a which is assigned 1 is also a satisfying assignment for ψ, because ψ satisfies $((u \leftrightarrow v) \wedge (w \vee \neg w))$. But since $\mathrm{xor}(\beta_1(x), \beta_2(x))$ equals $\beta(x)$ for all $x \in V$, this shows that ψ is not preserved by xor, which is what we wanted to show. □

We now turn to the second part of the proof of Theorem 5.1. The idea to reduce the split problem for $B(\mathcal{C})$ to the rooted phylogeny problem for clauses from $\mathcal{C}$ is to construct instances Φ of the phylogeny problem for $\mathcal{C}$ in such a way that Φ is satisfiable *if and only if* $B(\Phi)$ has a non-trivial solution. To implement this idea, we construct an instance of the phylogeny problem Φ that fragments into simple and satisfiable pieces already after the first step of the recursion of the algorithm from Section 4.

Proposition 5.5. *Let $\mathcal{C}$ be a finite class of triple clauses. Then the split problem for $B(\mathcal{C})$ can be reduced in polynomial time to the rooted phylogeny problem for clauses from $\mathcal{C}$.*

Proof. Recall that a triple clause is called trivial if the clause is satisfied by any injective mapping from the variables into the leaves of any rooted tree, or if it is unsatisfiable. If all clauses in $\mathcal{C}$ are trivial, then the split problem for $B(\mathcal{C})$ is clearly in P. Otherwise, we can assume that $\mathcal{C}$ contains the clause that just consists of $ab|c$ since this clause can be simulated by non-trivial clauses from $\mathcal{C}$ by appropriately equating variables (Lemma 2.10).

Suppose we are given an instance of the split problem for $B(\mathcal{C})$ with clauses $\psi_1, \dots, \psi_m$ and variables $V = \{x_0, \dots, x_{n-1}\}$. We create an instance Φ of the rooted phylogeny problem for $\mathcal{C}$ as follows. The variables V' of Φ are triples (x, i, j) where $x \in V$, $i \in \{0, \dots, m-1\}$, and $j \in \{1, \dots, n-1\}$. In the following, all indices of variables from V are modulo n. Moreover, if $m > 1$ we will also write (x, i, n) for $(x, i+1, 1)$ for all $i \in \{0, \dots, m-2\}$. The clauses of Φ consist of two groups.

- To define the first group of clauses, suppose that ψ_i has variables $y_1, \dots, y_q$. Let $\phi_i(y_1, \dots, y_q)$ be the triple clause that defines the Boolean relation from $B(\mathcal{C})$ used in $\psi_i(y_1, \dots, y_q)$. By the assumption that $\mathcal{C}$ and $B(\mathcal{C})$ are finite it is clear that ϕ_i can be computed efficiently (in constant time). We then add the clause $\phi_i((y_1, i, 1), \dots, (y_q, i, 1))$ to Φ.
- The second group of clauses in Φ has for all $x_s \in V$, $i \in \{0, \dots, m-2\}$ (if $m = 1$ the second group of clauses is empty), and $j \in \{1, \dots, n-1\}$ the clause

$$(x_s, i, j)(x_s, i, j+1)|(x_{s+j}, i, 1) \; .$$

Note that the second group of clauses Ψ only consists of rooted triples, and that the graph F_Ψ consists of exactly n paths of length $(n-1)(m-1)$.

We claim that Φ is satisfiable if and only if $\psi_1 \wedge \cdots \wedge \psi_m$ has a non-trivial solution. First suppose that Φ has a solution (T, α). Then the variables V' of Φ can be partitioned into the variables that are mapped via α below the left child of $yca(\alpha(V'))$, and the ones mapped below the right child. Note that both parts of the partition are non-empty. Variables (x, i, j) of V' that share the first coordinate are in the same part of the partition due to the clauses in Φ in the second group. Hence, the mapping that sends $x \in V$ to 0 if (x, i, j) is mapped to the first part, and that sends x to 1 otherwise is well-defined, and a non-trivial assignment. It also satisfies all clauses $\psi_1, \dots, \psi_m$, because of the first group of clauses in Φ.

Conversely, suppose that there is a non-trivial solution s for $\psi_1 \wedge \cdots \wedge \psi_m$. Let S be the subset of the variables V of Φ assigned to 0 by s, and consider the instances $\Phi_1 := \Phi[S]$

and $\Phi_2 := \Phi[V \setminus S]$. Since the assignment is non-trivial, there is a variable $x_p \in V$ that is mapped to 1 and a variable $x_q \in V$ that is mapped to 0. Hence, for all $i \in \{0, \ldots, m-1\}$ the clauses $(x_p, i, q-p)(x_p, i, q-p+1)|(x_q, i, 1)$ from the second group are neither in Φ_1 nor in Φ_2, because they contain variables from both parts of the partition. Therefore, any clause from the first group in Φ_1 will be disconnected in $G[\Phi_1]$ from any other clause in the first group in Φ_1. In particular, Φ_1 has a solution (T_1, α_1). The same statements holds for Φ_2; let (T_2, α_2) be a solution for Φ_2. Let T be the rooted tree obtained from T_1 and T_2 by creating a new vertex r, linking the roots of T_1 and T_2 below r, and making r the root of T. Let α be the common extension of α_1 and α_2 to all of V. Then (T, α) is clearly a solution to Φ.

Both groups of clauses together consist of $m + n(n-1)(m-1)$ many clauses, and it is easy to see that the reduction can be implemented in polynomial time. □

We conclude this section with a combination of the results above.

Proof of Theorem 5.1. As mentioned, the rooted phylogeny problem for $\mathcal{C}$ is clearly in NP. Let $\mathcal{C}$ be a class of triple clauses that is not a subset of $\mathcal{T}$. We prove NP-hardness as follows. By Proposition 5.4, $B(\mathcal{C})$ is neither Horn, dual Horn, affine, nor bijunctive. Theorem 5.3 asserts that there exists a finite subset $\mathcal{B}$ of $B(\mathcal{C})$ such that the split problem for $\mathcal{B}$ is NP-hard. This means that there is a subset $\mathcal{C}'$ of $\mathcal{C}$ such that the split problem for $B(\mathcal{C}')$ is NP-hard. Proposition 5.5 shows that the rooted phylogeny problem for clauses from $\mathcal{C}'$ (and hence also for clauses from $\mathcal{C}$) is NP-hard as well. □

6. Concluding Remarks

We have shown that consistency of rooted phylogeny data can be decided in polynomial time when the data consists of tame disjunctions of rooted triples. Our algorithm extends previous algorithmic results about the rooted triple consistency problem, without sacrificing worst-case efficiency. The class $\mathcal{T}$ of tame triple clauses that can be handled efficiently is also motivated by another result of this paper, which states that any set of triple clauses that is not contained in $\mathcal{T}$ has an NP-complete rooted phylogeny problem. Here we use known results about the complexity of surjective Boolean constraint satisfaction problems.

We also show that no Datalog program can solve the rooted triple consistency problem, using a pebble game that captures the expressive power of Datalog for constraint satisfaction problems with infinite ω-categorical structures. In fact, our result follows from a more general result that also applies to many constraint satisfaction problems outside of phylogenetic reconstruction. We show that a constraint satisfaction problem for a structure with a large automorphism group cannot be solved by Datalog if, roughly, for all k there exists a unsatisfiable instance of girth at least k.

The class of phylogeny problems studied in this paper has a natural generalization to a larger class of computational problems, namely problems of the form CSP(Γ) where Γ has a first-order definition in Λ, the ω-categorical relatively 3-transitive C-set introduced in Section 3. This class contains several additional problems that have been studied in phylogenetic reconstruction, for instance the quartet consistency problem [Ste92]. The larger class also contains new problems that can be solved in polynomial time, and where the split problem consists in finding non-trivial solutions to Boolean linear equation systems. A complexity classification for this larger class of computational problems remains open and is left for future research.

Acknowledgement

We would like to thank the reviewers for their helpful comments.

References

[AHV95] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley, 1995.

[All83] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[AN98] Samson Adepoju Adeleke and Peter M. Neumann. *Relations related to betweenness: their structure and automorphisms*, volume 623 of *Memoirs of the AMS 131*. American Mathematical Society, 1998.

[ASSU81] A.V. Aho, Y. Sagiv, T.G. Szymanski, and J.D. Ullman. Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. *SIAM Journal on Computing*, 10(3):405–421, 1981.

[ASY91] F. Afrati, S.S.Comadakis, and M. Yannakakis. On Datalog vs. polynomial time. In *21st ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems*, pages 113–126, 1991.

[BD08] Manuel Bodirsky and Víctor Dalmau. Datalog and constraint satisfaction with infinite templates. *An extended abstract appeared in the proceedings of STACS'06. The full version is available online at arXiv:0809.2386 [cs.LO]*, 2008.

[Big98] Norman Biggs. Constructions for cubic graphs with large girth. *Electronic Journal of Combinatorics*, 5, 1998.

[BJ03] Mathias Broxvall and Peter Jonsson. Point algebras for temporal reasoning: Algorithms and complexity. *Artificial Intelligence*, 149(2):179–220, 2003.

[BJK05] A. Bulatov, P. Jeavons, and A. Krokhin. The complexity of constraint satisfaction: An algebraic approach (a survey paper). *In: Structural Theory of Automata, Semigroups and Universal Algebra (Montreal, 2003), NATO Science Series II: Mathematics, Physics, Chemistry*, 207:181–213, 2005.

[BK07] Manuel Bodirsky and Martin Kutz. Determining the consistency of partial tree descriptions. *Artificial Intelligence*, 171:185–196, 2007.

[BK10] Manuel Bodirsky and Jan Kára. A fast algorithm and Datalog inexpressibility for temporal reasoning. *ACM Transactions on Computational Logic*, 11(3), 2010.

[BN06] Manuel Bodirsky and Jaroslav Nešetřil. Constraint satisfaction with countable homogeneous templates. *Journal of Logic and Computation*, 16(3):359–373, 2006.

[Bod08] Manuel Bodirsky. Constraint satisfaction problems with infinite templates. In Heribert Vollmer, editor, *Complexity of Constraints (a collection of survey articles)*, pages 196–228. Springer, LNCS 5250, 2008.

[Bry97] David Bryant. Building trees, hunting for trees, and comparing trees. PhD-thesis at the University of Canterbury, 1997.

[BS95] D. Bryant and M. Steel. Extension operations on sets of leaf-labelled trees. *Advances in Applied Mathematics*, 16:425–453, 1995.

[Cam90] Peter J. Cameron. *Oligomorphic Permutation Groups*. Cambridge Univ. Press, Cambridge, 1990.

[CH97] Nadia Creignou and Jean-Jacques Hébrard. On generating all solutions of generalized satisfiability problems. *Informatique Thèorique et Applications*, 31(6):499–511, 1997.

[CKS01] Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*. SIAM Monographs on Discrete Mathematics and Applications 7, 2001.

[Cor94] Thomas Cornell. On determining the consistency of partial descriptions of trees. In *Proceedings of the ACL*, pages 163–170, 1994.

[Dec92] Rina Dechter. From local to global consistency. *Artificial Intelligence*, 55(1):87–108, 1992.

[Dek86] M. C. H. Dekker. Reconstruction methods for derivation trees. Masters thesis, Vrije Universiteit, Amsterdam, 1986.

[Due05] Ivo Duentsch. Relation algebras and their application in temporal and spatial reasoning. *Artificial Intelligence Review*, 23:315–357, 2005.

[EF99] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, Berlin, Heidelberg, New York, 1999. 2nd edition.

[FV99] Tomás Feder and Moshe Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. *SIAM Journal on Computing*, 28:57–104, 1999.

[Gro94] Martin Grohe. The structure of fixed-point logics. PhD-thesis at the Albert-Ludwigs Universität, Freiburg i. Br., 1994.

[Hir97] R. Hirsch. Expressive power and complexity in algebraic logic. *Journal of Logic and Computation*, 7(3):309 – 351, 1997.

[HKW96] Monika Henzinger, Valerie King, and Tandy Warnow. Constructing a tree from homeomorphic subtrees, with applications to computational evolutionary biology. In *Proceedings of the 7th Symposium on Discrete Algorithms (SODA'96)*, pages 333–340, 1996.

[Hod97] Wilfrid Hodges. *A shorter model theory*. Cambridge University Press, Cambridge, 1997.

[Imm98] N. Immerman. *Descriptive Complexity*. Graduate Texts in Computer Science, Springer, 1998.

[KV95] Phokion G. Kolaitis and Moshe Y. Vardi. On the expressive power of Datalog: Tools and a case study. *Journal of Computer and System Sciences*, 51(1):110–134, 1995.

[KV98] Phokion G. Kolaitis and Moshe Y. Vardi. Conjunctive-query containment and constraint satisfaction. In *Proceedings of PODS'98*, pages 205–213, 1998.

[Mac77] A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.

[Mon74] Ugo Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Information Sciences*, 7:95–132, 1974.

[NSW00] Meei Pyng Ng, Mike Steel, and Nicholas C. Wormald. The difficulty of constructing a leaf-labelled tree including or avoiding given subtrees. *Discrete Applied Mathematics*, 98:227–235, 2000.

[Pos41] Emil L. Post. The two-valued iterative systems of mathematical logic. *Annals of Mathematics studies*, 5, 1941.

[Ste92] Michael Steel. The complexity of reconstructing trees from qualitative charaters and subtrees. *Journal of Classification*, 9:91–116, 1992.

[THdL98] Mikkel Thorup, Jacob Holm, and Kristian de Lichtenberg. Poly-logarithmic deterministic fully-dynamic graph algorithms I: connectivity and minimum spanning tree. Technical report, Department of Computer Science, University of Copenhagen, 1998.