The Complexity of Rooted Phylogeny Problems

Manuel Bodirsky
CNRS/LIX, École Polytechnique
Palaiseau, France
bodirsky@lix.polytechnique.fr

Jens K. Mueller
Friedrich-Schiller-University
Jena, Germany
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 atomic formulas are *rooted triples*, is there a rooted binary tree that satisfies the formula? If the formulas do not contain disjunctions and negations, 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.

Categories and Subject Descriptors
F.4.1 [**Theory of Computation**]: Mathematical Logic—*logic and constraint programming, model theory*

General Terms
Algorithms, Theory

Keywords
Constraint Satisfaction Problems, Phylogenetic Reconstruction, Computational Complexity, Datalog, ω-categorical structures

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 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 [4], motivated by problems in database theory. This algorithm was later rediscovered for phylogenetic analysis [34]. Henzinger, King, and Warnow [25] showed how to use decremental graph connectivity algorithms to improve the quadratic runtime of the algorithm by Aho et al.

Dekker [18] 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 [12]. Dekker's 'rules' have a very natural interpretation in terms of Datalog programs. Datalog as an algorithmic tool for rooted phylogeny problems is much more powerful than Dekker's rules. 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 [28, 3, 22, 23, 8]. 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 [29]. A recent generalization of the intimate connection between Datalog and the existential pebble game to a broad class of infinite domain constraint satisfaction problems [7] allows us to apply the game to study the expressive power of Datalog for the rooted triple consistency problem.

There are several important phylogeny problems that are related to the rooted triple consistency problem. One is the *subtree avoidance problem*, introduced by [32], or the *forbidden triple problem* [11]; both are NP-hard. It turns out that all of the problems and many other rooted phylogeny problems can be conveniently put into a common framework, which we introduce in this paper.

A *triple formula* is a formula Φ in conjunctive normal form where all literals are of the form $ab|c$ or $ab\nmid c$. 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 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 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 [16]. 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 $\mathcal{C}$ 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 $u_1, \dots, u_k$. The tree T is called *binary* if the root has two neighbors, and every other vertex has either three neighbors or one neighbor. 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. We write $uv\nmid w$ if u, v, w are distinct leaves of T and $yca(u, v)$ does not lie strictly below $yca(u, w)$ in T.

DEFINITION 1. *A* triple formula *is a (quantifier-free) formula Φ in conjunctive normal form with atomic formulas of the form $xy|z$. In other words, Φ is a conjunction of* clauses *(also called* triple clauses*) where each clause is a disjunction of* positive literals *of the form $xy|z$ and* negative literals *of the form $xy\nmid z$.*

EXAMPLE 1. *An example of a triple formula is $(x_1y_1|y_2 \vee x_2x_1\nmid y_1) \wedge (x_2y_1|y_2 \vee x_2x_1|y_2)$, with the clauses $(x_1y_1|y_2 \vee x_2x_1\nmid y_1)$ and $(x_2y_1|y_2 \vee x_2x_1|y_2)$.*

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

DEFINITION 2. *A 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 $xy|z$ ($xy\nmid z$) is* satisfied *by (T, α) if $\alpha(x), \alpha(y), \alpha(z)$ are distinct and if $yca(\alpha(x), \alpha(y))$ lies (does not lie) strictly below $yca(\alpha(x), \alpha(z))$ in T.*

A pair (T, α) with the properties in Definition 2 is also called a *solution* to Φ.

We would like to remark that a 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 (in the usual sense), i.e., a literal $xy|z$ ($xy\nmid z$) is *satisfied* by (T, α) if $\alpha(x)\alpha(y)|\alpha(z)$ is (is not) true in T. This is straightforward to verify. The definition of rooted phylogeny problems chosen in this paper will be more convenient when we use results from constraint satisfaction.

A fundamental problem in phylogenetic reconstruction is the rooted triple consistency problem [25, 12, 34, 4]. This problem can be stated conveniently in terms of triple formulas.

Rooted-Triple-Consistency
INSTANCE: A triple formula Φ without disjunction and negation.
QUESTION: Is Φ satisfiable?

The following NP-complete problem was introduced and studied in an equivalent formulation by Ng, Steel, and Wormald [32].

Subtree-Avoidance
INSTANCE: A triple formula Φ where each clause is a disjunction of negative literals.
QUESTION: Is Φ satisfiable?

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

Rooted-Phylogeny for clauses from $\mathcal{C}$
INSTANCE: A triple formula Φ where each clause can be

obtained from clauses in $\mathcal{C}$ by substitution of variables.
QUESTION: Is Φ satisfiable?

It is straightforward to see from the definition that all of these problems belong to NP. Note that both the rooted triple consistency problem and the subtree avoidance problem are examples of rooted phylogeny problems, by appropriately choosing the class $\mathcal{C}$. The rooted triple consistency problem equals the rooted phylogeny problem for the class $\mathcal{C}$ that consists of all clauses of the form $xy|z$. The subtree avoidance problem equals the rooted phylogeny problem for the class $\mathcal{C}$ that consists of clauses of the form $x_1y_1 \nmid z_1 \vee \cdots \vee x_sy_s \nmid z_s$.

The following problem has been studied in [11]; also here, NP-hardness of the following problem follows directly from our complexity classification result in Section 5.

Forbidden-Triple-Consistency
INSTANCE: A triple formula Φ where each clause consists of a single negative literal.
QUESTION: Is Φ satisfiable?

Phylogeny Problems as Infinite Domain CSPs. Phylogeny problems can be viewed as infinite domain *constraint satisfaction problems* (CSPs), which are defined as follows. Let Γ be a structure with a finite relational signature τ. A first-order formula over τ is called *primitive positive* if it is of the form $\exists x_1, \dots, x_n. \psi_1 \wedge \cdots \wedge \psi_m$ where $\psi_1, \dots, \psi_m$ are atomic formulas over τ, i.e., of the form $x = y$ or $R(x_1, \dots, x_k)$ 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. We cannot give a full introduction to constraint satisfaction and to constraint satisfaction on infinite domains, but point the reader to [13, 6]. 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 other rooted phylogeny problems 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 $\mathbb{N} \to \mathbb{Q}$, i.e., the set of all functions from the natural numbers to the rational numbers (hence, the domain of Δ is uncountable). For two elements f, g of Δ, let $\text{lcp}(f, g)$ be the set $\{1, \dots, n\}$ where n is the largest natural number i such that $f(j) = g(j)$ for all $j \in \{1, \dots, i\}$; if no such i exists, we set $\text{lcp}(f, g) := \emptyset$, and if $f = g$, we set $\text{lcp}(f, g) := \mathbb{N}$. The ternary relation $fg|h$ in Δ holds on elements f, g, h of Δ if they are pairwise distinct and either $|\text{lcp}(f, g)| > |\text{lcp}(f, h)|$, or $|\text{lcp}(f, g)| = |\text{lcp}(f, h)|$ and $f(|\text{lcp}(f, g)| + 1) < h(|\text{lcp}(f, h)| + 1)$.

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

Lemma 1. *A rooted triple formula $\Phi(x_1, \dots, x_n)$ is satisfiable if and only if $\exists x_1, \dots, x_n.\ \Phi(x_1, \dots, x_n)$ is true in Δ.*

Proof. Suppose that $\exists x_1, \dots, x_n.\Phi(x_1, \dots, x_n)$ is true in Δ, and let $f_1, \dots, f_n : \mathbb{N} \to \mathbb{Q}$ be witnesses for $x_1, \dots, x_n$ 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 $\text{lcp}(f_i, f_j)$ for all $1 \leq i, j \leq n$. Vertex g is above g' in T if g' extends g; it is clear that this describes T uniquely. Note that the leaves of T are exactly $f_1, \dots, f_n$. It is straightforward to verify that (T, α) satisfies Φ, where α maps x_i to f_i.

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 in T that have larger distance from the root than v. Let h be the length of the path $r = p_1, \dots, p_h = \alpha(x_i)$ from the root r to $\alpha(x_i)$ in T. Define $f_i : \mathbb{N} \to \mathbb{Q}$ by setting $f_i(j) = 0$ if $1 \leq j < h$ and $p_{j+1} = l(p_j)$, and $f_i(j) = 1$ otherwise. Clearly, the elements $f_1, \dots, f_n$ of Δ show that $\exists x_1, \dots, x_n.\Phi(x_1, \dots, x_n)$ is true in Δ. $\Box$

This shows that the rooted triple consistency problem is indeed a constraint satisfaction problem (if we allow that the template has an infinite domain). This observation will be useful in Section 3 to apply known techniques for proving Datalog inexpressibility of the rooted triple consistency problem.

The following lemma (Lemma 2) shows that the rooted triple consistency problem is among the simplest rooted phylogeny problems, that is, for every (non-trivial) class of triple clauses $\mathcal{C}$ the rooted phylogeny problem for $\mathcal{C}$ can simulate the rooted triple consistency problem in a simple way.

A triple clause (i.e., a disjunction of positive and negative triples over the variables $x_1, \dots, x_k$) 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*).

Lemma 2. *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 $\phi(x_1, x_2, x_3) = x_1x_2 \nmid x_3$ for three distinct variables x_1, x_2, x_3 (or, equivalently, 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 $ab|c$. It is straightforward to derive from this observation that the claim holds 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 $k = 3$. $\Box$

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 studied in this paper.

3. DATALOG

Datalog is an important algorithmic concept originating both in logic programming and in database theory [1, 21, 27]. Feder and Vardi [22] observed that Datalog programs can be used to formalize efficient constraint propagation algorithms used in Artificial Intelligence [5, 31, 17, 30]. Such algorithms have also been studied for the phylogenetic reconstruction problem. It was asked by Dekker [18] whether there exists a set of rules for inferring rooted triples from a set of rooted triples Φ such that a rooted triple can be derived by these rules if and only if it is logically implied by Φ. This was answered negatively by Bryant and Steel [12].

In this section, we will 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).

In our proof, we use a pebble-game that was introduced to describe the expressive power of Datalog [28] and which was later used to study Datalog as a tool for finite domain constraint satisfaction problems [22]. The correspondence between Datalog and pebble games extends to infinite domain constraint satisfaction, if the template is a countably infinite ω-categorical structure. A countably structure is called *ω-categorical* if its first-order theory[1] has exactly one countable model up to isomorphism.

It can be verified (see [2]) that the first-order theory of Δ introduced in the previous section has exactly one countable model Λ up to isomorphism. The ω-categorical structure Λ has been studied in the context of infinite permutation groups [19, 2]. Since Δ and Λ have the same first-order theory, we have in particular that a rooted triple formula is satisfiable in Δ if and only if it is satisfiable in Λ. The observation that the rooted triple consistency problem can be formulated as a CSP with an ω-categorical template has been made in [24]. 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 [7]. The definition of this game given below is adapted to the special case of the rooted triple consistency problem, to keep notation simple.

The existential k-pebble game (for the rooted triple consistency problem) is played by the players Spoiler and Duplicator on an instance Φ of the rooted triple consistency problem and Λ. Each player has k pebbles, $p_1, \dots, p_k$ for Spoiler and $q_1, \dots, q_k$ for Duplicator. Spoiler places his pebbles on the variables of Φ, Duplicator her pebbles on elements of Λ. Initially, no pebbles are 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 Λ. Let $i_1, \dots, i_m$ be the indices of the pebbles that are placed on Φ (and Λ) after the i-th round. Let $a_{i_1}, \dots, a_{i_m}$ ($b_{i_1}, \dots, b_{i_m}$) be the variables of Φ (the elements of Λ) pebbled with the pebbles $p_{i_1}, \dots, p_{i_m}$ ($q_{i_1}, \dots, q_{i_m}$) after the i-th round. If $\Phi[\{a_{i_1}, \dots, a_{i_m}\}]$ contains a rooted triple $a_{i_j} a_{i_k} | a_{i_l}$ but $b_{i_j} b_{i_k} | b_{i_l}$ does not hold in Λ then the game is over, and Spoiler wins. Duplicator wins if the game continues forever.

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

THEOREM 1 (FOLLOWS FROM THEOREM 5 IN [7]). *There is no Datalog program that solves the rooted triple consistency problem if and only if for every k there exists an unsatisfiable instance Φ_k of the rooted triple consistency problem such that Duplicator wins the existential k-pebble game on Φ_k and Λ.*

To construct the instance Φ_k, we choose prime numbers r, s, t such that $2k < r < s < t$. The instance has the variables $v_0, \dots, v_{t-1}$, and contains the triples $v_i v_{i+r} | v_{i+s}$ where all indices are modulo t.

Note that because r and t are pairwise prime, for any pair v, v' from $v_0, \dots, v_{t-1}$ there exists a sequence $v = u_0, \dots, u_l = v'$ such that for all $0 \leq i < l$ there is a triple $u_i u_{i+1} | u'_i$ in the instance for some u'_i. Hence, a condition due to Aho et al. [4] implies that Φ_k is unsatisfiable for all $k \geq 1$. This can also be seen by Lemma 6 in Section 4. We are left with the proof that Duplicator wins the existential k-pebble game on Φ_k and Λ.

DEFINITION 3. *The* incidence graph $G(\Phi)$ *of an instance Φ of the rooted triple consistency problem is the (simple undirected) graph whose vertex set is the union of the set of variables and the set of triples of Φ, and where a variable x is adjacent to a rooted triple if x is one of the arguments of the triple.*

LEMMA 3. *If S is a subset of at most $2k$ variables from Φ_k, then $G(\Phi_k[S])$ is acyclic.*

PROOF. Since r, s, t pairwise do not have common divisors, and $2k < r < s < t$, all cycles in $G(\Phi_k)$ have length at least $2k+1$. Since S has size at most $2k$, the graph $G(\Phi_k[S])$ cannot contain cycles. $\square$

Suppose we are at some stage of the existential k-pebble game on Φ_k and Λ. The following definition is inspired by a Datalog inexpressibility result that was established for a very different problem in temporal reasoning [8].

DEFINITION 4. *A subset S of at least 2 and most $2k$ variables of Φ_k is called* dominated *if $G_S := G(\Phi_k[S])$ is connected (and hence by Lemma 3 a tree), and if all but at most one of the leaves of G_S are pebbled.*

Note that the unpebbled leaf in G_S (if it exists) is always an element of S. The notion of dominated sets allows us to specify the winning strategy for Duplicator for the existential k-pebble game. First, we prove a lemma that shows that Duplicator always finds an answer to the unpebbled leaf in a dominated set.

LEMMA 4. *Let S be a dominated set of Φ_k, and let $a_1, \ldots, a_m$ be the pebbled leaves of $G(\Phi_k[S])$. Then any injective mapping from $a_1, \ldots, a_m$ to Λ can be extended to a mapping α from S to Λ that is a satisfying assignment for $\Phi_k[S]$.*

PROOF. The proof is by induction on the number i of pebbled leaves of $G(\Phi_k[S])$. If $i = 1$, there is nothing to show. Otherwise, there exists a vertex u in $G(\Phi_k[S])$ which is adjacent to two pebbled leaves x and y. There are two cases to consider: u is a rooted triple $xy|z$, or u is a rooted triple $xz|y$. In both cases, it follows straightforwardly from the properties of Λ that any mapping from $\{x, y\}$ to Λ can be extended to z such that the extension satisfies the rooted triple u. Hence, we can consider a smaller dominated set where z is pebbled instead of x, y, and apply the induction hypothesis. $\square$

LEMMA 5. *Duplicator wins the existential k-pebble game on Φ_k and Λ.*

PROOF. Let u be a previously 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 always maintains the following invariant. *Whenever Spoiler places a pebble on u, Duplicator can play a value v from Λ such that the mapping that assigns a_i to b_i for $1 \leq i \leq l$ and that maps u to v can be extended to all of S such that this extension is a satisfying assignment for $\Phi_k[S]$.*

Clearly, the invariant is satisfied at the beginning of the game. Suppose that during the game Spoiler pebbles a variable u. Let $S_1, \ldots, S_p$ be the dominated sets where u is the unpebbled leaf before Spoiler put his pebble on u. (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 u. 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. Note that the union S of the sets $S_1, \ldots, S_p$ was itself a dominated set already before Spoiler played on u, unless $p = 0$ (here we use the fact that there are at most k pebbles in the game). The next move of Duplicator is the value v from the invariant applied to S. If $p = 0$, Duplicator plays an arbitrary element in Λ. To show that this move satisfies the invariant, we apply Lemma 4 to $T_1, \ldots, T_q$. $\square$

COROLLARY 1. *There is no Datalog program that solves the rooted triple consistency problem.*

PROOF. As stated above, for all k the structure Φ_k is unsatisfiable. However, by Lemma 5, Duplicator wins the existential k-pebble game on Φ_k and Λ. The statement now follows from Theorem 1. $\square$

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 5. *A disjunction $\psi := x_1y_1|z_1 \vee \cdots \vee x_py_p|z_p$ is called* admissible *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 admissible 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 [4, 25].

We start with a general observation how one can solve rooted phylogeny problems. For a given rooted triple formula Φ, we proceed recursively as follows: for each recursive call, we try to find a subset S of the variables V of Φ such that there is a solution (T, α) where

- r is the root of T,
- $\alpha(S)$ are the leaves in T below the left child of r, and
- $\alpha(V \setminus S)$ are the leaves in T below the right child of r.

After we have found a candidate set for S, the procedure recursively solves the restriction of Φ to S and the restriction of Φ to $V \setminus S$. If both recursive calls succeed and find the solutions (T_1, α_1) and (T_2, α_2), respectively, then the algorithm creates a new vertex r, and returns (T, α), where T is the tree rooted at r where the left child of r is the root of T_1 and the right child is the root of T_2, and where α is the common extension of α_1 and α_2.

Of course, the crucial question is how to find such a set S. It turns out that this problem is closely related to a Boolean constraint satisfaction problem (Boolean CSP) on the same set of variables, which we will call the *split problem for* Φ (the split problem will also be important in Section 5). The variables that are mapped to 0 in a satisfying assignment to the split problem are precisely the variables that end up in S, i.e., we obtain a valid set S by computing a solution s to the Boolean CSP, and we then set $S = s^{-1}(0)$. In order for the recursion to terminate, we will always only be interested in *non-trivial* solutions to the Boolean problem, i.e., in solutions where at least one variable is set to 0 and at least one variable is set to 1.

DEFINITION 6 (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 $(x \leftrightarrow y) \wedge (z \vee \neg z)$, and negative literals of the form $xy \nmid z$ by $(x \leftrightarrow z) \vee (y \leftrightarrow z)$.*

Since we are only interested in non-trivial solutions of the split formula, the tautological conjuncts of the form $(z \vee \neg z)$ cannot be omitted. In general, finding non-trivial solutions for Boolean formulas might be a hard task (see Section 5).

However, observe that if Φ is a rooted triple formula whose clauses come from $\mathcal{T}$, then the split formula for Φ is in fact a 2-SAT formula. We will show in Section 5 that if the clauses are restricted in a different way such that the Boolean problem is not a 2-SAT instance, then the phylogenetic reconstruction problem is NP-hard.

One of the central ideas for the polynomial-time algorithm for the rooted triple consistency problem in [4] is to associate a certain undirected graph to an instance of the rooted triple consistency problem. We generalize this idea to admissible clauses as follows.

Definition 7. *Let Φ be an instance of the rooted triple consistency problem with clauses from $\mathcal{T}$. Then $L_\Phi := (V, E)$ 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 clauses from $\mathcal{T}$.

Lemma 6. *Let Φ be an instance of the rooted triple consistency problem with clauses from $\mathcal{T}$. If L_Φ is connected then Φ is unsatisfiable.*

Proof. Let V be the set of variables in Φ. Suppose that there is a rooted tree T with leaves X and a mapping $\alpha : V \to X$ such that for every clause $xy|z_1 \vee \cdots \vee xy|z_p$ in Φ the yca of $\alpha(x), \alpha(y)$ lies strictly below the yca of $\alpha(x), \alpha(z_i)$ in T for some $1 \leq i \leq p$. Let r be the yca of $\alpha(V)$, i.e., 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 (otherwise the child would have been the yca of $\alpha(V)$). Since the graph L_Φ is connected, there is an edge $\{x, y\}$ in L_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. $\Box$

Theorem 2. *The algorithm Solve in Figure 1 determines in time $O(m \log^2 n)$ whether a given instance Φ of the rooted triple consistency problem with clauses from $\mathcal{T}$ is satisfiable (where m is the number of triples in all clauses, and n is the number of variables of Φ).*

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 L_Φ (we discuss details of this step in the paragraph about the running time of the algorithm); if $S = V$, i.e., if L_Φ is connected, then Lemma 6 implies that Φ is unsatisfiable.

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 the instance is clearly unsatisfiable as well. 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 $\psi = xy|z_1 \vee \cdots \vee xy|z_p$, it cannot be that x and y are in distinct components, since they are connected by an edge in L_Φ. If all variables of ψ lie completely inside one component, say inside the i-th component, we are done by inductive assumption, because (T_i, α_i) is a solution to $\Phi[V_i]$. Hence, there must be a j, $1 \leq j \leq p$, such that z_j is not in the i-th component. But in this case the yca of $\alpha(x)$ and $\alpha(y)$ lies below the root of T_i and hence 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 1.

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 sublinear 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 [25]. We can use the same approach and obtain an $O(m \log^2 n)$ bound for the worst-case running time of our algorithm. $\Box$

5. COMPLEXITY CLASSIFICATION

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

Theorem 3. *Let $\mathcal{C}$ be a set of rooted triple clauses. If $\mathcal{C}$ is not a subset of the class $\mathcal{T}$ presented in Section 4, then the rooted phylogeny problem for clauses from $\mathcal{C}$ is NP-complete.*

Our proof of Theorem 3 consists of two parts. In the first part, we show that if $\mathcal{C}$ is not a subset of $\mathcal{T}$, then it is NP-hard to decide whether a certain Boolean split problem (defined below) has a non-trivial solution. In the second part we show that the split problem reduces to the rooted phylogeny problem for $\mathcal{C}$.

If $\mathcal{C}$ is a class of rooted triple clauses, we define $B(\mathcal{C})$ to be the set of split formulas for clauses from $\mathcal{C}$. Recall that 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 rooted 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 [16], and is due to [15]. The notion of Horn, dual Horn, affine, and bijunctive Boolean relations are standard and introduced in detail in [16]; detailed knowledge of these notions will not be required in the further proof.

```
Solve($\Phi$)
Input: A rooted triple formula $\Phi$ with variables $V$ and clauses from $\mathcal{T}$.
Output: true if $\Phi$ is satisfiable, false otherwise.

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

Figure 1: The algorithm for the rooted phylogeny problem for clauses from $\mathcal{T}$.

THEOREM 4. *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 1. *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 from $\mathcal{C} \setminus \mathcal{T}$. Clearly, the split formula ψ for ϕ is preserved by $x \mapsto \neg x$, and is also preserved by constant operations. Moreover, it is known (and follows from [33]) 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.

We assume that ϕ is a disjunction of triples of the form $xy|z$; this is without loss of generality, since we can replace literals for the form $xy \nmid z$ by $yz|z \vee xz|y$. 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, observe that ϕ must contain triples $ab|c$ and $uv|z$ where $\{a, b\} \neq \{u, v\}$: otherwise, if for all disjuncts $ab|c$ and $uv|w$ we have $\{a, b\} = \{u, v\}$, then ψ would be a 2-SAT formula, in contradiction to the assumption that $\phi \notin \mathcal{T}$.

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

We now turn to the second part of the proof of Theorem 3. The idea to reduce the split problem for $B(\mathcal{C})$ to the rooted phylogeny problem for $\mathcal{C}$ is to construct instances of the phylogeny problem in such a way that they are unsatisfiable *if and only if* their split problem does not have a non-trivial solution. To implement this idea, we construct an instance of the phylogeny problem that fragments into simple and satisfiable pieces after one step of the recursion of the algorithm from Section 4.

PROPOSITION 2. *Let $\mathcal{C}$ be a finite class of rooted triple clauses. Then the split problem for $B(\mathcal{C})$ can be reduced in polynomial time to the rooted phylogeny problem for $\mathcal{C}$.*

PROOF. Suppose we are given an instance of the split problem for $B(\mathcal{C})$ with constraints $\psi_1, \ldots, \psi_m$ and variables $V = \{x_0, \ldots, x_{n-1}\}$. We create an instance Φ of the rooted phylogeny problem for $\mathcal{C}$ as follows. Because of Lemma 2, we can assume that $\mathcal{C}$ contains the clause that just consists of $ab|c$. The variables V' of Φ are triples (x, i, j) where $x \in V$, $i \in \{0, \ldots, m-1\}$, and $j \in \{1, \ldots, n-1\}$. 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 $i \in \{0, \ldots, m-2\}$. The clauses of Φ consist of two groups.

- To define the first group of clauses, suppose that ψ_i has variables $y_1, \ldots, y_q$. Let $\phi_i(y_1, \ldots, y_q)$ be the rooted triple clause that defines the Boolean relation from $B(\mathcal{C})$ used in $\psi_i(y_1, \ldots, y_q)$. 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 $\phi_i((y_1, i, 1), \ldots, (y_q, i, 1))$ to Φ.
- The second group of clauses in Φ has for all $x_s \in V$, $i \in \{0, \ldots, m-2\}$ (if $m = 1$ the second group of clauses is empty), and $j \in \{1, \ldots, n-1\}$ the clause

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

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 constraints $\psi_1, \ldots, \psi_m$, because of the first group of clauses in Φ.

Conversely, suppose that there is a non-trivial solution for $\psi_1 \wedge \cdots \wedge \psi_m$. Note that this solution is a solution to the split problem of Φ. So, on the first level of the recursion the algorithm from Section 4 does not reject Φ. Consider the instances Φ_1 and Φ_2 of the recursive calls of the algorithm. 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. The same statement holds for Φ_2. It is then easy to see that the two recursive calls of the algorithm on Φ_1 and on Φ_2 do not fail. By the correctness of the algorithm the instance Φ is satisfiable.

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

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

Proof of Theorem 3. As mentioned, the rooted phylogeny problem for $\mathcal{C}$ is clearly in NP. Let $\mathcal{C}$ be a class of rooted triple clauses that is not a subset of $\mathcal{T}$. We prove NP-hardness as follows. By Proposition 1, $B(\mathcal{C})$ is neither Horn, dual Horn, affine, nor bijunctive. Theorem 4 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 2 shows that the rooted phylogeny problem for clauses from $\mathcal{C}'$ (and hence also for clauses from $\mathcal{C}$) is NP-hard as well. $\Box$

6. CONCLUDING REMARKS

We have shown that we can decide the consistency of rooted phylogeny data even if the data consists of restricted forms of disjunctions of rooted triples. Our algorithm extends previous algorithmic results about the rooted triple consistency problem, without sacrificing worst-case efficiency. Our class of rooted triple clauses $\mathcal{T}$ that can be handled efficiently is also motivated by another result of this paper, which states that any set of rooted 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 (Boolean) constraint satisfaction problems, witnessing how techniques in constraint satisfaction theory can be applied to classes of problems that previously have not been studied in connection with constraint satisfaction.

Another such transfer of tools from constraint satisfaction to phylogenetic reconstruction is our result that no Datalog program can solve the rooted triple consistency problem. To show this result we use a pebble game that captures the expressive power of Datalog for constraint satisfaction problems with infinite ω-categorical templates.

We would like to mention that 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 [10];
- The network consistency problem of the left-linear-point algebra [20, 26];
- Cornell's tree description logic [14, 9];

Note that these problems can simulate the rooted triple consistency problem by simple reductions [9].

7. REFERENCES

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

[2] S. Adeleke and P. M. Neumann. Relations related to betweenness: their structure and automorphisms. *AMS Memoir*, 131(623), 1998.

[3] 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.

[4] A. Aho, Y. Sagiv, T. Szymanski, and J. 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.

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

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

[7] M. Bodirsky and V. 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.

[8] M. Bodirsky and J. Kára. A fast algorithm and Datalog inexpressibility for temporal reasoning. *Preprint, CoRR abs/0805.1473. Accepted for publication in the ACM Transactions of Computing Systems*, 2008.

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

[10] M. Broxvall and P. Jonsson. Point algebras for temporal reasoning: Algorithms and complexity. *Artif. Intell.*, 149(2):179–220, 2003.

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

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

[13] 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.

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

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

[16] N. Creignou, S. Khanna, and M. Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems.* SIAM Monographs on Discrete Mathematics and Applications 7, 2001.

[17] R. Dechter. From local to global consistency. *Artif. Intell.*, 55(1):87–108, 1992.

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

[19] M. Droste. Structure of partially ordered sets with transitive automorphism groups. *AMS Memoir*, 57(334), 1985.

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

[21] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory.* Springer, 1999. 2nd edition.

[22] T. Feder and M. 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.

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

[24] P. Hell and J. Nesetril. The core of a graph. *Discrete Math.*, 109:117–126, 1992.

[25] M. Henzinger, V. King, and T. 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.

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

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

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

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

[30] A. K. Mackworth. Consistency in networks of relations. *AI*, 8:99–118, 1977.

[31] U. Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Inf. Sci.*, 7:95–132, 1974.

[32] M. P. Ng, M. Steel, and N. C. Wormald. The difficulty of constructing a leaf-labelled tree including or avoiding given subtrees. *Discrete Applied Mathematics*, 98:227–235, 2000.

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

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