Dominance Constraints: Algorithms and Complexity

Alexander Koller
Department of Computational Linguistics
Universität des Saarlandes
Saarbrücken, Germany
www.coli.uni-sb.de/~koller

Joachim Niehren
Programming Systems Lab
Universität des Saarlandes
Saarbrücken, Germany
www.ps.uni-sb.de/~niehren

Ralf Treinen
Laboratoire de Recherche en Informatique
Université de Paris-Sud
Orsay, France
www.lri.fr/~treinen

Abstract

Dominance constraints for finite tree structures are widely used in several areas of computational linguistics including syntax, semantics, and discourse. In this paper, we investigate algorithmic and complexity questions for dominance constraints and their first-order theory. We present two NP algorithms for solving dominance constraints, which have been implemented in the concurrent constraint programming language Oz. The main result of this paper is that the satisfiability problem of dominance constraints is NP-complete. Despite this intractability result, the more sophisticated of our algorithms performs well in an application to scope underspecification. We also show that the existential fragment of the first-order theory of dominance constraints is NP-complete and that the full first-order theory has non-elementary complexity.

Keywords. *Constraints, complexity, computational linguistics, underspecification.*

Introduction. Dominance constraints for finite tree structures are widely used in computational linguistics. In syntax, they provide for underspecified tree descriptions employed in deterministic parsing [6] and to combine TAG with unification grammars [12]. In the underspecified treatment of scope ambiguities in semantics, dominance constraints are omnipresent. While they are somewhat implicit in earlier approaches [9, 2], they are used explicitly in two recent formalisms [4, 7]. Finally, an application of dominance constraints in discourse semantics has recently been proposed in [5]. In this paper, we investigate algorithmic and complexity questions for dominance constraints and several fragments of their first-order theory. The language DOM of dominance constraints is built up from variables X, Y and a ranked signature of function symbols f, g (with arities $n \geq 0$) according to the following formal syntax.

$$\varphi ::= X \lhd^* Y \mid X{:}f(X_1, \ldots, X_n) \mid \varphi \wedge \varphi'$$

Dominance constraints are interpreted over finite tree structures. These are first-order structures whose domains are sets of nodes of finite trees, where we take a tree to be a ground term according to the ranked signature. A solution of a constraint in DOM consists of a finite tree structure and an assignment of variables to nodes of this structure. (Reflexive, transitive) dominance $X \lhd^* Y$ means that in a solution of the constraint, the node denoted by X has to dominate the one denoted by Y. Labeling $X{:}f(X_1, \ldots, X_n)$ requires X to be labeled by f and immediately dominate $X_1, \ldots, X_n$, from left to right; n is the arity of f. Beyond the language DOM, we can also consider more expressive fragments by allowing the other propositional connectives and quantification over nodes.

For readability, we represent a dominance constraint as a graph which has one node for every variable in the constraint. In Figure 1, an example for a constraint and its representing constraint graph are given.

$$X{:}f(Y) \wedge Y{:}a \wedge X \lhd^* Z \wedge Z \lhd^* Y$$

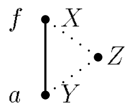

Figure 1: A constraint and its graph.

For every dominance constraint $X \lhd^* Y$, we connect the graph nodes corresponding to X and Y with a dotted line. For every labeling constraint $X : f(X_1, \ldots, X_n)$, we label X with f and connect it to its children with solid edges. The constraint in Figure 1 has two solutions, each of which is based on the tree structure $f(a)$, containing two nodes. In both solutions, X is mapped to the upper of these two, and Y, to the lower one. The difference is in the denotation of Z, which can be either node. From a different perspective, "dominance triangles" as in the example can be used to describe that Z must be one (and can be either) of two nodes. This way of representing disjunction will be very useful later.

Note the difference between the graph representation of a constraint and its solutions. The nodes in the graph represent variables in the constraint, which denote tree nodes in the solutions.

An Application to Scope Underspecification. As a more serious example, we briefly review how dominance constraints can be used for scope underspecification as in CLLS [4]. It is well known that the sentence (1) has two different readings that can be represented by the first-order formulae in (2) and (3).

(1) *Every man loves a woman.*

(2) $\forall x.\mathsf{man}(x) \rightarrow (\exists y.\mathsf{woman}(y) \wedge \mathsf{love}(x, y))$

(3) $\exists y.\mathsf{woman}(y) \wedge (\forall x.\mathsf{man}(x) \rightarrow \mathsf{love}(x, y))$

These formulae can be read as trees by interpreting every logical connective as a tree constructor of appropriate arity. The essential difference between the two readings is that in one of them, the existential quantifier is in the scope of the universal one, and in the other, vice versa. This structural difference can be expressed with a dominance constraint according to Figure 2.

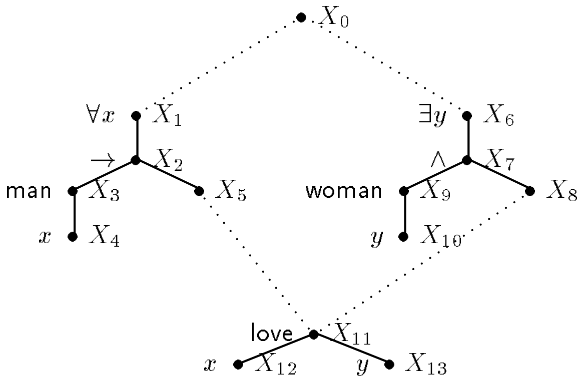

Figure 2: An underspecified representation of the meaning of Example 1.

Every tree satisfying the constraint must contain all the fragments shown in the picture. In addition, the quantifiers must dominate the $\mathsf{love}(x, y)$ subformula. Because of the tree structure of the solutions, one quantifier must dominate the other in any solution of the constraint; but there is no requirement which of them dominates the other. Hence, both readings of the sentence satisfy the constraint.

Technical Contributions. This paper presents sharp complexity results for several fragments of the first-order theory of DOM. We show that both the satisfiability problem of dominance constraints and the satisfiability problem of the existential fragment of the first-order theory of DOM are NP-complete. Part of our proof is to present a simple algorithm that decides satisfiability in NP. In addition, we present a refinement of this algorithm that we have implemented in the concurrent constraint programming language Oz [8, 11]. This implementation is based on finite set constraints in

Oz following an idea presented in [3]. Despite the intractability result, the implementation runs very efficiently on all examples from scope underspecification that we have tested it on.
Furthermore, we give a new proof of the decidability of the first-order theory of DOM both in the class of finite and in the class of arbitrary tree models by reduction to $S2S$, thus obtaining an upper bound that grows with an exponential tower of 2's of height n (the size of the formula). It is not possible to obtain a more feasible algorithm since, as we show, the problem is of non-elementary complexity in both cases.

Related Work. In [10], it was shown how to solve formulae from the propositional language over (a variant of) DOM. There, so-called quasi-trees (from which equivalent trees can be extracted in a straightforward manner) are derived from the formula by a set of logical inference rules. This solution procedure terminates, but there are no complexity results.
A complete axiomatisation of the first-order theory of dominance constraints in the class of finite tree models has been given in [1]. Decidability of the theory does not follow immediately from the completeness of the axiomatisation since the theory is not complete (e.g., $\exists X\ X : a$, stating that the tree contains the constant a, is neither valid in all models nor falsified in all models). However, as noticed by Backofen (personal communication) and Vijay-Shanker, decidability of the theory is a consequence of the completeness of the axiomatisation together with the finite model property: The theory is recursively enumerable by the completeness of the axiomatisation, and co-recursively enumerable since we can enumerate all models in search for a model that falsifies the given formula.
Another implementation of dominance constraints is presented in [3]. Their approach to the problem is rather different from ours, and as they do not give an abstract algorithm, the exact relation of their implementation and ours remains unclear.

Plan of the paper. In this abstract, we only illustrate the NP-hardness result for the satisfiablity problem of DOM and try to convey the basic intuition. The full version of the paper contains the following additional contributions. We give a formal proof of NP-hardness. We also present the solution algorithms (obtaining NP-completeness along the way) and briefly review our implementation. We present our proof of decidability of the first-order theory of DOM by an encoding into $S2S$. Finally, we show that the complexity of the first-order theory is non-elementary.

NP-Hardness. We present an example for the encoding of the satisfiability problem of propositional logic formulae in CNF into dominance constraints.
Figure 3 shows the encoding of the formula $(X_1 \vee \neg X_2 \vee X_3) \wedge (\neg X_1 \vee X_2)$ as a dominance constraint. We employ the graphical notation introduced above. Here, the signature is $\{\mathsf{f} : 2, \mathsf{true} : 0, \mathsf{false} : 0\}$. All internal nodes of the graph are labeled with f, but we have left this out of the picture for sake of readibility.
The lower left part of the graph holds a variable assignment for the propositional formula; assignments of truth values to a propositional variable can be expressed by labeling the corresponding node in a solution of the constraint with *true* or *false*. For each literal in the formula, there is one node L_{ij} whose right daughter selects one of these variables and imposes a constraint on the label of the variable node, thus restricting the possible propositional variable assignments. For each clause, there is a node C_i that must be identified with one of the L_{ij}. In doing so, the literal that is satisfied in this clause is chosen. The mechanism for enforcing the identification of C_i with one of the L_{ij} is exactly the “dominance triangle” that we have seen in our first example. If the original propositional formula is satisfied by a variable assignment, this assignment can be modeled in a straightforward way by a labeling of the variable nodes and by picking the satisfied literal in each clause. Unsatisfiability means that it would be necessary to assign two different truth values to the same propositional variable at once. In such a situation, the dominance constraint encoding will

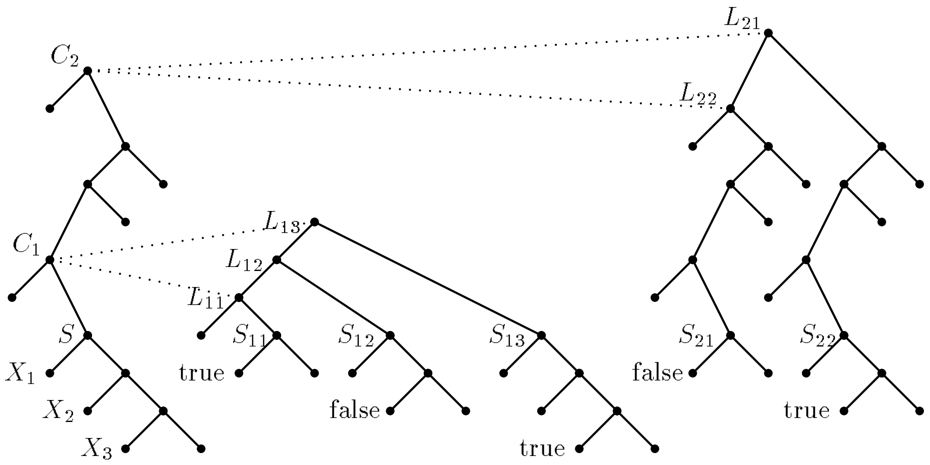

Figure 3: An encoding of $(X_1 \vee \neg X_2 \vee X_3) \wedge (\neg X_1 \vee X_2)$ as a dominance constraint.

require the labeling of the same node with both *true* and *false*, which is likewise unsatisfiable.

Theorem 1 *The satisfiability problem of dominance constraints is NP-hard.*

Acknowledgments. We would like to thank Manuel Bodirsky for implementing dominance constraints in Oz and Denys Duchier for having provided us with the idea of doing so via finite set constraints.

References

[1] R. Backofen, J. Rogers, and K. Vijay-Shanker. A first-order axiomatization of the theory of finite trees. *Journal of Logic, Language, and Information*, 4(1):5–39, 1995.

[2] J. Bos. Predicate logic unplugged. In *Proc. of the 10th Amsterdam Colloquium*, pages 133–143, 1996.

[3] D. Duchier and C. Gardent. A constraint-based treatment of descriptions. In *Proceedings of IWCS-3*, Tilburg, 1999. To appear.

[4] M. Egg, J. Niehren, P. Ruhrberg, and F. Xu. Constraints over lambda-structures in semantic underspecification. In *Proc. of COLING/ACL*, pages 253–359, Aug. 1998.

[5] C. Gardent and B. Webber. Describing discourse semantics. In *Proceedings of the TAG+ workshop*, Philadelphia, 1998.

[6] M. P. Marcus, D. Hindle, and M. M. Fleck. D-theory: Talking about talking about trees. In *Proc. ACL*, 1983.

[7] R. Muskens. Underspecified semantics. Technical Report 95, Universität des Saarlandes, Saarbrücken, 1998.

[8] Oz. The Oz Programming System, 1993. Programming Systems Lab, Universität des Saarlandes: `www.ps.uni-sb.de`.

[9] U. Reyle. Dealing with ambiguities by underspecification: construction, representation, and deduction. *J. of Semantics*, 10:123–179, 1993.

[10] J. Rogers and K. Vijay-Shanker. Extracting trees from their descriptions: An application to tree-adjoining grammars. *Computational Intelligence*, 10(4):401–421, 1994.

[11] G. Smolka. The Oz Programming Model. In *Computer Science Today*, volume 1000 of *LNCS*, pages 324–343, 1995.

[12] K. Vijay-Shanker. Using descriptions of trees in a tree adjoining grammar. *Computational Linguistics*, 18(4):481–518, 1992.