April 25, 2007

FG 2007: The 12th conference on Formal Grammar Dublin, Ireland August 4-5, 2007

**Organizing Committee: Laura Kallmeyer
Paola Monachesi Gerald Penn
Giorgio Satta**

**CENTER FOR THE STUDY
OF LANGUAGE
AND INFORMATION**

April 25, 2007

1

Relating dominance formalisms

ALEXANDER KOLLER AND OWEN RAMBOW

Abstract

We establish for the first time a formal relationship between dominance graphs, used for modeling semantics, and grammar formalisms with underspecified dominance links, used for modeling syntax. We present a translation of normal dominance graphs into Unordered Vector Grammars with Dominance Links (UVG-DL) and prove that the configurations of the dominance graph correspond to the derivation trees of the grammar. Moreover, the standard algorithms for both formalisms compute isomorphic charts.

1.1 Introduction

One of the most useful relations that is being used in describing sets of trees in a compact way is the dominance relation, i.e. the relation between a node in a directed graph or tree and all the nodes that are reachable from it. This relation has been at the foundation of at least two very different lines of research on formal grammars. The first line of research is the use in underspecification formalisms for scope ambiguities in computational semantics (Bos, 1996, Egg et al., 2001, Althaus et al., 2003, Koller et al., 2003, Bodirsky et al., 2004, Flickinger et al., 2005, Koller and Thater, 2005); these approaches use dominance to model the outscopes relation between quantifiers. The second line of research is the development of generative grammar formalisms in which the use of grammar production rules can be constrained by dominance relations and of dominance-based tree description formalisms (Vijay-Shanker, 1992, Rambow, 1994, Becker and Rambow, 1995, Rambow and Satta, 1996, Rambow et al., 2001, Rogers, 2003, Gerdes and Kahane, 2006); this work is typically motivated from the syntax of free word-order languages.

In this paper, we establish a formal relationship between these two lines of research for the first time. We present a translation of normal dominance

FG-2007.
Organizing Committee:, Laura Kallmeyer, Paola Monachesi, Gerald Penn, Giorgio Satta.

graphs into vector grammars with dominance links (UVG-DL) and prove that the configurations of the dominance graph and the derivation trees of the grammar correspond to each other. In addition, we also prove that the *charts* that are computed by the standard solver for dominance graphs and the standard parser for UVG-DL are isomorphic for those dominance graphs for which the comparison is meaningful. This means that normal dominance graphs can be seen as a fragment of UVG-DL with particularly benign computational properties.

Our results are theoretically interesting because they bridge a gap between two previously disconnected families of formalisms. Applications include the first ever algorithm for computing configurations of dominance graphs directly and an NP-completeness proof for the word problem of UVG-DL where the grammar is part of the input. Furthermore, our results point the way towards future extensions of dominance graphs, and open up new insights into the division of labor in the syntax-semantics interface.

The paper is structured as follows. We will first introduce the two formalisms – dominance graphs in Section 1.2 and UVG-DL in Section 1.3. We will then show how to translate dominance graphs into UVG-DL in Section 1.4. Finally, we will define the charts computed by the standard solvers of both formalisms and show that they are isomorphic in Section 1.5.

1.2 Dominance graphs

We start by defining dominance graphs and their configurations.

1.2.1 Definition

Definition 1 A *dominance graph* is a directed graph $(V, E \uplus D)$ with two kinds of edges, *tree edges* E and *dominance edges* D, such that the graph (V, E) defines a collection of node disjoint trees. We call the trees in (V, E) the *fragments* of the graph.

A node v is called a *root* if v does not have incoming tree edges. It is called a *leaf* if it does not have outgoing tree edges.

A *labelled dominance graph* over a ranked signature Σ is a triple $G = (V, E \uplus D, L)$ such that $(V, E \uplus D)$ is a dominance graph and $L : V \rightsquigarrow \Sigma$ is a partial *labelling function*. If $L(v)$ is defined, then the number of tree edges out of v must be equal to the arity of $L(v)$. If L does not assign a label to some node v, then v must be a leaf, and it must not be a root. In this case, v is also called a *hole*.

A dominance graph is called *normal* iff no node is both a root and a hole, and all dominance edges go from holes to roots.

Normal dominance graphs are a standard formalism for scope underspecification in computational semantics (Egg et al., 2001, Flickinger et al., 2005).

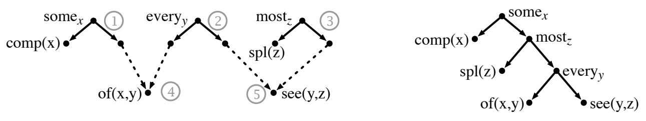


FIGURE 1 An example dominance graph, with one of its five configurations.

By way of example, consider the dominance graph in Fig. 1, which is an underspecified semantic representation for the sentence "Every researcher of a company saw most samples." It consists of five fragments, and it is normal. In this paper, we will only consider normal dominance graphs.

We use some standard concepts from graph theory below, such as *weakly connected* (there is an undirected path between any two nodes of a subgraph) and *weakly connected component* (a maximal connected subgraph). Whenever we say "subgraph" below, we mean a subgraph such that either all nodes or no nodes of each fragment belong to the subgraph.

1.2.2 Configurations

A dominance graph can be taken as a representation of a finite set of trees, called *configurations*, as follows:

Definition 2 A *solution* of a labelled dominance graph $G = (V, E \uplus D, L)$ is a pair (t, α) of a labelled tree $t = (V', E', L')$ and a node assignment function $\alpha : V \to V'$ such that for all $u, v \in V$, $L(u) = L'(\alpha(u))$ and $E' = \{(\alpha(u), \alpha(v)) \mid (u, v) \in E\}$ and if $(u, v) \in D$, then there is a path from $\alpha(u)$ to $\alpha(v)$ in t.

A solution (t, α) of G is called a *configuration* of G iff every node of t is the α-image of a non-hole in G.

Intuitively, a solution is a tree into which the dominance graph can be embedded while preserving labels and tree edges and realizing dominance edges as reachability. Configurations are arrangements of dominance graphs into trees in which all holes have been "plugged" by exactly one root.

The example graph in Fig. 1 has five configurations, one of which is shown on the right-hand side of the figure.

1.3 Unordered Vector Grammars with Dominance Links

UVG-DL (Rambow, 1994) is a member of a family of extensions of context-free grammars in which the use of the production rules in a derivation is regulated by dominance relations. UVG-DL achieves this regulation by directly extending the CFG formalism; however, there are closely related formalisms (Vijay-Shanker, 1992, Rambow et al., 2001, Rogers, 2003) which describe sets of trees as the models of a logical formula using primitives including

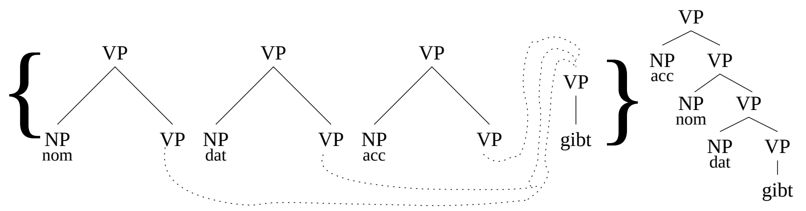


FIGURE 2 An example vector from a UVG-DL, with one of its derivation trees; internal structure of NPs omitted.

dominance. In this paper, we use the UVG-DL perspective rather than the tree description perspective, since it is easier to present formally and the parsing algorithm is more directly understandable.

Definition 3 An *Unordered Vector Grammar with Dominance Links* (UVG-DL) is a 4-tuple $(V_\mathrm{N}, V_\mathrm{T}, V, S)$, where V_N and V_T are sets of nonterminals and terminals, respectively, S is the start symbol, and V is a set of vectors (i.e., ordered multisets) of context-free productions equipped with dominance links. For a given vector $v \in V$, the dominance links form a binary relation dom_v over the set of occurrences of non-terminals in the productions of v such that if $\mathsf{dom}_v(A, B)$, then the symbol instance A occurs on the right-hand side of some production in v, and B is the left-hand symbol instance of some production in v.

If G is a UVG-DL, $L(G)$ consists of all words $w \in V_\mathrm{T}^*$ which have a standard context-free derivation ϱ of the form

$$S \xRightarrow{p_1} w_1 \xRightarrow{p_2} w_2 \dots w_{r-1} \xRightarrow{p_r} w_r = w,$$

such that ϱ meets the following two conditions:

1. $\{p_1, \dots, p_r\}$ (as a multiset) is the multiset union of some vectors;
2. the dominance relations of V, when interpreted as the standard dominance relation defined on trees, hold in the derivation tree of ϱ.

The second condition can be formulated as follows: if v in V contributes instances of productions p_1 and p_2 (and perhaps others), and the k-th daughter in the right-hand side of p_1 is related by a dominance link to the left-hand nonterminal of p_2, then in the context-free derivation tree associated with ϱ (the unique node associated with) the k-th daughter node of p_1 dominates (the unique node associated with) p_2.

We now give a simple syntactic example. In German, as in several other head-final languages, the arguments of the verb can appear in any order before the verb (we give an embedded context and omit the complementizer for expository simplicity):

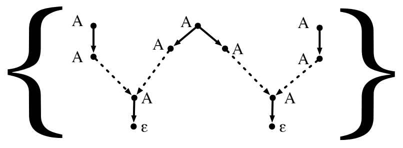


FIGURE 3 The UVG-DL grammar for the example graph.

den Hasen	der Hans	dem Zauberer	gibt
the rabbit (acc)	the Hans (nom)	the magician (dat)	gives

‘Hans gives the magician the rabbit.’

All six possible orders of the three arguments of the ditransitive verb result in grammatical sentences. These can all be derived with the unique UVG-DL vector shown in Fig. 2 on the left, and the derivation tree for our example is shown on the right.

1.4 Dominance graphs as vector grammars

We will now show how to translate dominance graphs into UVG-DL grammars, in such a way that configurations and derivation trees correspond. This connects the two formalisms; we will use the formal connection to translate algorithms and complexity results from one formalism to another. The translation is defined as follows.

Definition 4 Let G be a normal dominance graph. Then we define the *UVG-DL grammar for* G as $T(G) = (\{A\}, \emptyset, \{V\}, A)$, where V is a vector such that:

- For each fragment F in G with k holes, V contains a production rule $A \to A^k$ in V, where A^k denotes a sequence of k occurrences of the non-terminal symbol A; in particular, A^0 is ϵ. We write p_F for the production rule encoding the fragment F.
- For each dominance edge (u, v) in G, where u is the i-th hole of the fragment F_1 and v is the root of the fragment F_2, V contains a dominance link between the i-th occurrence of A on the right-hand side of p_{F_1} and the occurrence of A on the left-hand side of p_{F_2}.

It is obvious that for any dominance graph G, the language accepted by $T(G)$ can only be the empty language or the language $\{\epsilon\}$. We will see below that the language is non-empty iff the graph is configurable. Furthermore, if ϵ is in the language, the different derivation trees of ϵ according to $T(G)$ will correspond exactly to the different configurations of G.

Fig. 3 shows the result of translating the dominance graph in Fig. 1 into a grammar. This grammar contains a single vector with five production rules (two of which are $A \to A$) and four dominance links. There is an obvious visual similarity between the dominance graph and the grammar: Fragments correspond to production rules, and dominance links correspond to dominance edges. Indeed, this similarity extends to the sets of described trees, in the following way.

Proposition 1 *Let G be a normal dominance graph. Then there is a one-to-one correspondence between the configurations of G and the derivation trees of ϵ with respect to $T(G)$.*

Proof. The proof follows straightforwardly from the construction of $T(G)$, which establishes a one-to-one correspondence between fragments of G and rules of $T(G)$, with isomorphic dominance constraints. We omit the details for lack of space. □

Proposition 1 bridges two formalisms whose precise relationship was unknown up to this point. It also characterizes configurability of dominance graphs as emptiness of UVG-DL grammars. This has two immediate consequences. One is to establish a new lower bound for the parsing complexity of UVG-DL.

Corollary 2 *The parsing problem of UVG-DL where the grammar is part of the input and the emptiness problem of UVG-DL grammars are both NP-complete.*

Proof. This follows immediately from Prop. 1 and the fact that configurability of normal dominance graphs is NP-complete (Althaus et al., 2003). □

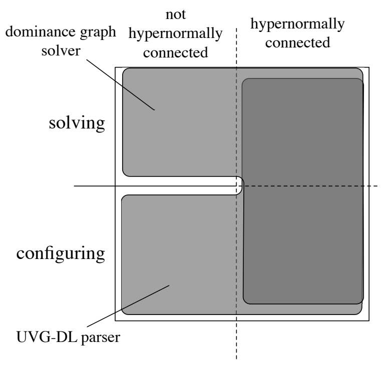
So far, the complexity of the parsing problem of UVG-DL was known to be polynomial in the size of the string, but the fastest known parsing algorithm was exponential in the grammar size. Cor. 2 shows that we can't expect to be more efficient in the grammar size.

The second consequence is that the UVG-DL parser can be used as a direct configuration algorithm for dominance graphs.

Corollary 3 *The configurations of a normal dominance graphs G can be computed by parsing ϵ with the UVG-DL grammar $T(G)$.*

1.5 Charts of hypernormally connected graphs

Corollary 3 is more important than it seems at first glance. There is a variety of efficient solvers for dominance graphs in the literature (Althaus et al., 2003, Bodirsky et al., 2004, Koller and Thater, 2005), but all these solvers determine whether a dominance graph has a *solution* and not whether it has a configuration. In general, it is possible for a dominance graph to have solutions but no configurations. Corollary 3 makes the UVG-DL parser the first known algorithm for computing the configurations of a dominance graph.



However, it is also known that every solvable graph that is *hypernormally connected* and *leaf-labelled* also has a configuration (Koller et al., 2003). A graph is called hypernormally connected iff every pair of nodes is connected by a hypernormal path; a hypernormal path is an undirected path that doesn't use any two dominance edges with the same source node. A graph is leaf-labelled iff every hole has an outgoing dominance edge. For such dominance graphs, solving and configuring coincide, and so we now have two independent algorithms for solving the same problem: the UVG-DL parser and the standard dominance graph solver.

This means we can make an even stronger claim than in the previous section: Not only do the configurations of G correspond to the derivation trees of $T(G)$ – these trees are even computed in the same way by the standard algorithms for the respective formalisms. More precisely, there is a bijective correspondence between the items in the charts computed by each of the two algorithms. We will now define the two algorithms; then we will characterize the items in each chart and prove the correspondence.

1.5.1 Charts of UVG-DL grammars

We follow Shieber et al. (1995) in taking a chart to be a set of *items* that are derived from a set of *axioms* by applying *inference rules*. A chart parsing algorithm computes a chart from a sentence by saturating the axioms with the inference rules, and claims that the sentence is in the language if it can derive a *goal item*. If the goal item can be derived, the parser can then extract all derivations of the sentence by recursively pursuing all the different ways that an item in the chart can be constructed from smaller items, starting with the goal item.

In the case of UVG-DL grammars, the parser is a basic bottom-up parser for CFG, but augmented to keep track of open dominance links (Becker and Rambow, 1995, Rambow et al., 2001), i.e. of dominance links for which we have recognized the target node, but not yet the origin node. This parser uses items (A, i, j, I) in which A is a nonterminal symbol, i and j are the start and end position in the input string, and I is a multiset of open dominance links. (There may be multiple open instances of the same link from different vector instances.)

Grammars that encode dominance graphs as defined above use only a single nonterminal symbol, can only describe input strings of length 0, and have only a single vector. This means that in the special case considered here, it is sufficient to represent each item as a set of dominance edges. The axioms are induced by the terminal productions, i.e. by the fragments with no holes; that is, we have an axiom $\mathsf{in}(r)$ for each node r that is the root of a fragment without holes. (We write $\mathsf{in}(u)$ for the set of dominance edges into, and $\mathsf{out}(u)$ for the set of dominance edges out of a node u.) Note that we know that we will use exactly one instance of our single vector in the parse, so that we also know exactly how many of each axiom to use (despite the fact that they represent epsilon-productions).

The production rules of the grammar encode the operation of combining a fragment with some subgraphs. This means that we have one inference rule U_F for each fragment F in the graph. It captures the fact that in our bottom-up algorithm, the fragment allows us to "use up" the outgoing edges and we gain the newly incoming edges. All other edges are passed on from below.

$$[U_F]\ \frac{\mathsf{out}(h_1) \uplus S_1 \qquad \dots \qquad \mathsf{out}(h_n) \uplus S_n}{\mathsf{in}(r) \cup S_1 \cup \dots \cup S_n} \quad \begin{cases} F \text{ is a fragment with root } r \\ \text{and holes } h_1, \dots, h_n \end{cases}$$

The goal of parsing is to have used all fragments and have no open dominance edges; so the goal item is simply the empty set, $\emptyset$.

1.5.2 Charts of dominance graphs

The standard chart solver for dominance graphs (Koller and Thater, 2005) is usually defined in a way that looks very different, but we will now present it (equivalently) within the same framework to make it easier to compare the two formalisms. Given a (normal, hypernormally connected, leaf-labelled) dominance graph G, the algorithm computes a chart whose items are weakly connected subgraphs of G, using a top-down algorithm. It starts with a single axiom, namely G itself. Then it saturates the chart by applying the following inference rule:

$$[D_F]\ \frac{G_0}{G_1 \qquad \dots \qquad G_n} \quad \begin{cases} G_0 \text{ is connected, } F \text{ is a free fragment in} \\ G_0\text{, and the weakly connected components} \\ \text{of } G_0 - \{F\} \text{ are } G_1, \dots, G_n \end{cases}$$

Here a fragment F in a dominance graph G is called *free* iff it has no incoming dominance edges and all of its holes are in different biconnected components of G (Bodirsky et al., 2004). It can be shown that every configuration of a graph has a free fragment at the root, and if the graph has any configurations, then there is a configuration whose root is this fragment.

Unlike a standard chart parser, the dominance graph solver does not use a goal item to determine whether the graph is configurable or not. Instead, it declares the entire graph to be unconfigurable if it ever encounters an item which is a subgraph containing more than one fragment, and to which the inference rule cannot be applied (i.e., it has no free fragment). This is possible because if any application of a rule to a subgraph leads to a configuration, then all do, i.e. the algorithm never computes unproductive items. In fact, the following lemma can be shown based on this insight (we omit a detailed proof for lack of space):

Lemma 4 *A subgraph of G can be derived from the axiom G using the inference rules top-down iff it can be derived bottom-up from the axioms $\{F_1\}, \ldots, \{F_n\}$, where the F_i are the fragments without holes.*

So if the chart computation phase finishes with success, we can enumerate all configurations by pretending that the chart is the result of a bottom-up parser starting with the axioms $\{F_1\}, \ldots, \{F_n\}$ and deriving the goal item G. In other words, the algorithm computes the chart top-down, but enumerates configurations as if it were a bottom-up algorithm.

By way of example, consider the dominance graph in Fig. 1. The start items for this graph are the subgraphs $\{4\}$ and $\{5\}$ (where “4” and “5” stand for the fragments marked with these numbers in the picture and subgraphs are identified by the fragments in them). The complete chart looks as follows:

$\{1,2,3,4,5\}$	$\{2,3,4,5\}$	$\{1,2,4,5\}$	$\{1,4\}$
$\{3,5\}$	$\{2,4,5\}$	$\{4\}$	$\{5\}$

For instance, we can justify the presence of the complete graph, $\{1,2,3,4,5\}$, in the chart because 2 is a free fragment in this graph, and if we remove it from the graph, the graph is split into the connected components $\{1,4\}$ and $\{3,5\}$, and each of these subgraphs is in the chart. These items, in turn, can be justified by recognizing that 1 is free in $\{1,4\}$ (and $\{4\}$ is a start item), and that 3 is free in $\{3,5\}$ (and $\{5\}$ is a start item).

1.5.3 Cuts

At first glance, there is no obvious connection between the sets of dominance edges computed by the UVG-DL parser and the subgraphs computed by the dominance graph solver. However, these objects are in fact very closely related, because certain sets of dominance edges *(cuts)* can be used to specify subgraphs *(cut subgraphs)* uniquely.

Let’s illustrate this by an example; consider the dominance graph in Fig. 1, and write e_{ik} for the (only) dominance edge connecting the fragments i and k. Then the set $\{e_{14}, e_{24}\}$ identifies the subgraph $\{4\}$, because this is the subgraph for which all targets of these dominance edges are inside the subgraph, and all sources are outside of it. Similarly, the set $\{e_{24}\}$ identifies the subgraph $\{1,4\}$, and so on. The entire graph is identified by the edge set $\emptyset$.

However, these subgraphs are only unique if we assume that they must be *weakly connected* and *downward closed*, i.e. if some node belongs to the subgraph, then all other nodes that it dominates must also be in the subgraph. For instance, the set of dominance edges $\{e_{14}, e_{24}\}$ could be taken to identify the subgraph $\{3, 4\}$ in the same way that it identifies $\{4\}$, because 3 has no incoming dominance edges. However, $\{3, 4\}$ is neither connected nor downward closed, and in fact $\{4\}$ is unique if we assume these additional properties. Conversely, not every set of dominance edges can be used to identify a connected and downward closed subgraph; for example, $\{e_{14}, e_{25}\}$.

As these examples show, there are certain classes of edge sets and subgraphs that correspond bijectively to each other. We call the former *cuts* and the latter *cut subgraphs*, and prove their equivalence below.

Definition 5 A set $D' = \{e_1, \ldots, e_n\}$ of dominance edges in a dominance graph G is called a *cut* iff for each $1 \leq i, k \leq n$, (a) the source and target node of e_i are disconnected in $G - E'$, and (b) there is an undirected path from the target of e_i to the target of e_k in $G - D'$.

If G is a dominance graph, then a subgraph G' of G is called a *cut subgraph* iff it is weakly connected, downwards closed, and the set of dominance edges into G' is a cut.

Lemma 5 *For each cut D' in G, there is a unique cut subgraph G whose set of incoming dominance edges is D'.*

Proof. Given a cut D', we can obtain a cut subgraph by taking the weakly connected component of $G - D'$ that contains the target of any element of D'. The subgraph is unique because if G_1 and G_2 are different subgraphs for the same cut, then we can choose a node in $G_1 - G_2$ and another node in $G_2 - G_1$ whose connecting undirected path uses a dominance edge that points into one of G_1 or G_2 but not the other. □

1.5.4 The structure of the charts

We can use this correspondence to prove that the charts computed by the two algorithms are isomorphic.

Definition 6 Two charts C and C' are *isomorphic* iff there is a bijection f between the items of C and the items of C' such that for all items $i_0, \ldots, i_n$ in C, $(i_0, \ldots, i_n)$ is an instance of an inference rule for C iff $(f(i_0), \ldots, f(i_n))$ is an instance of an inference rule for C'.

The structure of the proof is as follows. We prove that dominance graph charts consist exactly of the cut subgraphs of the dominance graph (Prop. 7). Then we prove that the UVG-DL charts consist exactly of the cuts of the dominance graph (Prop. 9). Together with the above result that the cuts and cut subgraphs correspond to each other bijectively (Lemma 5), this shows that the charts have isomorphic item sets. In other words, the chart solver for dominance graphs and the UVG-DL parser perform essentially the same computations.

Below we present the main propositions and the key lemmas. Unfortunately, we must omit the proofs for lack of space; if the paper is accepted, we will make them available as a tech report.

Lemma 6 *Let G be a normal dominance graph, and let G' be a connected, downwards closed subgraph of G'. Let F be a free fragment in G', and let $G_1, \dots, G_n$ be the connected components of $G' - F$. Then G' is a cut subgraph of G iff all the $G_1, \dots, G_n$ are.*

Proposition 7 *Let G be a configurable, hypernormally connected dominance graph. Then the items of the chart for G are exactly the cut subgraphs of G.*

Lemma 8 *Let G be a hypernormally connected, configurable dominance graph, and let $S_0, \dots, S_n$ be edge sets that are related by an inference rule for UVG-DL charts. If all the $S_1, \dots, S_n$ are cuts, then S_0 is a cut as well.*

Proposition 9 *Let G be a configurable, hypernormally connected dominance graph. Then the items of the chart for $T(G)$ are exactly the cuts of G.*

Corollary 10 *If G is a configurable, hypernormally connected dominance graph, then the chart for G and the chart for $T(G)$ are isomorphic.*

1.6 Discussion and conclusion

In this paper, we have related two strands of research on formalisms using dominance links. We have shown that normal dominance graphs, which were primarily motivated by computational semantics, can be translated into UVG-DL grammars, a grammar formalism for free word order languages. We have shown that the configurations of a dominance graph correspond to the UVG-DL derivation trees allowed by its translation. Moreover, we have shown that if the graph is hypernormally connected, then the standard solver for dominance graphs and the standard parser for UVG-DL perform essentially the same computation, in the sense that the charts they compute are isomorphic.

Next to the obvious benefit that our results relate previously unconnected formalisms, they also allow us to port algorithms and theory from each formalism to the other. We have presented a simple NP-completeness proof for the UVG-DL word problem where the grammar is part of the input, and the UVG-DL parser can now be used as the first known algorithm for computing configurations of arbitrary dominance graphs. We hope that this connection will bear more fruits of this kind in the future, e.g. by providing a straightforward way to extend dominance graphs with more powerful constraints.

More fundamentally, the work presented here opens up new perspectives for research on the nature of the syntax-semantics interface. The dominance graph tradition takes the view that scope ambiguities are purely semantic and can be computed separately from syntax, whereas the UVG-DL and tree description traditions focus on syntactic analysis. The equivalence results presented here may permit us a more flexible choice for the division of labor between syntax and semantics and lead to a broad spectrum of possible ways of arranging syntactic and semantic processing (joint, synchronous, sequential, etc.).

References

Althaus, E., D. Duchier, A. Koller, K. Mehlhorn, J. Niehren, and S. Thiel. 2003. An efficient graph algorithm for dominance constraints. *J. Algorithms* 48:194–219.

Becker, T. and O. Rambow. 1995. Parsing non-immediate dominance relations. In *Proceedings of the Fourth International Workshop on Parsing Technologies*.

Bodirsky, M., D. Duchier, J. Niehren, and S. Miele. 2004. An efficient algorithm for weakly normal dominance constraints. In *ACM-SIAM Symposium on Discrete Algorithms*. The ACM Press.

Bos, J. 1996. Predicate logic unplugged. In *Amsterdam Colloquium*, pages 133–143.

Egg, M., A. Koller, and J. Niehren. 2001. The Constraint Language for Lambda Structures. *Logic, Language, and Information* 10.

Flickinger, D., A. Koller, and S. Thater. 2005. A new well-formedness criterion for semantics debugging. In *Proc. 12th International Conference on HPSG*. Lisbon.

Gerdes, K. and S. Kahane. 2006. A polynomial parsing algorithm for the topological model: Synchronizing constituent and dependency grammars, illustrated by german word order phenomena. In *Proc. 21st COLING/44th ACL*. Sydney.

Koller, A., J. Niehren, and S. Thater. 2003. Bridging the gap between underspecification formalisms: Hole semantics as dominance constraints. In *Proc. 10th EACL*.

Koller, A. and S. Thater. 2005. The evolution of dominance constraint solvers. In *Proceedings of the ACL-05 Workshop on Software*. Ann Arbor.

Rambow, Owen. 1994. *Formal and Computational Aspects of Natural Language Syntax*. Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania, Philadelphia.

Rambow, O. and G. Satta. 1996. Synchronous models of language. In *Proc. 34th ACL*.

Rambow, Owen, K. Vijay-Shanker, and David Weir. 2001. D-Tree Substitution Grammars. *Computational Linguistics* 27(1).

Rogers, James. 2003. Syntactic structures as multi-dimensional trees. *Research on Language and Computation* 1(3–4):265–305.

Shieber, S., Y. Schabes, and F. Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming* 24(1 & 2):3–36.

Vijay-Shanker, K. 1992. Using descriptions of trees in a Tree Adjoining Grammar. *Computational Linguistics* 18(4):481–518.