Hausarbeit für das Hauptseminar *Dependenzgrammatik* SoSe 99

An Introduction to Dependency Grammar

Ralph Debusmann
Universität des Saarlandes
Computerlinguistik
rade@coli.uni-sb.de

January 2000

Contents

1 Introduction

Many linguists consider Dependency Grammar (DG) to be inferior to established phrase structure based theories like GB (Chomsky 1986), LFG (Kaplan & Bresnan 1982) and HPSG (Pollard & Sag 1994). The aim of this article is to remedy this state of affairs by seeking to make those unconvinced of DG perceive the benefits it offers.

To this end, section 2 makes the reader acquainted with the basic concepts of DG, before section 3 sets the theory against phrase structure based theories, arguing that it has considerable advantages in the analysis of languages with relatively free word order (e.g. German, Finnish, Japanese, Korean, Latin, Russian ...). Section 4 describes Duchier's (1999) DG axiomatization as a prototypical example of a DG that separates dependencies and surface order. Thereafter, section 5 proceeds with an overview of current Dependency Grammar formalisms and section 6 rounds the paper up.

2 Basic Concepts

This section serves to make those interested in Dependency Grammar aquainted with its basic concepts. It starts by illustrating the intuition behind DG.

2.1 The Intuition

Modern Dependency Grammar has been created by the French linguist Lucien Tesnière (1959), but as Covington (1990) argues, DG has already been used by traditional grammarians since the Middle Ages. The observation which drives DG is a simple one: In a sentence, all but one word depend on other words. The one word that does not depend on any other is called the *root*[1] of the sentence. A typical DG analysis of the sentence *A man sleeps* is demonstrated below in (1):

(1) *a* depends on *man*
man depends on *sleep*
sleep depends on nothing (i.e. is the root of the sentence)

or, put differently

a modifies *man*
man is the subject of *sleep*
sleep is the matrix verb of the sentence

This is Dependency Grammar.

[1]The *root* is alternatively termed *main* or *central element*.

Dependencies are motivated by *grammatical function*, i.e. both syntactically and semantically. A word depends on another either if it is a complement or a modifier of the latter. In most formulations of DG for example, functional heads or governors (e.g. verbs) subcategorize for their complements. Hence, a transitive verb like 'love' requires two complements (dependents), one noun with the grammatical function subject and one with the function object. This notion of subcategorization or valency is similar to HPSG's SUBCAT-list and even closer resembles LFG's functional completeness and coherence criteria.

Figure 1 represents the analysis of (1) graphically (dependent lexemes and categories below their heads). In Tesnière's terminology, dependency graphs (or dependency trees) of this form are called *stemmas*. The left graph in figure 1 exhibits a *real stemma*, with its nodes labeled by words, the right one a *virtual stemma*, its nodes labeled by lexical categories. Nowadays, it is however common practice to use an equivalent representation which collapses both stemmas into one tree (figure 2), and this is also adopted for the rest of this article. Unlike Tesnière's stemma, this representation also includes information about the surface order of the analyzed string (written under the dividing line from left to right).

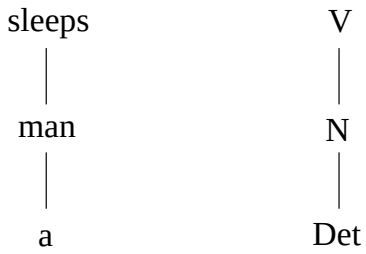


Figure 1: Stemma representations of *A man sleeps*

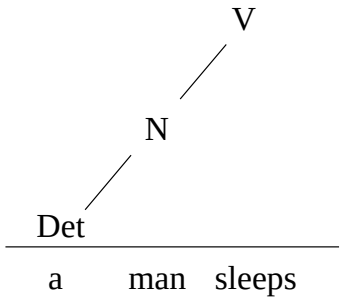


Figure 2: Dependency tree for *A man sleeps*

2.2 Robinson's Axioms

Loosely based on Tesnière's formulation, Hays (1964) and Gaifman (1965) were the first to study the mathematical properties of DG. Their results will be presented in section

3. A couple of years later, Robinson (1970) formulated four axioms to govern the well-formedness of dependency structures, depicted below in (2):

(2) 1. One and only one element is independent.
2. All others depend directly on some element.
3. No element depends directly on more than one other.
4. If A depends directly on B and some element C intervenes between them (in the linear order of the string), then C depends directly on A or B or some other intervening element.

The first three of these axioms fairly elegantly capture the essential conditions for the well-formedness of dependency structures, i.e. that they shall be trees. Axioms 1 and 2 state that in each sentence, one and only one element is independent and all others depend directly on some other element. Axiom 3 states that if an element A depends directly on another one B, it must not depend on a third one C. This requirement is often referred to as *single-headedness* or *uniqueness* and is assumed in most DG formulations, starting from Tesnière's.

The fourth axiom is often called the requirement of *projectivity* and (roughly speaking) disallows crossing edges in dependency trees. Tesnière did not impose this condition, and not without reason. I will argue in section 3 that if enforced, it deprives Dependency Grammar of its most relevant asset.

2.3 The Dependency Relation

To round up this account of basic DG concepts, I will restate the notion of dependency as a binary relation R, ranging over the elements W of a sentence. A mapping M maps W to the actual words of a sentence (for an example, see figure 3). Now for $w_1, w_2 \in W$, $\langle w_1, w_2 \rangle \in R$ asserts that w_1 is dependent on w_2. The properties of R presented are, as Robinson's (1970) axioms, nothing else but treeness constraints on dependency graphs.

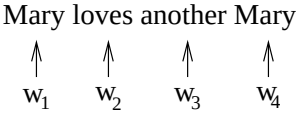


Figure 3: Example mapping M ($w_1 ... w_4 \in W$)

(3) 1. $R \subset W \times W$
2. $\forall w_1 w_2 ... w_{k-1} w_k \in W : \langle w_1, w_2 \rangle \in R \; ... \; \langle w_{k-1}, w_k \rangle \in R : w_1 \neq w_k$ (acyclicity)
3. $\exists! w_1 \in W : \forall w_2 \in W : \langle w_1, w_2 \rangle \notin R$ (rootedness)
4. $\forall w_1 w_2 w_3 \in W : \langle w_1, w_2 \rangle \in R \wedge \langle w_1, w_3 \rangle \in R \rightarrow w_2 = w_3$ (single-headedness)

From acyclicity follows that R is also asymmetrical (i.e. $\forall w_1 w_2 \in W : \langle w_1, w_2 \rangle \in R \rightarrow \langle w_2, w_1 \rangle \notin R$). The asymmetry of R is accounted for in dependency trees by an implicitly assumed ordering from top to bottom, i.e. for every two elements connected by a dependency edge, the lower one depends on the upper one. Asymmetry also guarantees that R is irreflexive ($\forall w_1 \in W : \langle w_1, w_1 \rangle \notin R$). Moreover, observe that condition 4 is a counterpart to Robinson's single-headedness axiom 3. The projectivity requirement is not reflected by any of the conditions in (3).

3 Dependency Grammar and PSG

This section sets Dependency Grammar against phrase structure based approaches. It begins with a description of a DG formulation developed by Hays (1964) and Gaifman (1965).

3.1 The Hays and Gaifman DG

A couple of years after Tesnière had defined DG, Hays (1964) and Gaifman (1965) were the first to study its mathematical properties. Their aim was to find a mathematical axiomatization of DG to facilitate development of parsing and generation algorithms, and they came up with a formulation of DG that is still highly influential among DG theorists (e.g. Lai & Huang 1998, Lombardo & Lesmo 1998).

The intuition behind Hays and Gaifman's axiomatization is this: If a dependency relation R (as defined in section 2.3) holds for $\langle w_1, x \rangle ... \langle w_k, x \rangle$, all w_i ($i \in \{1...k\}$) are dependent on x (or alternatively, x governs all w_i or is the head of all w_i). Hays and Gaifman use the following rule (plus two special case rules) to capture this notion:

(4)
1. $x(w_1, ..., *, ..., w_k) : w_1 ... w_k$ are dependent on x
2. $x(*) : x$ is a leaf node
3. $*(x) : x$ is a sentence root node

The star $*$ indicates the position of governor x in the linear order of words[2] $w_1 ... w_k$.

Such a DG for the sentence *John loves a woman* consists of five rules. (5) and figure 4 depict these and a dependency tree analyzing the sentence respectively.

(5)
$*(V)$
$V(N, *, N)$
$N(Det, *)$
$N(*)$
$Det(*)$

The attentive reader will note the addition of dotted lines connecting the lexical category nodes with the words of the sentence. They shall depict a *projection* of the lexical categories onto the words of the sentence.

[2] Actual Hays and Gaifman DG rules are expressed over categories instead of words.

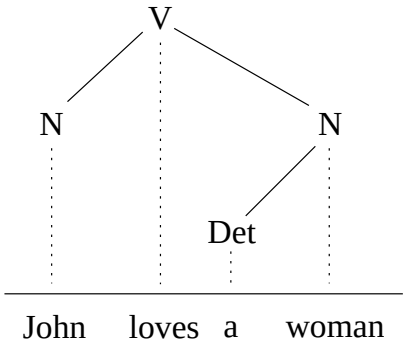


Figure 4: Hays and Gaifman DG analysis of *John loves a woman*

To conclude the explanation of Hays and Gaifman DGs, (6) shows how such DGs look like in formal terms:

(6) $DG = \langle R, L, C, F \rangle$

R a set of dependency rules over the auxiliary symbols C
L a set of terminal symbols (lexemes)
C a set of auxiliary symbols (lexical categories)
F an assignment function ($F : L \rightarrow C$)

A Hays and Gaifman DG (HGDG) complies with all of Robinson's axioms (section 2.2) for well-formed dependency structures, and to axiom 4 (projectivity) in particular. Graphically, this condition requires that no projection edge is to be crossed by any dependency edge in HGDG analysis trees.

3.2 DG Set Against CFG

By specifying the mathematical properties of DG, Hays (1964) and Gaifman (1965) did not only aim at easing development of algorithms for DG parsing and generation, but also at being able to formally set DG against Context-Free Grammars (CFGs). Hays (1964) for instance establishes that DGs as defined in section 3.1 are weakly equivalent to CFGs in the sense that:

(7)
- they have the same terminal alphabet;
- for every string over that alphabet, every structure attributed by either grammar corresponds to a structure attributed by the other.

So both CFG and HGDG are able to produce corresponding analyses for any string, but the analysis structures assigned to these strings are not necessarily the same. Thus as Hays (1964) points out, CFG and HGDG are not strongly equivalent. It is not possible to map any CFG to a corresponding HGDG that attributes equivalent tree structures to any string, simply because HGDG cannot produce nonterminal nodes other than

preterminals. The reverse however is possible. (8) exhibits a proof procedure to map any HGDG to a corresponding CFG attributing equivalent structures to any string analyzed. This procedure has been applied to (9), to generate its corresponding CFG (10).

(8) $DG = \langle R, L, C, F \rangle$: A Hays and Gaifman DG consists of a set of dependency rules R, a set of terminal symbols L, a set of nonterminal symbols C and an assigment function F ($F : L \rightarrow C$).

$CFG = \langle P, T, N, S \rangle$: A CFG consists of sets of production rules P, terminal symbols T, nonterminal symbols N and start symbols S.

I will now present a proof procedure to map any DG of the kind Hays and Gaifman proposed onto a CFG attributing equivalent analysis trees to any string. To this end, every DG rule must be converted into one or more CFG production rules. Recall that these are the three rule types of a Hays and Gaifman DG:

1. $x(w_1, ..., *, ..., w_k) : w_1 ... w_k$ are dependent on x
2. $x(*) : x$ is a leaf node
3. $*(x) : x$ is a sentence root node

Rules of the first type are converted to CFG rules using the following procedure: Collect all terminal symbols that are of category x in a set X, i.e. $X = F^{-1}(x)$. Now postulate a CFG production rule for each $y \in X$:

$x \rightarrow w_1 \ ... \ y \ ... \ w_k$

Rules of the second type are a special case of the first type. For each $y \in X$, postulate:

$x \rightarrow y$

Rules of the third type are not converted to production rules but make up the set of start symbols S for the corresponding CFG. Hence S is the union of all auxiliary symbols contained in rules of the type $*(x)$.

Finally, the set of terminal symbols T of the corresponding CFG is equal to the set of terminal symbols L of the Hays and Gaifman DG. The set of nonterminal symbols N is equal to the set of auxiliary symbols C.

(9) $DG = \langle R, L, C, F \rangle$

$R = \{*(V), V(N, *, N), N(Det, *), N(*), Det(*)\}$
$L = \{loves, woman, John, a\}$

$C = \{V, N, Det\}$
$F(loves) = V$
$F(woman) = N$
$F(John) = N$
$F(a) = Det$

(10) $CFG = \langle P, T, N, S \rangle$

$P = \{V \rightarrow N\ loves\ N, N \rightarrow Det\ woman, N \rightarrow Det\ John, N \rightarrow woman,$
$N \rightarrow John, Det \rightarrow a\}$
$T = L$
$N = C$
$S = \{V\}$

As can be seen from (8), a CFG converted from a HGDG must have one and only one terminal symbol on the right hand side of any production rule. This resembles the *Greibach Normal Form*, where each production rule is of the form $A \rightarrow a\ x_1\ ...\ x_n$, where a must be a terminal symbol and all x_1 nonterminal symbols. *Greibach Normal Forms* are, as HGDG, weakly equivalent to CFG.

Figure 5 now depicts an analysis of *John loves a woman* using the CFG given in (10). As can be seen, the analysis tree of the same sentence using a converted CFG is equivalent to the dependency tree from figure 4, except that the dotted projection edges are replaced by solid edges from preterminal to terminal nodes.

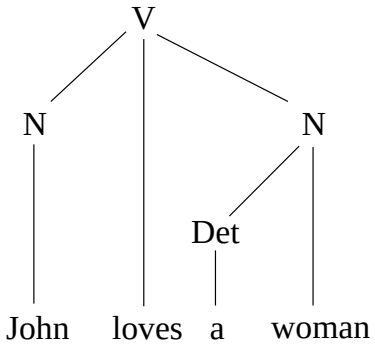


Figure 5: CFG analysis tree using (10)

Using this result, it is easy to see that the projectivity condition must hold for all analyses of a Hays and Gaifman DG:

(11) Violating the projectivity condition would require crossing edges in dependency trees resulting from such a DG. But as every DG can be mapped 1:1 to a CFG that generates exactly the same structures, and because in CFG analysis trees, crossing edges can not occur, crossing edges can also not occur in dependency trees resulting from the analysis of a HGDG. Hence projectivity is a consequence of the definition of the HGDG.

3.3 The Issue Of Projectivity

In the previous section, an examination of Hays and Gaifman's axiomatization of DG has lead to the conclusion that such a DG is merely a notational variant of CFG. So why not simply stop here and quit doing any research on dependency grammars?

Because DGs of the form Hays and Gaifman proposed do not represent Dependency Grammar in general. If they did, i.e. if dependency trees in general had to be projective, perfectly sensible DG analyses had to be abandoned, especially when analyzing languages with a high degree of word order variation like Latin, Russian or German. Consider for instance the analysis of the Latin sentence *ultima Cumaei venit iam carminis aetas*, taken from Covington (1990). Figure 6 displays a perfectly intuitive analysis of the sentence in terms of dependency, but if one complied to the condition of projectivity, it had to be abandoned and other, probably less intuitive solutions seeked.

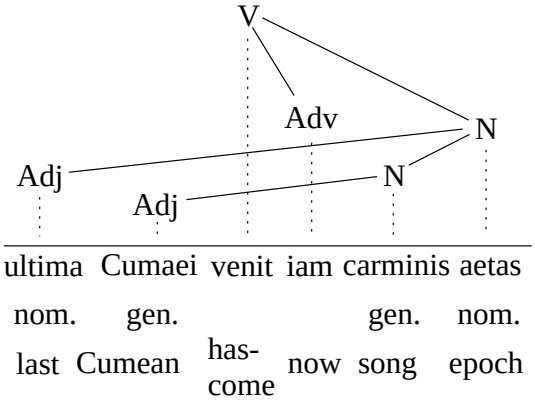


Figure 6: Non-projective DG analysis of *The last epoch of the Cumean Song has now arrived.* in Latin (Vergil, Eclogues IV.4)

If the latter sounds familiar to the reader, it is because this is precisely what PSG theorists do when confronted with discontinuous constituents or other word order variation phenomena. In generative syntax for instance, one solution are *scrambling rules* (e.g. Ross 1967). But at least insofar as computational linguistics is concerned, these rules are like any other rules involving transformations — they become unfeasible if one proceeds to implement efficient parsing algorithms.

Another solution for treating discontinuous constituents in a phrase structure based framework has been pioneered (and then rejected) by Uszkoreit (1986, 1987), in the paradigm of ID/LP formalisms like GPSG (Gazdar, Klein, Pullum & Sag 1985). He proposed to replace the transformationalists' *scrambling rules* by *flattening rules* that discard constituent structure and make most words hang directly from the S node. The problem with this approach is that a flat structure is no structure or tree at all — it only claims that the words below the S-node form a sentence. Imagine how hard a task it would constitute to construct a semantics out of such a syntactic description.

A recent proposal on treating discontinuous constituents in HPSG is due to Müller (e.g. Müller 1999) and makes use of *word order domains* to describe extraposition and

permutability of constituents in German. Constraining analysis trees to binary branching ones, Müller (1999) utilizes the *shuffle*-Relation (see Reape 1994) to allow for discontinuous constituents. Müller's (1999) proposal does succeed in providing a treatment of German scrambling phenomena within the HPSG framework, but since in HPSG, word order is not totally separable from other syntactic (functional) considerations, his analyses lack the elegance of non-projective dependency analyses like the one exhibited in Figure 6.

So none of the three popular solutions for the treatment of discontinuous phrase structure based theories described above is perfect, and that stems from the inability of PSG to totally separate word order from configurational or dependency issues. The same difficulties consequently plague projective Dependency Grammars based on the Hays and Gaifman axiomatization, since these are just notational variants of CFG. But there is a feasible escape route. As will be shown in the next section, all these problems can be solved by employing a non-projective[3] formulation of DG, separating dependency relations from surface order.

4 Separating Dependency Relations And Surface Order

In this section, I will present a formulation of a Dependency Grammar that succeeds in cleanly separating dependency relation and surface order, and by this means also lifts the projectivity condition. The approach described is due to Duchier (1999).

4.1 A non-projective Dependency Grammar

Duchier's (1999) DG can be regarded as a 7-tuple consisting of finite sets

(12) $DG = \langle Words, Cats, Agrs, Comps, Mods, Lexicon, Rules \rangle$

where *Words* is a set of strings of fully inflected word forms, *Cats* a set of lexical categories such as `V` for verb and `N` for noun, and *Agrs* a set of agreement tuples such as ⟨`masc sing 3 nom`⟩. The union of *Comps* and *Mods* forms the set of all role types *Roles* (e.g. `subject`, `adv`). Furthermore, *Lexicon* is a set of lexical entries like the one shown in (13):

(13)
$$\begin{bmatrix} \text{string} & \text{loves} \\ \text{cat} & \text{V} \\ \text{agr} & \langle \texttt{sing 3 nom} \rangle \\ \text{comps} & \{\texttt{subject}, \texttt{object}\} \end{bmatrix}$$

Finally, *Rules* is a family of binary predicates Γ_ρ for each $\rho \in Roles$ called *role constraints*.

[3] Interestingly, the founder of modern DG, Tesnière, has never imposed anything like the projectivity constraint.

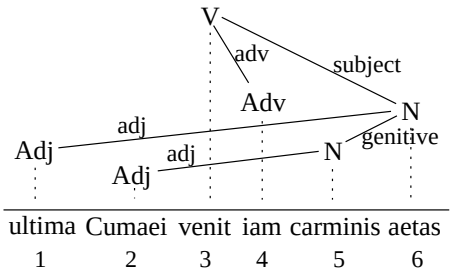
4.2 Non-projective Dependency Trees

Duchier (1999) assumes an infinite set *Nodes* of nodes and defines a labeled directed edge to be an element of $Nodes \times Nodes \times Roles$. Hence, given a set $\mathcal{V} \subseteq Nodes$ representing the words of a sentence and a set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} \times Roles$ of labeled edges between these nodes, $\langle \mathcal{V}, \mathcal{E} \rangle$ is a directed graph with labeled edges. For these graphs, Duchier (1999) imposes *treeness contraints* like those in section 2.2 (Robinson's first three axioms) or section 2.3 (dependency relation R).

Every node of a dependency tree now contributes a word to the sentence, whose position is represented by a mapping *index* from nodes to integers. Furthermore, every node must be assigned syntactic features, which will be realized by a mapping *entry* from nodes to lexical entries. A dependency tree is then defined as a 4-tuple:

(14) $T = \langle \mathcal{V}, \mathcal{E}, index, entry \rangle$

An analysis of the Latin sentence from section 3.3 in Duchier's (1999) DG is shown in figure 7.



1
$$\begin{bmatrix} \text{string} & \text{ultima} \\ \text{cat} & \text{Adj} \\ \text{agr} & \langle \texttt{fem sing 3 nom} \rangle \\ \text{comps} & \{\} \\ \text{mods} & \{\} \end{bmatrix}$$

2
$$\begin{bmatrix} \text{string} & \text{Cumaei} \\ \text{cat} & \text{Adj} \\ \text{agr} & \langle \texttt{fem sing 3 gen} \rangle \\ \text{comps} & \{\} \\ \text{mods} & \{\} \end{bmatrix}$$

3
$$\begin{bmatrix} \text{string} & \text{venit} \\ \text{cat} & \text{V} \\ \text{agr} & \langle \texttt{fem sing 3 nom} \rangle \\ \text{comps} & \{\texttt{subject}\} \\ \text{mods} & \{\texttt{adv}\} \end{bmatrix}$$

4
$$\begin{bmatrix} \text{string} & \text{iam} \\ \text{cat} & \text{Adv} \\ \text{agr} & \langle\rangle \\ \text{comps} & \{\} \\ \text{mods} & \{\} \end{bmatrix}$$

5
$$\begin{bmatrix} \text{string} & \text{carminis} \\ \text{cat} & \text{N} \\ \text{agr} & \langle \texttt{fem sing 3 gen} \rangle \\ \text{comps} & \{\} \\ \text{mods} & \{\texttt{adj}\} \end{bmatrix}$$

6
$$\begin{bmatrix} \text{string} & \text{aetas} \\ \text{cat} & \text{N} \\ \text{agr} & \langle \texttt{fem sing 3 nom} \rangle \\ \text{comps} & \{\} \\ \text{mods} & \{\texttt{genitive},\texttt{adj}\} \end{bmatrix}$$

Figure 7: Analysis of the Latin sentence, edges annotated with functional labels

The outstanding feature of Duchier (1999) is that he defines the well-formedness of dependency trees without reference to word order. In addition to basic *treeness constraints*, the only constraints he imposes upon dependency structures are *valency constraints* and *role constraints*. Roughly, the former express that each complement role (comps, e.g. `subject`, `object`) of a node must be licensed by exactly one complement dependent, and that modifiers (mods, e.g. `adv`, `adj`) can optionally occur as dependents. The latter specify additional restrictions Γ_ρ for each $\rho \in Roles$, e.g. that adjectives may only modify nouns.

4.3 Separate Specification of Word Order

Nothing has been said as yet about word order in Duchier's (1999) approach. Clearly, even languages like Latin do have some restrictions on word order variation that need to be captured to avoid overgeneration. For example, prepositions must appear before the noun they modify in the linear order of words.

An easy solution for specifying word order constraints is by using *role constraints*: $\Gamma_{preposition}(w_1, w_2) : i < j$ would constrain prepositions to stand before their noun dependents.

Sometimes, it is more adequate to specify conditions about the linear order of more than two nodes at a time. In this case, *role constraints* do not suffice, since they can only specify binary conditions. For the English language one would for instance wish to express that *noun phrases* typically look like this: $Det < Adj < N$, i.e. a noun may be preceded by adjectives and a determiner, but the determiner always precedes the adjectives (if realized). In Duchier's (1999) axiomatization, this requirement can be expressed as shown below:

(15) $\text{Seq}(\texttt{det}(w), \texttt{adj}(w), \texttt{n}(w))$

(15) constrains the determiner to always precede the adjectives and the adjectives to precede the noun they modify[4].

As can be seen, Duchier (1999) has created a non-projective DG which very cleanly separates dependency relations and word order. Such DGs are able to declaratively and precisely describe grammars of natural languages with any degree of word order variation. Duchier has already developed highly efficient parsers for English and German, applying state-of-the-art constraint technology embedded in the Oz Programming Language (*Mozart* 1998). And because of the semantic nature of dependency analyses, it is fairly easy to extend Duchier (1999) with a semantics component. In the CHORUS-project at the University of Saarland, an underspecified semantics construction module has been seamlessly integrated into Duchier's parsers.

[4]Furthermore, the determiner is optional, and an unrestricted number of adjectives may be placed between it and the noun.

5 Dependency Grammar Formalisms

This section provides an overview of current DG flavors, with an emphasis on how these formulations of DG cope with word order variation.

Functional Generative Description (Sgall, Hajicova & Panevova 1986)

Sgall et al. assume a language-independent *underlying order*, represented as a projective dependency tree, mapped via *ordering rules* to the concrete surface realization. The theory is multistratal, distinguishing five levels of representation.

Dependency Unification Grammar (Hellwig 1986)

DUG defines a tree-like data structure for the representation of syntactic analyses. The theory is non-projective and handles surface order using *positional features*. By these, also partial orderings and discontinuities can be handled.

Meaning Text Theory (Mel'čuk 1988)

Mel'čuk's formalism assumes seven strata of representation, and uses rules for mapping unordered dependency trees of surface-syntactic representations onto the annotated lexeme sequences of deep-morphological representations. Discontinuities are accounted for by *global ordering rules*.

Word Grammar (Hudson 1990)

WG is based on general graphs instead of trees. The ordering of two linked words is specified together with their dependency relation, and extraction of, e.g. objects is analyzed by establishing an additional dependency called *visitor* between the verb and the extractee. Hence WG does not cleanly separate dependencies from word order.

Functional Dependency Grammar (Järvinen & Tapanainen 1997)

FDG distinguishes between dependency rules and rules for surface linearization. It follows Tesnière's model in not only in being non-projective but also by adopting Tesnière's notion of *nuclei*. *Nuclei* are the primitive elements of FDG structures, possibly consisting of multiple lexemes.

Bröker (1998)

Surface order and dependency structures constitute two separate pieces of information. Bröker links structurally dissimilar word order domain structures to dependency trees to achieve a lexicalized, declarative and formally precise natural language description.

This list is not entirely complete, but should provide a first brief overview of current DG formalisms and their methods for dealing with word order in particular.

6 Conclusion

The aim of this article was to get through to those that think of Dependency Grammar as being inferior to phrase structure based approaches, often because of a lack of familiarity with the theory. Therefore, the basic DG concepts have been presented in section 2, starting from the original intuition and closing with the specification of a formal dependency relation R. The next section (section 3) set DG against phrase structure based theories, beginning with a presentation of the Hays and Gaifman DG and a comparison with Context-Free Grammar. Section 3 went on with discussing the requirement of projectivity, and ended up in proposing to drop this constraint to be able to adequately analyze word order variation. Hereafter, section 4 described Duchier's (1999) axiomatization as a prototypical example of a non-projective DG, before section 5 continued with an overview of current DG flavors, with an emphasis on how they treat word order variation.

All in all I think that DGs have undeniable advantages for describing languages with a higher degree of word order variation than English. But these advantages can only crop up if one lifts the constraint of projectivity and treats surface order separately from dependency. An argument by Rambow & Joshi (1994), stating that no *well-behaved* parsers for such DGs exist and that for this reason, non-projective DGs are hampered, can be turned down by mentioning recent advances (e.g. Bröker 1998, Duchier 1999, Järvinen & Tapanainen 1997) alone.

Last but not least, I wish to thank Denys Duchier, Malte Gabsdil, Christian Pietsch and Stefan Thater for useful criticism and suggestions in the course of writing this article.

References

Bröker, N. (1998), Separating surface order and syntactic relations in a dependency grammar, *in* 'COLING-ACL 98 - Proc. of the 17th Intl. Conf. on Computational Linguistics and 36th Annual Meeting of the ACL.', Montreal/CAN.

Chomsky, N. (1986), *Knowledge of Language, Its Nature, Origin, and Use*, Praeger, New York/NY.

Covington, M. A. (1990), A dependency parser for variable-word-order languages, Research Report AI-1990-01, Artificial Intelligence Programs, University of Georgia, Athens/GA.

Duchier, D. (1999), Axiomatizing dependency parsing using set constraints, *in* 'Sixth Meeting on Mathematics of Language', Orlando/FL.

Gaifman, H. (1965), 'Dependency systems and phrase-structure systems', *Information and Control* **8**(3), 304–337.

Gazdar, G., Klein, E., Pullum, G. & Sag, I. (1985), *Generalized Phrase Structure Grammar*, B. Blackwell, Oxford/UK.

Hays, D. G. (1964), 'Dependency theory: A formalism and some observations', *Language* **40**, 511–525.

Hellwig, P. (1986), Dependency unification grammar, *in* 'Proc. of the 11th Int. Conf. on Computational Linguistics', pp. 195–198.

Hudson, R. A. (1990), *English Word Grammar*, B. Blackwell, Oxford/UK.

Järvinen, T. & Tapanainen, P. (1997), A dependency parser for english, Technical report, Department of General Linguistics, University of Helsinki, Helsinki/FIN.

Kaplan, R. M. & Bresnan, J. (1982), Lexical-functional grammar: A formal system for grammatical representation, *in* J. Bresnan & R. Kaplan, eds, 'The Mental Representation of Grammatical Relations', MIT Press, Cambridge/MA, pp. 173–281.

Lai, T. B. & Huang, C. (1998), Complements and adjuncts in dependency grammar parsing emulated by a constrained context-free grammar, *in* 'Processing of Dependency-based Grammars: Proceedings of the Workshop, COLING-ACL 98', Montral/CAN, pp. 102–108.

Lombardo, V. & Lesmo, L. (1998), Unit coordination and gapping in dependency theory, *in* 'Processing of Dependency-based Grammars: Proceedings of the Workshop, COLING-ACL 98', Montral/CAN, pp. 11–20.

Mel'čuk, I. (1988), *Dependency Syntax: Theory and Practice*, State Univ. Press of New York, Albany/NY.

Mozart (1998). http://www.mozart-oz.org/.

Müller, S. (1999), *Deutsche Syntax deklarativ. Head-Driven Phrase Structure Grammar für das Deutsche*, number 394 *in* 'Linguistische Arbeiten', Max Niemeyer Verlag, Tübingen.

Pollard, C. & Sag, I. (1994), *Head-Driven Phrase Structure Grammar*, Univ. of Chicago Press, Chicago/IL.

Rambow, O. & Joshi, A. K. (1994), A formal look at dependency grammars and phrase-structure grammars, with special consideration of word-order phenomena, *in* L. Wanner, ed., 'Current Issues in Meaning-Text Theory', Pinter, London/UK.

Reape, M. (1994), 'Domain union and word order variation in german'.

Robinson, J. J. (1970), 'Dependency structures and transformation rules', *Language* **46**, 259–285.

Ross, J. R. (1967), Constraints on Variables in Syntax, PhD thesis, MIT.

Sgall, P., Hajicova, E. & Panevova, J. (1986), *The Meaning of the Sentence in its Semantic and Pragmatic Aspects*, D. Reidel, Dordrecht/NL.

Tesnière, L. (1959), *Eléments de Syntaxe Structurale*, Klincksiek, Paris/FRA.

Uszkoreit, H. (1986), Categorial unification grammar, *in* 'COLING 86,', pp. 187–194.

Uszkoreit, H. (1987), *Word Order and Constituent Structure in German*, CSLI, Stanford/CA.