Dependency Grammar and Dependency Parsing

Joakim Nivre

1 Introduction

Despite a long and venerable tradition in descriptive linguistics, dependency grammar has until recently played a fairly marginal role both in theoretical linguistics and in natural language processing. The increasing interest in dependency-based representations in natural language parsing in recent years appears to be motivated both by the potential usefulness of bilexical relations in disambiguation and by the gains in efficiency that result from the more constrained parsing problem for these representations.

In this paper, we will review the state of the art in dependency-based parsing, starting with the theoretical foundations of dependency grammar and moving on to consider both grammar-driven and data-driven methods for dependency parsing. We will limit our attention to systems for dependency parsing in a narrow sense, i.e. systems where the analysis assigned to an input sentence takes the form of a dependency structure. This means that we will not discuss systems that exploit dependency relations for the construction of another type of representation, such as the head-driven parsing models of Collins (1997, 1999). Moreover, we will restrict ourselves to systems for full parsing, which means that we will not deal with systems that produce a partial or underspecified representation of dependency structure, such as Constraint Grammar parsers (Karlsson, 1990; Karlsson et al., 1995).

2 Dependency Grammar

Although its roots may be traced back to Pāṇini's grammar of Sanskrit several centuries before the Common Era (Kruijff, 2002) and to medieval theories of grammar (Covington, 1984), dependency grammar has largely developed as a form for syntactic representation used by traditional grammarians, especially in Europe, and particularly in Classical and Slavic domains (Mel'čuk, 1988). This grammatical tradition can be said to culminate with the seminal work of Tesnière (1959), which

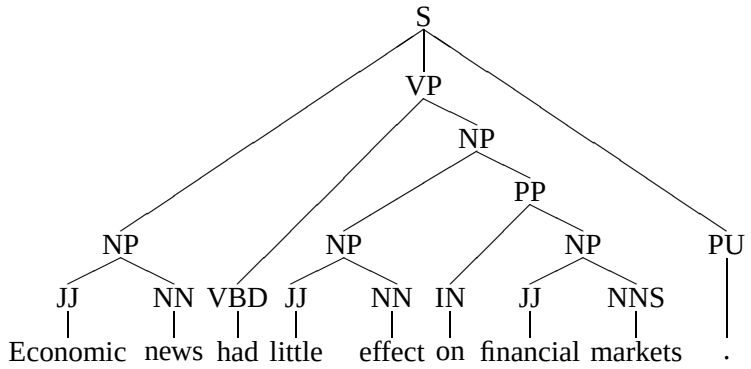

Figure 1: Constituent structure for English sentence from the Penn Treebank

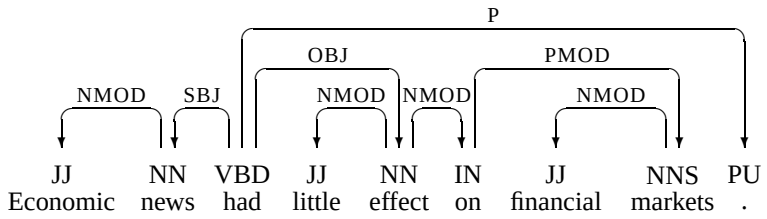

Figure 2: Dependency structure for English sentence from the Penn Treebank

is usually taken as the starting point of the modern theoretical tradition of dependency grammar.

This tradition comprises a large and fairly diverse family of grammatical theories and formalisms that share certain basic assumptions about syntactic structure, in particular the assumption that syntactic structure consists of *lexical* elements linked by binary asymmetrical relations called *dependencies*. Thus, the common formal property of dependency structures, as compared to representations based on constituency is the lack of phrasal nodes. This can be seen by comparing the constituency representation of an English sentence in Figure 1, taken from the Wall Street Journal section of the Penn Treebank (Marcus et al., 1993, 1994), to the corresponding dependency representation in Figure 2.

Among the more well-known theories of dependency grammar, besides the theory of structural syntax developed by Tesnière (1959), we find Word Gram-

mar (WG) (Hudson, 1984, 1990), Functional Generative Description (FGD) (Sgall et al., 1986), Dependency Unification Grammar (DUG) (Hellwig, 1986, 2003), Meaning-Text Theory (MTT) (Mel'čuk, 1988), and Lexicase (Starosta, 1988). In addition, constraint-based theories of dependency grammar have a strong tradition, represented by Constraint Dependency Grammar (CDG) (Maruyama, 1990; Harper and Helzerman, 1995; Menzel and Schröder, 1998) and its descendant Weighted Constraint Dependency Grammar (WCDG) (Schröder, 2002), Functional Dependency Grammar (FDG) (Tapanainen and Järvinen, 1997; Järvinen and Tapanainen, 1998), largely developed from Constraint Grammar (CG) (Karlsson, 1990; Karlsson et al., 1995), and finally Topological Dependency Grammar (TDG) (Duchier and Debusmann, 2001), subsequently evolved into Extensible Dependency Grammar (XDG) (Debusmann et al., 2004). A synthesis of dependency grammar and categorial grammar is found in the framework of Dependency Grammar Logic (DGL) (Kruijff, 2001).

We will make no attempt at reviewing all these theories here. Instead, we will try to characterize their common core of assumptions, centered upon the notion of dependency, and discuss major points of divergence, such as the issue of projective versus non-projective representations.

2.1 The Notion of Dependency

The fundamental notion of *dependency* is based on the idea that the syntactic structure of a sentence consists of binary asymmetrical relations between the words of the sentence. The idea is expressed in the following way in the opening chapters of Tesnière (1959):

> La phrase est un *ensemble organisé* dont les éléments constituants sont les *mots*. [1.2] Tout mot qui fait partie d'une phrase cesse par lui-même d'être isolé comme dans le dictionnaire. Entre lui et ses voisins, l'esprit aperçoit des *connexions*, dont l'ensemble forme la charpente de la phrase. [1.3] Les connexions structurales établissent entre les mots des rapports de *dépendance*. Chaque connexion unit en principe un terme *supérieur* à un terme *inférieur*. [2.1] Le terme supérieur reçoit le nom de *régissant*. Le terme inférieur reçoit le nom de *subordonné*. Ainsi dans la phrase *Alfred parle* [...], *parle* est le régissant et *Alfred* le subordonné. [2.2] (Tesnière, 1959, 11–13, emphasis in the original)[1]

[1] English translation (by the author): 'The sentence is an *organized whole*, the constituent elements of which are *words*. [1.2] Every word that belongs to a sentence ceases by itself to be isolated as in the dictionary. Between the word and its neighbors, the mind perceives *connections*, the totality

In the terminology used in this paper, a dependency relation holds between a *head* and a *dependent*. Alternative terms in the literature are *governor* and *regent* for *head* (cf. Tesnière's *régissant*) and *modifier* for *dependent* (cf. Tesnière's *subordonné*).

Criteria for establishing dependency relations, and for distinguishing the head and the dependent in such relations, are clearly of central importance for dependency grammar. Such criteria have been discussed not only in the dependency grammar tradition, but also within other frameworks where the notion of syntactic head plays an important role, including all constituency-based frameworks that subscribe to some version of $\overline{\text{X}}$ theory (Chomsky, 1970; Jackendoff, 1977). Here are some of the criteria that have been proposed for identifying a syntactic relation between a head H and a dependent D in a construction C (Zwicky, 1985; Hudson, 1990):

1. H determines the syntactic category of C and can often replace C.
2. H determines the semantic category of C; D gives semantic specification.
3. H is obligatory; D may be optional.
4. H selects D and determines whether D is obligatory or optional.
5. The form of D depends on H (agreement or government).
6. The linear position of D is specified with reference to H.

It is clear that this list contains a mix of different criteria, some syntactic and some semantic, and one may ask whether there is a single coherent notion of dependency corresponding to all the different criteria. This has led some theorists, such as Hudson (1990), to suggest that the concept of head has a prototype structure, i.e. that typical instances of this category satisfy all or most of the criteria while more peripheral instances satisfy fewer. Other authors have emphasized the need to distinguish different kinds of dependency relations. According to Mel'čuk (1988), the word forms of a sentence can be linked by three types of dependencies: *morphological*, *syntactic* and *semantic*. According to Nikula (1986), we must distinguish between syntactic dependency in *endocentric* and *exocentric* constructions (Bloomfield, 1933).

---

of which forms the structure of the sentence. [1.3] The structural connections establish *dependency* relations between the words. Each connection in principle unites a *superior* term and an *inferior* term. [2.1] The superior term receives the name *governor*. The inferior term receives the name *subordinate*. Thus, in the sentence *Alfred parle* [...], *parle* is the governor and *Alfred* the subordinate. [2.2]'

Thus, in Figure 2, the NMOD relation holding between the noun *markets* and the adjective *financial* is an endocentric construction, where the head can replace the whole without disrupting the syntactic structure:

Economic news had little effect on [financial] markets. (1)

Endocentric constructions may satisfy all the criteria listed above, although number 4 is usually considered less relevant, since dependents in endocentric constructions are taken to be optional and not selected by their heads. By contrast, the PMOD relation holding between the preposition *on* and the noun *markets* is an exocentric construction, where the head cannot readily replace the whole:

Economic news had little effect on [markets]. (2)

Exocentric constructions, by their definition, fail on criterion number 1, at least with respect to subsitutability of the head for the whole, but they may satisfy the remaining criteria. Considering the rest of the relations exemplified in Figure 2, the SBJ and OBJ relations are clearly exocentric, and the NMOD relation from the noun *news* to the adjective *Economic* clearly endocentric, while the remaining NMOD relations (effect → little, effect → on) have a more unclear status.

The distinction between endocentric and exocentric constructions is also related to the distinction between *head-complement* and *head-modifier* (or *head-adjunct*) relations found in many contemporary syntactic theories, since head-complement relations are exocentric while head-modifier relations are endocentric. Many theories also recognize a third kind of relation, the *head-specifier* relation, typically exemplified by the determiner-noun relation, which is exocentric like the head-complement relation, but where there is no clear selection of the dependent element by the head.

The distinction between complements and modifiers is often defined in terms of *valency*, which is a central notion in the theoretical tradition of dependency grammar. Although the exact characterization of this notion differs from one theoretical framework to the other, valency is usually related to the semantic predicate-argument structure associated with certain classes of lexemes, in particular verbs but sometimes also nouns and adjectives. The idea is that the verb imposes requirements on its syntactic dependents that reflect its interpretation as a semantic predicate. Dependents that correspond to arguments of the predicate can be obligatory or optional in surface syntax but can only occur once with each predicate instance. By contrast, dependents that do not correspond to arguments can have more than one occurrence with a single predicate instance and tend to be optional. The *valency frame* of the verb is normally taken to include argument dependents, but some theories also allow obligatory non-arguments to be included (Sgall et al., 1986).

The notion of valency will not play a central role in the present paper, but we will sometimes use the terms *valency-bound* and *valency-free* to make a rough distinction between dependents that are more or less closely related to the semantic interpretation of the head. Returning to Figure 2, the subject and the object would normally be treated as valency-bound dependents of the verb *had*, while the adjectival modifiers of the nouns *news* and *markets* would be considered valency-free. The prepositional modification of the noun *effect* may or may not be treated as valency-bound, depending on whether the entity undergoing the effect is supposed to be an argument of the noun *effect* or not.

While head-complement and head-modifier structures have a fairly straightforward analysis in dependency grammar, there are also many constructions that have a relatively unclear status. This group includes constructions that involve grammatical function words, such as articles, complementizers and auxiliary verbs, but also structures involving prepositional phrases. For these constructions, there is no general consensus in the tradition of dependency grammar as to whether they should be analyzed as head-dependent relations at all and, if so, what should be regarded as the head and what should be regarded as the dependent. For example, some theories regard auxiliary verbs as heads taking lexical verbs as dependents; other theories make the opposite assumption; and yet other theories assume that verb chains are connected by relations that are not dependencies in the usual sense.

Another kind of construction that is problematic for dependency grammar (as for most theoretical traditions) is *coordination*. According to Bloomfield (1933), coordination is an endocentric construction, since it contains not only one but several heads that can replace the whole construction syntactically. However, this characterization raises the question of whether coordination can be analyzed in terms of binary asymmetrical relations holding between a head and a dependent. Again, this question has been answered in different ways by different theories within the dependency grammar tradition.

In conclusion, the theoretical tradition of dependency grammar is united by the assumption that an essential part of the syntactic structure of sentences resides in binary asymmetrical relations holding between lexical elements. Moreover, there is a core of syntactic constructions for which the analysis given by different frameworks agree in all important respects. However, there are also important differences with respect to whether dependency analysis is assumed to exhaust syntactic analysis, and with respect to the analysis of certain types of syntactic constructions. We will now turn to a discussion of some of the more important points of divergence in this tradition.

2.2 Varieties of Dependency Grammar

Perhaps the most fundamental open question in the tradition of dependency grammar is whether the notion of dependency is assumed to be not only *necessary* but also *sufficient* for the analysis of syntactic structure in natural language. This assumption is not made in the theory of Tesnière (1959), which is based on the three complementary concepts of *connection* (connexion), *junction* (jonction) and *transfer* (translation), where connection corresponds to dependency (cf. the quotation on page 3) but where junction and transfer are other kinds of relations that can hold between the words of a sentence. More precisely, junction is the relation that holds between coordinated items that are dependents of the same head or heads of the same dependent, while transfer is the relation that holds between a function word or other element that changes the syntactic category of a lexical element so that it can enter into different dependency relations. An example of the latter is the relation holding between the preposition *de* and *Pierre* in the construction *le livre de Pierre* (Pierre's book), where the preposition *de* allows the proper name *Pierre* to modify a noun, a dependency relation otherwise reserved for adjectives. Another way in which theories may depart from a pure dependency analysis is to allow a restricted form of constituency analysis, so that dependencies can hold between strings of words rather than single words. This possibility is exploited, to different degrees, in the frameworks of Hellwig (1986, 2003), Mel'čuk (1988) and Hudson (1990), notably in connection with coordination.

A second dividing line is that between mono-stratal and multi-stratal frameworks, i.e. between theories that rely on a single syntactic representation and theories that posit several layers of representation. In fact, most theories of dependency grammar, in contrast to frameworks for dependency parsing that will be discussed in Section 3, are multi-stratal, at least if semantic representations are considered to be a stratum of the theory. Thus, in FGD (Sgall et al., 1986) there is both an *analytical* layer, which can be characterized as a surface syntactic representation, and a *tectogrammatical* layer, which can be regarded as a deep syntactic (or shallow semantic) representation. In a similar fashion, MTT (Mel'čuk, 1988) recognizes both *surface syntactic* and *deep syntactic* representations (in addition to representations of deep phonetics, surface morphology, deep morphology and semantics). By contrast, Tesnière (1959) uses a single level of syntactic representation, the so-called *stemma*, which on the other hand includes junction and transfer in addition to syntactic connection.[2] The framework of XDG (Debusmann et al., 2004) can be seen as a compromise in that it allows multiple layers of dependency-based linguistic representations but requires that all layers, or *dimensions* as they are called in

[2]Tesnière's representations also include *anaphors*, which are described as supplementary semantic connections without corresponding syntactic connections.

XDG, share the same set of nodes. This is in contrast to theories like FGD, where e.g. function words are present in the analytical layer but not in the tectogrammatical layer.

The different requirements of XDG and FGD point to another issue, namely what can constitute a node in a dependency structure. Although most theories agree that dependency relations hold between *lexical* elements, rather than *phrases*, they can make different assumptions about the nature of these elements. The most straightforward view is that the nodes of the dependency structure are simply the word forms occurring in the sentence, which is the view adopted in most parsing systems based on dependency grammar. But it is also possible to construct dependency structures involving more abstract entities, such as lemmas or lexemes, especially in deeper syntactic representations. Another variation is that the elements may involve several word forms, constituting a *dissociate nucleus* (nucléus dissocié) in the terminology of Tesnière (1959), or alternatively correspond to smaller units than word forms, as in the morphological dependencies of Mel'čuk (1988).

A fourth dimension of variation concerns the inventory of specific dependency types posited by different theories, i.e. functional categories like SBJ, OBJ and NMOD that are used to label dependency arcs in the representation in Figure 2. Broadly speaking, most theories either assume a set of more surface-oriented grammatical functions, such as *subject*, *object*, *adverbial*, etc., with a more or less elaborate subclassification, or a set of more semantically oriented role types, such as *agent*, *patient*, *goal*, etc., belonging to the tradition of *case roles* or *thematic roles* (Fillmore, 1968; Jackendoff, 1972; Dowty, 1989).[3] Multi-stratal theories often combine the two relation types. Thus, FGD (Sgall et al., 1986) uses grammatical functions in the analytical layer and semantic roles in the tectogrammatical layer. An alternative scheme of representation, which is found in MTT (Mel'čuk, 1988), is to use numerical indices for valency-bound dependents to reflect a canonical ordering of arguments (argument 1, 2, 3, etc.) and to use descriptive labels only for valency-free dependents. Finally, it is also possible to use unlabeled dependency structures, although this is more common in practical parsing systems than in linguistic theories.

There are several open issues in dependency grammar that have to do with formal properties of representations. Since a dependency representation consists of lexical elements linked by binary asymmetrical relations, it can be defined as a *labeled directed graph*, where the set of nodes (or vertices) is the set of lexical elements (as defined by the particular framework), and the set of labeled arcs

[3]The notion of a semantic role can be traced back to Pāṇini's *kānaka* theory (Misra, 1966), which is sometimes also seen as the earliest manifestation of dependency grammar. The notion of a grammatical function also has a long history that extends at least to the work of Appolonius Dyscolus in the second century of the Common Era (Robins, 1967).

represent dependency relations from heads to dependents. In order to provide a complete syntactic analysis of a sentence, the graph must also be *connected* so that every node is related to at least one other node (Mel'čuk, 1988). Again, we refer to Figure 2 as an illustration of this representation, where the nodes are the word tokens of the sentence (annotated with parts-of-speech) and the arcs are labeled with grammatical functions.[4]

Given this general characterization, we may then impose various additional conditions on these graphs. Two basic constraints that are assumed in most versions of dependency grammar are the *single-head* constraint, i.e. the assumption that each node has at most one head, and the *acyclicity* constraint, i.e. the assumption that the graph should not contain cycles. These two constraints, together with connectedness, imply that the graph should be a rooted tree, with a single root node that is not a dependent of any other node. For example, the representation in Figure 2 is a rooted tree with the verb *had* as the root node. Although these constraints are assumed in most versions of dependency grammar, there are also frameworks that allow multiple heads as well as cyclic graphs, such as WG (Hudson, 1984, 1990). Another issue that arises for multi-stratal theories is whether each level of representation has its own set of nodes, as in most theories, or whether they only define different arc sets on top of the same set of nodes, as in XDG (Debusmann et al., 2004).

However, the most important and hotly debated issues concerning formal representations have to do with the relation between dependency structure and word order. According to Tesnière (1959), dependency relations belong to the *structural order* (l'ordre structural), which is different from the *linear order* (l'ordre linéaire) of a spoken or written string of words, and *structural syntax* is based on the relations that exist between these two dimensions. Most versions of dependency grammar follow Tesnière in assuming that the nodes of a dependency structure are not linearly ordered in themselves but only in relation to a particular surface realization of this structure. A notable exception to this generalization is FGD, where the representations of both the analytical layer and the tectogrammatical layer are linearly ordered in order to capture aspects of information structure (Sgall et al., 1986). In addition, there are frameworks, such as TDG (Duchier and Debusmann, 2001), where the linear order is represented by means of a linearly ordered dependency structure, the Linear Precedence (LP) tree, while the proper dependency representation, the Immediate Dominance (ID) tree, is unordered.

[4]There seems to be no general consensus in the literature on dependency grammar as to whether the arcs representing dependency relations should be drawn pointing from heads to dependents or vice versa (or indeed with arrowheads at all). We have chosen to adopt the former alternative, both because it seems to be the most common representation in the literature and because it is consistent with standard practice in graph theory.

However, whether dependency relations introduce a linear ordering or not, there may be constraints relating dependency structures to linear realizations. The most well-known example is the constraint of *projectivity*, first discussed by Lecerf (1960), Hays (1964) and Marcus (1965), which is related to the contiguity constraint for constituent representations. A dependency graph satisfies the constraint of projectivity with respect to a particular linear order of the nodes if, for every arc $h \rightarrow d$ and node w, w occurs between h and d in the linear order only if w is dominated by h (where *dominates* is the reflexive and transitive closure of the arc relation). For example, the representation in Figure 2 is an example of a *projective* dependency graph, given the linear order imposed by the word order of the sentence.

The distinction between *projective* and *non-projective* dependency grammar often made in the literature thus refers to the issue of whether this constraint is assumed or not. Broadly speaking, we can say that whereas most practical systems for dependency parsing do assume projectivity, most dependency-based linguistic theories do not. More precisely, most theoretical formulations of dependency grammar regard projectivity as the norm but also recognize the need for non-projective representations of certain linguistic constructions, e.g. long-distance dependencies (Mel'čuk, 1988; Hudson, 1990). It is also often assumed that the constraint of projectivity is too rigid for the description of languages with free or flexible word order.

Some multi-stratal theories allow non-projective representations in some layers but not in others. For example, FGD assumes that tectogrammatical representations are projective while analytical representations are not (Sgall et al., 1986). Similarly, TDG (Duchier and Debusmann, 2001) assume projectivity for LP trees but not for ID trees. Sometimes a weaker condition called *planarity* is assumed, which allows a node w to occur between a head h and a dependent d without being dominated by h only if w is a root (Sleator and Temperley, 1993).[5] Further relaxations of these constraints are discussed in Kahane et al. (1998) and Yli-Jyrä (2003).

The points of divergence considered up till now have all been concerned with aspects of representation. However, as mentioned at the end of the previous section, there are also a number of points concerning the substantive linguistic analysis where different frameworks of dependency grammar make different assumptions, in the same way as theories differ also within other traditions. We will limit ourselves to a brief discussion of two such points.

The first point concerns the issue of *syntactic* versus *semantic* heads. As noted in Section 2.1, the criteria for identifying heads and dependents invoke both syn-

[5] This constraint is related to but not equivalent to the standard notion of planarity in graph theory (see, e.g., Grimaldi, 2004).

tactic and semantic properties. In many cases, these criteria give the same result, but in others they diverge. A typical example is found in so-called case marking prepositions, exemplified in the following sentence:

I believe in the system. (3)

According to syntactic criteria, it is natural to treat the preposition *in* as a dependent of the verb *believe* and as the head of the noun *system*. According to semantic criteria, it is more natural to regard *system* as a direct dependent of *believe* and to treat *in* as a dependent of *system* (corresponding to a case marking affix in some other languages).[6] Most versions of dependency grammar treat the preposition as the head of the noun, but there are also theories that make the opposite assumption. Similar considerations apply to many constructions involving one function word and one content word, such as determiner-noun and complementizer-verb constructions. An elegant solution to this problem is provided by the theory of Tesnière (1959), according to which the function word and the content word form a *dissociate nucleus* (nucléus dissocié), united by a relation of *transfer* (translation). In multi-stratal theories, it is possible to treat the function word as the head only in more surface-oriented layers. For example, to return to example (3), FGD would assume that the preposition takes the noun as a dependent in the analytical layer, but in the tectogrammatical layer the preposition would be absent and the noun would instead depend directly on the verb.

The second point concerns the analysis of coordination, which presents problems for any syntactic theory but which seems to be especially hard to reconcile with the idea that syntactic constructions should be analyzed in terms of binary head-dependent relations. Consider the following example:

They operate ships and banks. (4)

It seems clear that the phrase *ships and banks* functions as a direct object of the verb *operate*, but it is not immediately clear how this phrase can be given an internal analysis that is compatible with the basic assumptions of dependency analysis, since the two nouns *ships* and *banks* seem to be equally good candidates for being heads. One alternative is to treat the conjunction as the head, as shown in Figure 3 (*top*), an analysis that may be motivated on semantic grounds and is adopted in FGD. Another alternative, advocated by Mel'čuk (1988), is to treat the conjunction as the head only of the second conjunct and analyze the conjunction as a dependent of the first conjunct, as shown in Figure 3 (*bottom*). The arguments for this analysis are essentially the same as the arguments for an asymmetric right-branching

[6] A third alternative is to treat both *in* and *system* as dependents of *believe*, since it is the verb that selects the preposition and takes the noun as an argument.

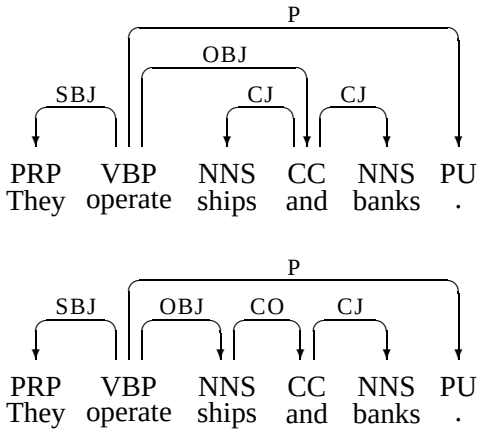

Figure 3: Two analyses of coordination

analysis in constituency-based frameworks. A third option is to give up a pure dependency analysis and allow a limited form of phrase structure, as in WG (Hudson, 1990). A fourth and final variant is the analysis of Tesnière (1959), according to which both *ships* and *banks* are dependents of the verb, while the conjunction marks a relation of *junction* (jonction) between the two nouns.

3 Parsing with Dependency Representations

So far, we have reviewed the theoretical tradition of dependency grammar, focusing on the common core of assumptions as well as major points of divergence, rather than on individual instantiations of this tradition. We will now turn to what is the main topic of this paper, namely the computational implementation of syntactic analysis based on dependency representations, i.e. representations involving lexical nodes, connected by dependency arcs, possibly labeled with dependency types.

Such implementations may be intimately tied to the development of a particular theory, such as the PLAIN system based on DUG (Hellwig, 1980, 2003) or the FDG parsing system (Tapanainen and Järvinen, 1997; Järvinen and Tapanainen, 1998). On the whole, however, the connections between theoretical frameworks and computational systems are often rather indirect for dependency-based analysis, probably more so than for theories and parsers based on constituency analysis. This may be due to the relatively lower degree of formalization of dependency grammar theories in general, and this is also part of the reason why the topic of this section

is described as parsing with dependency *representations*, rather than parsing with dependency *grammar*.

In discussing dependency-based systems for syntactic parsing, we will follow Carroll (2000) and distinguish two broad types of strategy, the *grammar-driven approach* and the *data-driven approach*, although these approaches are not mutually exclusive. We will conclude the paper with a brief discussion of some of the potential advantages of using dependency representations in syntactic parsing.

3.1 Grammar-Driven Dependency Parsing

The earliest work on parsing with dependency representations was intimately tied to formalizations of dependency grammar that were very close to context-free grammar, such as the proposals of Hays (1964) and Gaifman (1965). In the formulation of Gaifman (1965) a *dependency system* contains three sets of rules:[7]

1. L_{I}: Rules of the form $X(Y_1 \cdots Y_i * Y_{i+1} \cdots Y_n)$, where i may equal 0 and/or n, which say that the category X may occur with categories $Y_1, \ldots, Y_n$ as dependents, in the order given (with X in position $*$).

2. L_{II}: Rules giving for every category X the list of words belonging to it (where each word may belong to more than one category).

3. L_{III}: A rule giving the list of all categories the occurrence of which may govern a sentence.

A sentence consisting of words $w_1, \ldots, w_n$ is analyzed by assigning to it a sequence of categories $X_1, \ldots, X_n$ and a relation of dependency d between words such that the following conditions hold (where d^* is the transitive closure of d):

1. For no w_i, $d^*(w_i, w_i)$.

2. For every w_i, there is at most one w_j such that $d(w_i, w_j)$.

3. If $d^*(w_i, w_j)$ and w_k is between w_i and w_j, then $d^*(w_k, w_j)$.

4. The whole set of word occurrences is connected by d.

5. If $w_1, \ldots, w_i$ are left dependents and $w_{i+1}, \ldots, w_n$ right dependents of some word, and $X_1, \ldots, X_i, X_{i+1}, \ldots, X_n$ are the categories of $w_1, \ldots, w_i, w_{i+1}, \ldots, w_n$, then $X(X_1 \cdots X_i * X_{i+1} \cdots X_n)$ is a rule of L_{I}.

6. The word occurrence w_i that governs the sentence belongs to a category listed in L_{III}.

[7] The formulation of Hays (1964) is slightly different but equivalent in all respects.

Gaifman remarks that 1–4 are general structure requirements that can be made on any relation defined on a finite linearly ordered set whether it is a set of categories or not, while 5–6 are requirements which relate the relation to the specific grammar given by the three sets of rules L_{I}–L_{III}. Referring back to the discussion of graph conditions in Section 2.2, we may first of all note that Gaifman defines dependency relations to hold from dependent to head, rather than the other way round which is more common in the recent literature. Secondly, we see that condition 2 corresponds to the *single-head* constraint and condition 3 to the *projectivity* constraint. Conditions 1, 2 and 4 jointly entail that the graph is a rooted tree, which is presupposed in condition 6. Finally, it should be pointed out that this kind of dependency system only gives an unlabeled dependency analysis, since there are no dependency types used to label dependency relations.

Gaifman (1965) proves several equivalence results relating his dependency systems to context-free grammars. In particular, he shows that the two systems are weakly equivalent, i.e. that they both characterize the class of context-free languages. However, he also shows that whereas any dependency system can be converted to a strongly equivalent context-free grammar (modulo a specific mapping between dependency trees and context-free parse trees), the inverse construction is only possible for a restricted subset of context-free grammar (roughly grammars where all productions are lexicalized).

These results have been invoked to explain the relative lack of interest in dependency grammar within natural language processing for the subsequent twenty-five years or so, based on the erroneous conclusion that dependency grammar is only a restricted variant of context-free grammar (Järvinen and Tapanainen, 1998).[8] This conclusion is erroneous simply because the results only concern the specific version of dependency grammar formalized by Hays and Gaifman, which for one thing is restricted to projective dependency structures. However, it also worth emphasizing that with the increasing importance of problems like robustness and disambiguation, issues of (limited) generative capacity have lost some of their significance in the context of natural language parsing. Nevertheless, it seems largely true to say that, except for isolated studies of dependency grammar as an alternative to context-free grammar as the basis for transformational grammar (Robinson, 1970), dependency grammar has played a marginal role both in syntactic theory and in natural language parsing until fairly recently.

The close relation to context-free grammar in the formalization of dependency grammar by Hays and Gaifman means that essentially the same parsing methods

[8] A similar development seems to have taken place with respect to categorial grammar after the weak equivalence of a restricted type of categorial grammar with context-free grammar was proven by Bar-Hillel et al. (1960).

can be used for both types of system. Hence, the parsing algorithm outlined in Hays (1964) is a bottom-up dynamic programming algorithm very similar to the CKY algorithm proposed for context-free parsing at about the same time (Kasami, 1965; Younger, 1967). The use of dynamic programming algorithms that are closely connected to context-free parsing algorithms such as CKY and Earley's algorithm (Earley, 1970) is a prominent trend also in more recent grammar-driven approaches to dependency parsing. One example is the link grammar parser of Sleator and Temperley (1991, 1993), which uses a dynamic programming algorithm implemented as a top-down recursive algorithm with memoization to achieve parsing in $O(n^3)$ time. Link grammar is not considered an instance of dependency grammar by its creators, and it departs from the traditional view of dependency by using undirected links, but the representations used in link grammar parsing are similar to dependency representations in that they consist of words linked by binary relations. Other examples include a modification of the CKY algorithm (Holan et al., 1997) and an Earley-type algorithm with left-corner filtering in the prediction step (Lombardo and Lesmo, 1996; Barbero et al., 1998).

A common property of all frameworks that implement dependency parsing as a form of lexicalized context-free parsing is that they are restricted to the derivation of projective dependency structures, although some of the frameworks allow post-processing that may introduce non-projective structures (Sleator and Temperley, 1991, 1993). Many of these frameworks can be subsumed under the notion of *bilexical grammar* introduced by Eisner (2000). In Eisner's formulation, a bilexical grammar consists of two elements:

1. A vocabulary V of terminal symbols (words), containing a distinguished symbol ROOT.

2. For each word $w \in V$, a pair of deterministic finite-state automata l_w and r_w. Each automaton accepts some regular subset of V^*.

The language $L(G)$ defined by a bilexical dependency grammar G is defined as follows:

1. A *dependency tree* is a rooted tree whose nodes are labeled with words from V, and where the root node is labeled with the special symbol ROOT. The children of a node are ordered with respect to each other and the node itself, so that the node has both *left children* that precede it and *right children* that follow it.

2. A dependency tree is *grammatical* according to G iff for every word token w that appears in the tree, l_w accepts the (possibly empty) sequence of w's

left children (from right to left), and r_w accepts the sequence of w's right children (from left to right).

3. A string $x \in V^*$ is generated by G with analysis y if y is a grammatical dependency tree according to G and listing the node labels of y in infix order yields the string x followed by ROOT; x is called the *yield* of y.

4. The language $L(G)$ is the set of all strings generated by G.

The general parsing algorithm proposed by Eisner for bilexical grammar is again a dynamic programming algorithm, which proceeds by linking *spans* (strings where roots occur either leftmost or rightmost or both) instead of *constituents*, thereby reducing the time complexity from $O(n^5)$ to $O(n^3)$. More precisely, the running time is $O(n^3g^3t)$, where g is an upper bound on the number of possible senses (lexical entries) of a single word, and t is an upper bound on the number of states of a single automaton.

Eisner shows how the framework of bilexical grammar, and the cubic-time parsing algorithm, can be modified to capture a number of different frameworks and approaches such as Milward's (mono)lexical dependency grammar (Milward, 1994), Alshawi's head automata (Alshawi, 1996), Sleator and Temperley's link grammar (Sleator and Temperley, 1991, 1993), and Eisner's own probabilistic dependency models that will be discussed below in Section 3.2 (Eisner, 1996b,a).

The second main tradition in grammar-driven dependency parsing is based on the notion of *eliminative* parsing, where sentences are analyzed by successively eliminating representations that violate constraints until only valid representations remain. One of the first parsing systems based on this idea is the CG framework (Karlsson, 1990; Karlsson et al., 1995), which uses underspecified dependency structures represented as syntactic tags and disambiguated by a set of constraints intended to exclude ill-formed analyses. In CDG (Maruyama, 1990), this idea is extended to complete dependency structures by generalizing the notion of tag to pairs consisting of a syntactic label and an identifier of the head node. This kind of representation is fundamental for many different approaches to dependency parsing, since it provides a way to reduce the parsing problem to a tagging or classification problem. Typical representatives of this tradition are the extended CDG framework of Harper and Helzerman (1995) and the FDG system (Tapanainen and Järvinen, 1997; Järvinen and Tapanainen, 1998), where the latter is a development of CG that combines eliminative parsing with a non-projective dependency grammar inspired by Tesnière (1959).

In the eliminative approach, parsing is viewed as a constraint satisfaction problem, where any analysis satisfying all the constraints of the grammar is a valid

analysis. Constraint satisfaction in general is NP complete, which means that special care must be taken to ensure reasonable efficiency in practice. Early versions of this approach used procedures based on local consistency (Maruyama, 1990; Harper et al., 1995), which attain polynomial worst case complexity by only considering local information in the application of constraints. In the more recently developed TDG framework (Duchier, 1999, 2003), the problem is confronted head-on by using constraint programming to solve the satisfaction problem defined by the grammar for a given input string. The TDG framework also introduces several levels of representation (cf. Section 2.2), arguing that constraints can be simplified by isolating different aspects of the grammar such as Immediate Dominance (ID) and Linear Precedence (LP) and have constraints that relate different levels to each other (Duchier and Debusmann, 2001; Debusmann, 2001). This view is taken to its logical extension in the most recent version of the framework called Extensible Dependency Grammar (XDG), where any number of levels, or dimensions, can be defined in the grammatical framework (Debusmann et al., 2004)

From the point of view of parsing unrestricted text, parsing as constraint satisfaction can be problematic in two ways. First, for a given input string, there may be no analysis satisfying all constraints, which leads to a robustness problem. Secondly, there may be more than one analysis, which leads to a problem of disambiguation. Menzel and Schröder (1998) extends the CDG framework of Maruyama (1990) with *graded*, or *weighted*, constraints, by assigning a weight w ($0.0 \leq w \leq 1.0$) to each constraint indicating how serious the violation of this constraint is (where 0.0 is the most serious). In this extended framework, later developed into WCDG (Schröder, 2002), the best analysis for a given input string is the analysis that minimizes the total weight of violated constraints. While early implementations of this system used an eliminative approach to parsing (Menzel and Schröder, 1998), the more recent versions instead use a transformation-based approach, which successively tries to improve the analysis by transforming one solution into another guided by the observed constraint violations in the current solution. One advantage of this heuristic approximation strategy is that it can be combined with arbitrarily complex constraints, whereas standard eliminative procedures usually require constraints to be binary for efficiency reasons (Foth et al., 2004).

So far, we have distinguished two main trends in grammar-driven dependency parsing. The first is based on a formalization of dependency grammar that is closely related to context-free grammar, and therefore usually restricted to projective dependency structures, using standard techniques from context-free parsing to obtain good efficiency in the presence of massive ambiguity, in particular dynamic programming or memoization. The second is based on a formalization of dependency grammar in terms of constraints, not necessarily limited to projective structures,

where parsing is naturally viewed as a constraint satisfaction problem which can be addressed using eliminative parsing methods, although the exact parsing problem is often intractable.

In addition to these two traditions, which both involve fairly complex grammars and parsing algorithms, there is a third tradition which is based on a simpler notion of dependency grammar together with a deterministic parsing strategy (possibly with limited backtracking). As in other parsing paradigms, the study of deterministic parsing can be motivated either by a wish to model human sentence processing or by a desire to make syntactic parsing more efficient (or possibly both). According to Covington (2001), these methods have been known since the 1960's without being presented systematically in the literature. The fundamental parsing strategy comes in different versions but we will concentrate here on the left-to-right (or incremental) version, which is formulated in the following way by Covington (2001):

> Accept words one by one starting at the beginning of the sentence, and try linking each word as head or dependent of every previous word.

This parsing strategy is compatible with many different grammar formulations. All that is required is that a grammar G defines a boolean function f_G that, for any two words w_1 and w_2, returns **true** if w_1 can be the head of w_2 according to G (and **false**) otherwise.[9] Covington (2001) demonstrates how this parsing strategy can be used to produce dependency structures satisfying different conditions such as *uniqueness* (single head) and *projectivity* simply by imposing different constraints on the linking operation. Covington has also shown in previous work how this parsing strategy can be adapted to suit languages with free, flexible or rigid word order (Covington, 1990a,b, 1994). The time complexity of Covington's algorithm is $O(n^2)$ in the deterministic version.

The parsing algorithm proposed by Nivre (2003), which will be discussed in Section 3.2, can be derived as a special case of Covington's algorithm, although we will not give this formulation here, and the very first experiments with this algorithm used a simple grammar of the kind presupposed by Covington to perform unlabeled dependency parsing (Nivre, 2003). A similar approach can be found in Obrebski (2003), although this system is nondeterministic and derives a compact representation of all permissible dependency trees in the form of a directed acyclic graph. Yet another framework that shows affinities with the deterministic grammar-driven approach is that of Kromann (2004), although it is based on a

[9] In this formulation, the parsing strategy is limited to unlabeled dependency graphs. In principle, it is possible to perform labeled dependency parsing by returning a set of permissible dependency types instead of **true**, but this makes the process nondeterministic in general.

more sophisticated notion of grammar called Discontinuous Grammar. Parsing in this framework is essentially deterministic but subject to repair mechanisms that are associated with local cost functions derived from the grammar.

Before we close the discussion of grammar-driven dependency parsing, we should also mention the work of Oflazer (2003), which is an extended finite-state approach to dependency parsing similar to the cascaded partial parsers used for constituency-based parsing by Abney (1996) and Roche (1997). Oflazer's system allows violable constraints for robust parsing and uses total link length to rank alternative analyses, as proposed by Lin (1996).

3.2 Data-Driven Dependency Parsing

As for natural language parsing in general, the first attempts at data-driven dependency parsing were also grammar-driven in that they relied on a formal dependency grammar and used corpus data to induce a probabilistic model for disambiguation. Thus, Carroll and Charniak (1992) essentially use a PCFG model, where the context-free grammar is restricted to be equivalent to a Hays/Gaifman type dependency grammar. They report experiments trying to induce such a probabilistic grammar using unsupervised learning on an artificially created corpus but with relatively poor results.

A more successful and more influential approach was developed by Eisner (1996a,b), who defined several probabilistic models for dependency parsing and evaluated them using supervised learning with data from the Wall Street Journal section of the Penn Treebank. In later work, Eisner (2000) has shown how these models can be subsumed under the general notion of a *bilexical grammar* (BG), which means that parsing can be performed efficiently as discussed in Section 3.1. Eisner (2000) defines the notion of a *weighted bilexical grammar* (WBG) in terms of BG as follows (cf. Section 3.1):

1. A *weighted DFA* A is a deterministic finite automaton that associates a real-valued *weight* with each arc and each final state. Each accepting path through A is assigned a weight, namely the sum of all arc weights on the path and the weight of the final state. Each string x accepted by A is assigned the weight of its accepting path.

2. A WBG G is a BG in which all the automata l_w and r_w are weighted DFAs. The weight of a dependency tree y under G is defined as the sum, over all word tokens w in y, of the weight with which l_w accepts w's sequence of left children plus the weight with which r_w accepts w's sequence of right children.

Eisner (1996b) presents three different probabilistic models for dependency parsing, which can be reconstructed as different weighting schemes within the framework of WBG. However, the first two models (models A and B) require that we distinguish between an underlying string $x \in V^*$, described by the WBG, and a surface string x', which results from a possibly nondeterministic, possibly weighted finite-state transduction R on x. The surface string x' is then grammatical with analysis (y, p) if y is a grammatical dependency tree whose yield x is transduced to x' along an accepting path p in R. To avoid the distinction between underlying strings and surface strings, we will restrict our attention to model C, which was found to perform significantly better than the other two models in the experiments reported in Eisner (1996a).

First of all, it should be pointed out that all the models in Eisner (1996b) involve part-of-speech tags, in addition to word tokens and (unlabeled) dependency relations, and define the joint probability of the words, tags and dependency links. Model C is defined as follows:

$$P(tw(1), \ldots, tw(n), links) = \prod_{i=1}^{n} P(lc(i) \,|\, tw(i)) P(rc(i) \,|\, tw(i)) \tag{5}$$

where $tw(i)$ is the ith tagged word, and $lc(i)$ and $rc(i)$ are the left and right children of the ith word, respectively. The probability of generating each child is conditioned on the tagged head word and the tag of the preceding child (left children being generated from right to left):

$$P(lc(i) \,|\, tw(i)) = \prod_{j=1}^{m} P(tw(lc_j(i)) \,|\, t(lc_{j-1}(i)), tw(i)) \tag{6}$$

$$P(rc(i) \,|\, tw(i)) = \prod_{j=1}^{m} P(tw(rc_j(i)) \,|\, t(rc_{j-1}(i)), tw(i)) \tag{7}$$

where $lc_j(i)$ is the jth left child of the ith word and $t(lc_{j-1}(i))$ is the tag of the preceding left child (and analogously $rc_j(i)$ and $t(rc_{j-1}(i))$ for right children). This model can be implemented in the WBG framework by letting the automata l_w and r_w have weights on their arcs corresponding to the log of the probability of generating a particular left or right child given the tag of the preceding child. In this way, the weight assigned to a dependency tree T will be the log of $P(tw(1), \ldots, tw(n), links)$ as defined above (Eisner, 2000).

Eisner's work on data-driven dependency parsing has been influential in two ways. First, it showed that generative probabilistic modeling and supervised learning could be applied to dependency representations to achieve a parsing accuracy comparable to the best results reported for constituency-based parsing at the time,

although the evalutation metrics used in the two cases are not strictly comparable. Secondly, it showed how these models could be coupled with efficient parsing techniques that exploit the special properties of dependency structures. The importance of the second aspect can be seen in recent work by McDonald et al. (2005), applying discriminative estimation methods to probabilistic dependency parsing. Thanks to the more efficient parsing methods offered by Eisner's methods for bilexical parsing, training can be performed without pruning the search space, which is impossible for efficiency reasons when using lexicalized constituency representations with comparable lexical dependencies.

Collins et al. (1999) apply the generative probabilistic parsing models of Collins (1997, 1999) to dependency parsing, using data from the Prague Dependency Treebank. This requires preprocessing to transform dependency structures into flat phrase structures for the training phase and postprocessing to extract dependency structures from the phrase structures produced by the parser. The parser of Charniak (2000) has been adapted and applied to the Prague Dependency Treebank in a similar fashion, although this work remains unpublished.

Samuelsson (2000) proposes a probabilistic model for dependency grammar that goes beyond the models considered so far by incorporating labeled dependencies and allowing non-projective dependency structures. In this model, dependency representations are generated by two stochastic processes: one top-down process generating the tree structure y and one bottom-up process generating the surface string x given the tree structure, defining the joint probability as $P(x, y) = P(y)P(x|y)$. Samuelsson suggests that the model can be implemented using a bottom-up dynamic programming approach, but the model has unfortunately never been implemented and evaluated.

Another probabilistic approach to dependency parsing that incorporates labeled dependencies is the stochastic CDG parser of Wang and Harper (2004), which extends the CDG model with a generative probabilistic model. Parsing is performed in two steps, which may be tightly or loosely integrated, where the first step assigns to each word a set of SuperARVs, representing constraints on possible heads and dependents, and where the second step determines actual dependency links given the SuperARV assignment. Although the basic model and parsing algorithm is limited to projective structures, the system allows non-projective structures for certain *wh*-constructions. The system has been evaluated on data from the Wall Street Journal section of the Penn Treebank and achieves state-of-the-art performance using a dependency-based evaluation metric (Wang and Harper, 2004).

The first step in the parsing model of Wang and Harper (2004) can be seen as a kind of supertagging, which has turned out to be a crucial element in many recent approaches to statistical parsing, e.g. in LTAG (Joshi and Sarkar, 2003; Bangalore, 2003) and CCG (Clark and Curran, 2004; Curran and Clark, 2004). A similar

two-step process is used in the statistical dependency parser of Bangalore (2003), which uses elementary LTAG trees as supertags in order to derive a dependency structure in the second step. Supertagging is performed using a standard HMM trigram tagger, while dependency structures are derived using a heuristic deterministic algorithm that runs in linear time. Another data-driven dependency parser based on supertagging is Nasr and Rambow (2004), where supertags are derived from a lexicalized extended context-free grammar and the most probable analysis is derived using a modified version of the CKY algorithm.

Most of the systems described in this section are based on a formal dependency grammar in combination with a generative probabilistic model, which means that parsing conceptually consists in the derivation of all analyses that are permissible according to the grammar and the selection of the most probable analysis according to the generative model. This is in contrast to recent work based on purely discriminative models of inductive learning in combination with a deterministic parsing strategy, methods that do not involve a formal grammar.

The deterministic discriminative approach was first proposed by Kudo and Matsumoto (2000, 2002) and Yamada and Matsumoto (2003), using support vector machines (Vapnik, 1995) to train classifiers that predict the next action of a deterministic parser constructing unlabeled dependency structures. The parsing algorithm used in these systems implements a form of shift-reduce parsing with three possible parse actions that apply to two neighboring words referred to as *target nodes*:[10]

1. A *Shift* action adds no dependency construction between the target words w_i and w_{i+1} but simply moves the point of focus to the right, making w_{i+1} and w_{i+2} the new target words.

2. A *Right* action constructs a dependency relation between the target words, adding the left node w_i as a child of the right node w_{i+1} and reducing the target words to w_{i+1}, making w_{i-1} and w_{i+1} the new target words.

3. A *Left* action constructs a dependency relation between the target words, adding the right node w_{i+1} as a child of the left node w_i and reducing the target words to w_i, making w_{i-1} and w_i the new target words.

The parser processes the input from left to right repeatedly as long as new dependencies are added, which means that up to $n - 1$ passes over the input may be required to construct a complete dependency tree, giving a worst case time complexity of $O(n^2)$, although the worst case seldom occurs in practice. The features

[10]The parsers described in Kudo and Matsumoto (2000, 2002) are applied to Japanese, which is assumed to be strictly head-final, which means that only the actions *Shift* and *Right* are required.

used to predict the next parse action are the word forms and part-of-speech tags of the target words, of their left and right children, and of their left and right string context (in the reduced string). Yamada and Matsumoto (2003) evaluate the system using the standard data set from the Wall Street Journal section of the Penn Treebank and shows that deterministic discriminative dependency parsing can achieve an accuracy that is close to the state-of-the-art with respect to dependency accuracy. Further improvements with this model are reported in Isozaki et al. (2004).

The framework of inductive dependency parsing, first presented in Nivre et al. (2004) and more fully described in Nivre (2005), has many properties in common with the system of Yamada and Matsumoto (2003), but there are three differences. The first and most important difference is that the system of Nivre et al. (2004) constructs labeled dependency representations, i.e. representations where dependency arcs are labeled with dependency types. This also means that dependency type information can be exploited in the feature model used to predict the next parse action. The second difference is that the algorithm proposed in Nivre (2003) is a head-driven arc-eager algorithm that constructs a complete dependency tree in a single pass over the data. The third and final difference is that Nivre et al. (2004) use memory-based learning to induce classifiers for predicting the next parsing action based on conditional features, whereas Yamada and Matsumoto (2003) use support vector machines. However, as pointed out by Kudo and Matsumoto (2002), in a deterministic discriminative parser the learning method is largely independent of the rest of the system.

4 The Case for Dependency Parsing

As noted several times already, dependency-based syntactic representations have played a fairly marginal role in the history of linguistic theory as well as that of natural language processing. Saying that there is increasing interest in dependency-based approaches to syntactic parsing may therefore not be saying very much, but it is hard to deny that the notion of dependency has become more prominent in the literature on syntactic parsing during the last decade or so.

In conclusion, it therefore seems appropriate to ask what are the potential benefits of using dependency-based representations in syntactic parsing, as opposed to the more traditional representations based on constituency. According to Covington (2001), dependency parsing offers three *prima facie* advantages:

- Dependency links are close to the semantic relationships needed for the next stage of interpretation; it is not necessary to "read off" head-modifier or head-complement relations from a tree that does not show them directly.

- The dependency tree contains one node per word. Because the parser's job is only to connect existing nodes, not to postulate new ones, the task of parsing is in some sense more straightforward. [...]

- Dependency parsing lends itself to word-at-a-time operation, i.e., parsing by accepting and attaching words one at a time rather than by waiting for complete phrases. [...]

To this it is sometimes added that dependency-based parsing allows a more adequate treatment of languages with variable word order, where discontinuous syntactic constructions are more common than in languages like English (Mel'čuk, 1988; Covington, 1990b). However, this argument is only plausible if the formal framework allows non-projective dependency structures, which is not the case for most dependency parsers that exist today.

For us, the first two advantages identified by Covington seem to be the most important. Having a more constrained representation, where the number of nodes is fixed by the input string itself, should enable conceptually simpler and computationally more efficient methods for parsing. At the same time, it is clear that a more constrained representation is a less expressive representation and that dependency representations are necessarily underspecified with respect to certain aspects of syntactic structure. For example, as pointed out by Mel'čuk (1988), it is impossible to distinguish in a pure dependency representation between an element modifying the head of a phrase and the same element modifying the entire phrase. However, this is precisely the kind of ambiguity that is notoriously hard to disambiguate correctly in syntactic parsing anyway, so it might be argued that this kind of underspecification is actually beneficial. And as long as the syntactic representation encodes enough of the structural relations that are relevant for semantic interpretation, then we are only happy if we can constrain the problem of deriving these representations.

In general, there is a tradeoff between the expressivity of syntactic representations and the complexity of syntactic parsing, and we believe that dependency representations provide a good compromise in this respect. They are less expressive than most constituency-based representations, but they compensate for this by providing a relatively direct encoding of predicate-argument structure, which is relevant for semantic interpretation, and by being composed of bilexical relations, which are beneficial for disambiguation. In this way, dependency structures are sufficiently expressive to be useful in natural language processing systems and at the same time sufficiently restricted to allow full parsing with high accuracy and efficiency. At least, this seems like a reasonable working hypothesis.

References

Abney, S. (1996). Partial parsing via finite-state cascades. *Journal of Natural Language Engineering* **2**: 337–344.

Alshawi, H. (1996). Head automata and bilingual tiling: Translation with minimal representations. *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 167–176.

Bangalore, S. (2003). Localizing dependencies and supertagging. In Bod, R., Scha, R. and Sima'an, K. (eds), *Data-Oriented Parsing*, CSLI Publications, University of Chicago Press, pp. 283–298.

Bar-Hillel, Y., Gaifman, C. and Shamir, E. (1960). On categorial and phrase-structure grammars. *Bulletin of the Research Council of Israel* **9F**: 1–16.

Barbero, C., Lesmo, L., Lombardo, V. and Merlo, P. (1998). Integration of syntactic and lexical information in a hierarchical dependency grammar. In Kahane, S. and Polguère, A. (eds), *Proceedings of the Workshop on Processing of Dependency-Based Grammars (ACL-COLING)*, pp. 58–67.

Bloomfield, L. (1933). *Language*. The University of Chicago Press.

Carroll, G. and Charniak, E. (1992). Two experiments on learning probabilistic dependency grammars from corpora, *Technical Report TR-92*, Department of Computer Science, Brown University.

Carroll, J. (2000). Statistical parsing. In Dale, R., Moisl, H. and Somers, H. (eds), *Handbook of Natural Language Processing*, Marcel Dekker, pp. 525–543.

Charniak, E. (2000). A maximum-entropy-inspired parser. *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 132–139.

Chomsky, N. (1970). Remarks on nominalization. In Jacobs, R. and Rosenbaum, P. S. (eds), *Readings in English Transformational Grammar*, Ginn and Co.

Clark, S. and Curran, J. R. (2004). Parsing the WSJ using CCG and log-linear models. *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 104–111.

Collins, M. (1997). Three generative, lexicalised models for statistical parsing. *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 16–23.

Collins, M. (1999). *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania.

Collins, M., Hajič, J., Brill, E., Ramshaw, L. and Tillmann, C. (1999). A statistical parser for Czech. *Proceedings of the 37th Meeting of the Association for Computational Linguistics (ACL)*, pp. 505–512.

Covington, M. A. (1984). *Syntactic Theory in the High Middle Ages*. Cambridge University Press.

Covington, M. A. (1990a). A dependency parser for variable-word-order languages, *Technical Report AI-1990-01*, University of Georgia.

Covington, M. A. (1990b). Parsing discontinuous constituents in dependency grammar. *Computational Linguistics* **16**: 234–236.

Covington, M. A. (1994). Discontinuous dependency parsing of free and fixed word order: Work in progress, *Technical Report AI-1994-02*, University of Georgia.

Covington, M. A. (2001). A fundamental algorithm for dependency parsing. *Proceedings of the 39th Annual ACM Southeast Conference*, pp. 95–102.

Curran, J. R. and Clark, S. (2004). The importance of supertagging for wide-coverage CCG parsing. *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*, pp. 282–288.

Debusmann, R. (2001). *A declarative grammar formalism for dependency grammar*, Master's thesis, Computational Linguistics, Universität des Saarlandes.

Debusmann, R., Duchier, D. and Kruijff, G.-J. M. (2004). Extensible dependency grammar: A new methodology. *Proceedings of the Workshop on Recent Advances in Dependency Grammar*, pp. 78–85.

Dowty, D. (1989). On the semantic content of the notion of 'thematic role'. In Chierchia, G., Partee, B. H. and Turner, R. (eds), *Properties, Types and Meaning. Volume II: Semantic Issues*, Reider, pp. 69–130.

Duchier, D. (1999). Axiomatizing dependency parsing using set constraints. *Proceedings of the Sixth Meeting on Mathematics of Language*, pp. 115–126.

Duchier, D. (2003). Configuration of labeled trees under lexicalized constraints and principles. *Research on Language and Computation* **1**: 307–336.

Duchier, D. and Debusmann, R. (2001). Topological dependency trees: A constraint-based account of linear precedence. *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 180–187.

Earley, J. (1970). En efficient context-free parsing algorithm. *Communications of the ACM* **13**: 94–102.

Eisner, J. M. (1996a). An empirical comparison of probability models for dependency grammar, *Technical Report IRCS-96-11*, Institute for Research in Cognitive Science, University of Pennsylvania.

Eisner, J. M. (1996b). Three new probabilistic models for dependency parsing: An exploration. *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, pp. 340–345.

Eisner, J. M. (2000). Bilexical grammars and their cubic-time parsing algorithms. In Bunt, H. and Nijholt, A. (eds), *Advances in Probabilistic and Other Parsing Technologies*, Kluwer, pp. 29–62.

Fillmore, C. J. (1968). The case for case. In Bach, E. W. and Harms, R. T. (eds), *Universals in Linguistic Theory*, Holt, Rinehart and Winston, pp. 1–88.

Foth, K., Daum, M. and Menzel, W. (2004). A broad-coverage parser for German based on defeasible constraints. *Proceedings of KONVENS 2004*, pp. 45–52.

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

Grimaldi, R. P. (2004). *Discrete and Combinatorial Mathematics*. 5th edn, Addison-Wesley.

Harper, M. P. and Helzerman, R. A. (1995). Extensions to constraint dependency parsing for spoken language processing. *Computer Speech and Language* **9**: 187–234.

Harper, M. P., Helzermann, R. A., Zoltowski, C. B., Yeo, B. L., Chan, Y., Steward, T. and Pellom, B. L. (1995). Implementation issues in the development of the PARSEC parser. *Software: Practice and Experience* **25**: 831–862.

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

Hellwig, P. (1980). PLAIN – a program system for dependency analysis and for simulating natural language inference. In Bolc, L. (ed.), *Representation and Processing of Natural Language*, Hanser, pp. 195–198.

Hellwig, P. (1986). Dependency unification grammar. *Proceedings of the 11th International Conference on Computational Linguistics (COLING)*, pp. 195–198.

Hellwig, P. (2003). Dependency unification grammar. In Agel, V., Eichinger, L. M., Eroms, H.-W., Hellwig, P., Heringer, H. J. and Lobin, H. (eds), *Dependency and Valency*, Walter de Gruyter, pp. 593–635.

Holan, T., Kuboň, V. and Plátek, M. (1997). A prototype of a grammar checker for Czech. *Fifth Conference on Applied Natural Language Processing*, pp. 147–154.

Hudson, R. A. (1984). *Word Grammar*. Blackwell.

Hudson, R. A. (1990). *English Word Grammar*. Blackwell.

Isozaki, H., Kazawa, H. and Hirao, T. (2004). A deterministic word dependency analyzer enhanced with preference learning. *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*, pp. 275–281.

Jackendoff, R. (1972). *Semantic Interpretation in Generative Grammar*. MIT Press.

Jackendoff, R. S. (1977). $\overline{X}$ *Syntax: A Study of Phrase Structure*. MIT Press.

Järvinen, T. and Tapanainen, P. (1998). Towards an implementable dependency grammar. In Kahane, S. and Polguère, A. (eds), *Proceedings of the Workshop on Processing of Dependency-Based Grammars*, pp. 1–10.

Joshi, A. and Sarkar, A. (2003). Tree adjoining grammars and their application to statistical parsing. In Bod, R., Scha, R. and Sima'an, K. (eds), *Data-Oriented Parsing*, CSLI Publications, University of Chicago Press, pp. 253–281.

Kahane, S., Nasr, A. and Rambow, O. (1998). Pseudo-projectivity: A polynomially parsable non-projective dependency grammar. *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics*, pp. 646–652.

Karlsson, F. (1990). Constraint grammar as a framework for parsing running text. In Karlgren, H. (ed.), *Papers presented to the 13th International Conference on Computational Linguistics (COLING)*, pp. 168–173.

Karlsson, F., Voutilainen, A., Heikkilä, J. and Anttila, A. (eds) (1995). *Constraint Grammar: A language-independent system for parsing unrestricted text*. Mouton de Gruyter.

Kasami, T. (1965). An efficient recognition and syntax algorithm for context-free languages, *Technical Report AF-CRL-65-758*, Air Force Cambridge Research Laboratory.

Kromann, M. T. (2004). Optimality parsing and local cost functions in Discontinuous Grammar. *Electronic Notes of Theoretical Computer Science* **53**: 163–179.

Kruijff, G.-J. M. (2001). *A Categorial-Modal Logical Architecture of Informativity: Dependency Grammar Logic and Information Structure*. PhD thesis, Charles University.

Kruijff, G.-J. M. (2002). Formal and computational aspects of dependency grammar: History and development of DG, *Technical report*, ESSLLI-2002.

Kudo, T. and Matsumoto, Y. (2000). Japanese dependency structure analysis based on support vector machines. *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC)*, pp. 18–25.

Kudo, T. and Matsumoto, Y. (2002). Japanese dependency analysis using cascaded chunking. *Proceedings of the Sixth Workshop on Computational Language Learning (CoNLL)*, pp. 63–69.

Lecerf, Y. (1960). Programme des conflits, modèle des conflits. *Bulletin bimestriel de l'ATALA* **1(4): 11–18, 1(5): 17–36**.

Lin, D. (1996). On the structural complexity of natural language sentences. *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, pp. 729–733.

Lombardo, V. and Lesmo, L. (1996). An Earley-type recognizer for Dependency Grammar. *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, pp. 723–728.

Marcus, M. P., Santorini, B. and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* **19**: 313–330.

Marcus, M. P., Santorini, B., Marcinkiewicz, M. A., MacIntyre, R., Bies, A., Ferguson, M., Katz, K. and Schasberger, B. (1994). The Penn Treebank: Annotating predicate-argument structure. *Proceedings of the ARPA Human Language Technology Workshop*, pp. 114–119.

Marcus, S. (1965). Sur la notion de projectivité. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik* **11**: 181–192.

Maruyama, H. (1990). Structural disambiguation with constraint propagation. *Proceedings of the 28th Meeting of the Association for Computational Linguistics (ACL)*, Pittsburgh, PA, pp. 31–38.

McDonald, R., Crammer, K. and Pereira, F. (2005). Online large-margin training of dependency parsers. *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 91–98.

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

Menzel, W. and Schröder, I. (1998). Decision procedures for dependency parsing using graded constraints. In Kahane, S. and Polguère, A. (eds), *Proceedings of the Workshop on Processing of Dependency-Based Grammars*, pp. 78–87.

Milward, D. (1994). Dynamic dependency grammar. *Linguistics and Philosophy* **17**: 561–605.

Misra, V. N. (1966). *The Descriptive Technique of Panini*. Mouton.

Nasr, A. and Rambow, O. (2004). A simple string-rewriting formalism for dependency grammar. *Proceedings of the Workshop on Recent Advances in Dependency Grammar*, pp. 25–32.

Nikula, H. (1986). *Dependensgrammatik*. Liber.

Nivre, J. (2003). An efficient algorithm for projective dependency parsing. In Van Noord, G. (ed.), *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pp. 149–160.

Nivre, J. (2005). *Inductive Dependency Parsing of Natural Language Text*. PhD thesis, Växjö University.

Nivre, J., Hall, J. and Nilsson, J. (2004). Memory-based dependency parsing. In Ng, H. T. and Riloff, E. (eds), *Proceedings of the 8th Conference on Computational Natural Language Learning (CoNLL)*, pp. 49–56.

Obrebski, T. (2003). Dependency parsing using dependency graph. In Van Noord, G. (ed.), *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pp. 217–218.

Oflazer, K. (2003). Dependency parsing with an extended finite-state approach. *Computational Linguistics* **29**: 515–544.

Robins, R. H. (1967). *A Short History of Linguistics*. Longman.

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

Roche, E. (1997). Parsing with finite state transducers. In Roche, E. and Schabes, Y. (eds), *Finite-State Language Processing*, MIT Press, pp. 241–281.

Samuelsson, C. (2000). A statistical theory of dependency syntax. *Proceedings of the 18th International Conference on Computational Linguistics (COLING)*.

Schröder, I. (2002). *Natural Language Parsing with Graded Constraints*. PhD thesis, Hamburg University.

Sgall, P., Hajičová, E. and Panevová, J. (1986). *The Meaning of the Sentence in Its Pragmatic Aspects*. Reidel.

Sleator, D. and Temperley, D. (1991). Parsing English with a link grammar, *Technical Report CMU-CS-91-196*, Carnegie Mellon University, Computer Science.

Sleator, D. and Temperley, D. (1993). Parsing English with a link grammar. *Third International Workshop on Parsing Technologies (IWPT)*, pp. 277–292.

Starosta, S. (1988). *The Case for Lexicase: An Outline of Lexicase Grammatical Theory*. Pinter Publishers.

Tapanainen, P. and Järvinen, T. (1997). A non-projective dependency parser. *Proceedings of the 5th Conference on Applied Natural Language Processing*, pp. 64–71.

Tesnière, L. (1959). *Éléments de syntaxe structurale*. Editions Klincksieck.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer.

Wang, W. and Harper, M. P. (2004). A statistical constraint dependency grammar (CDG) parser. In Keller, F., Clark, S., Crocker, M. and Steedman, M. (eds), *Proceedings of the Workshop in Incremental Parsing: Bringing Engineering and Cognition Together (ACL)*, pp. 42–29.

Yamada, H. and Matsumoto, Y. (2003). Statistical dependency analysis with support vector machines. In Van Noord, G. (ed.), *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pp. 195–206.

Yli-Jyrä, A. (2003). Multiplanarity – a model for dependency structures in treebanks. In Nivre, J. and Hinrichs, E. (eds), *Proceedings of the Second Workshop on Treebanks and Linguistic Theories (TLT)*, Växjö University Press, pp. 189–200.

Younger, D. H. (1967). Recognition and parsing of context-free languages in time n^3. *Information and Control* **10**: 189–208.

Zwicky, A. M. (1985). Heads. *Journal of Linguistics* **21**: 1–29.