Dependency grammar for computational linguists. Course notes for ESSLLI 2010 in Copenhagen

Matthias Buch-Kromann
Center for Research and Innovation in Translation and Translation Technology
Copenhagen Business School

Draft — do not quote without permission!

July 10, 2010

Abstract

The dependency grammar course consists of five parts, one for each day. On the first day of the course, we give a brief overview of dependency grammar as well as Discontinuous Grammar (DG), the dependency theory that will be presented in detail in the course. We also describe the notion of deep tree used in DG to account for the primary dependency structure, the interface to compositional semantics, and illustrate three different ways of analyzing connectives that result in the same functor-argument structure. On the second day, we describe how DG accounts for complex word order phenomena, such as topicalizations, extrapositions, scrambling, partial verb phrases, etc. On the third and fourth day, we describe how DG uses a mechanism for creating simple and complex fillers to account for secondary dependencies, constructions where a phrase seems to satisfy two or more dependency roles simultaneously. Finally, on the fifth day, we outline how the mechanisms required for syntax can be extended to account for linguistic structure in morphology, discourse, coreference, and punctuation. The course notes build on material from (Buch-Kromann 2006, 2009).

Contents

Part I

Introduction to dependency grammar and Discontinuous Grammar

In the first part of the course, we provide a brief introduction to dependency grammar and Discontinuous Grammar, followed by a detailed description of the deep trees used in DG to account for primary dependencies.

1 Introduction to dependency grammar

In this section, we define dependency graphs formally, present three different ways of visualizing them, and describe how dependency structure relates to phrase structure. We then provide a brief account of the history of dependency grammar, and define the scope of the course.

1.1 Dependency graphs

The unifying idea in dependency grammar is that all linguistic analyses can be represented as *dependency graphs*, ie, as directed graphs where the nodes represent lexical elements and the edges represent *dependency relations*, understood as directed, binary, bilexical relations. The dependency relations go from one lexical element (the *head*) to another lexical element (the *dependent*), and are usually assumed to be typed, ie, labelled by a *dependency label* that specifies the kind of relation that holds between the two elements. The inventory of dependency relations typically includes syntactic relations such as subject, direct object, verbal object, attributival adjective, and temporal adverbial, to name a few, but may also include other kinds of relations, such as morphological relations, discourse relations, and coreference relations. The lexical elements in a dependency analysis are often taken to be words within a single sentence, but may in a broader conception of dependency grammar include subword units (morphemes) and multi-word units (idioms). A dependency graph may also provide an analysis for an entire discourse or dialogue rather than an isolated sentence, an issue we will return to in Section 6.

1.2 Visualization formats and implicit phrase structure

Figure 1 is an example of a dependency analysis in arc-graph format of the sentence (1) taken from the Penn Treebank (Marcus et al. 1994). The analysis follows the annotation guidelines from the Copenhagen Dependency Treebanks (Buch-Kromann et al. 2009),[1] a set of parallel dependency treebanks for Danish, English, German, Italian, and Spanish.

(1) That cold, empty sky was full of fire and light.

In the *arc-graph format*, the words in the analysis are drawn on a horizontal line, with the associated word attributes drawn immediately below the words (as with the Penn Treebank word class attribute shown in Figure 1); the dependency relations are drawn as directed arcs that go from the head word to the dependent word, with the dependency label written either at the arrow tip, or at the middle of the arc in analyses where a word has more than one in-coming arrow. For example, the *subj* arc from *'was'* to *'That'* encodes that the phrase headed by *'That'* functions as the subject of the main verb *'was'*, and the *attr* arc from *'That'* to *'cold'* encodes that *'cold'* is an attributive adjective headed by the determiner

[1] The CDT treebanks and the CDT annotation guidelines can be downloaded from (Buch-Kromann et al. 2010).

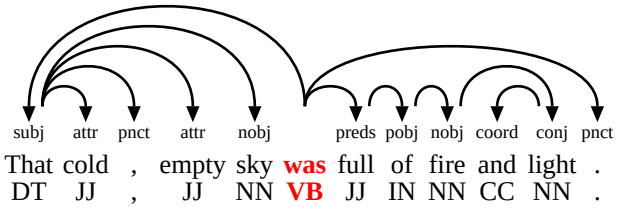

Figure 1: The Copenhagen Dependency Treebank analysis of (1) in arc-graph format.

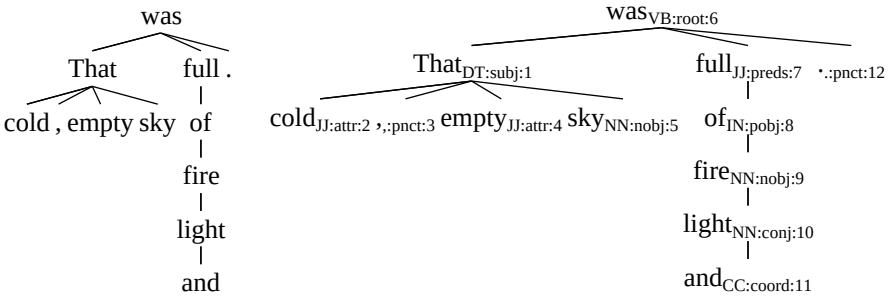

Figure 2: Classical dependency tree format (left) and enriched tree format (right) with word class, dependency label, and word order for the dependency analysis in Figure 1.

'That'. The most important dependency types for primary syntactic dependencies in the CDT treebanks are shown in Figure 6 on page 10.

The arc-graph format is not the only way of drawing dependency analyses. Other layouts include the *classical dependency tree format* introduced by Tesniere (1959), shown in Figure 2 (left), in which the lexical elements are drawn as nodes in a tree structure. From a formal point of view, the enriched form of the classical tree format shown in Figure 2 (right) encodes the same information as the arc-graph format in Figure 1. A variant of the enriched tree format is used in the Prague Dependency Treebank (Böhmová et al. 2001); many dependency frameworks prefer the arc-graph format because it provides a more appealing visualization for general graphs where lexical elements are allowed to have more than one incoming relation, and for large graphs that arise in the analysis of long sentences or entire discourses, because arc-graphs tend to be more flat and are easier to break up into lines and pages.

Dependency trees can also be drawn in an *X-bar phrase structure format* that resembles phrase-structure trees. In the X-bar phrase structure format, each word in the dependency tree is translated into three nodes: a terminal node that encodes the word (eg, *'full'*), a lexical node that encodes the word class (eg, JJ), and a phrasal node that encodes the phrasal projection of the word and the dependency role assigned to this phrase (eg, JJ"$_{\text{preds}}$), as shown in Figure 3. This translation gives a reversible mapping between classical dependency analyses in which all nodes represent lexical elements, and restricted phrase-structure analyses in which all phrasal nodes dominate a unique lexical node which in turn dominates a unique word that functions as the lexical head of the phrase. Comparing the representations in Figure 1, Figure 2 (right), and Figure 3, it should be obvious that they merely represent three different ways of encoding the same information, and that each format can be derived easily from the two other formats. From this perspective, a dependency tree is merely a phrase-structure tree with the additional restriction that each phrase has a unique lexical head.

We can illustrate the differences and similarities between dependency analyses and phrase structure analyses by comparing the dependency analysis in Figure 3 with the corresponding Penn Treebank analysis in Figure 4. There are many important similarities, but also some important differences. In particular: (i) the PTB analysis simplifies the tree structure by leaving out phrasal nodes for single-

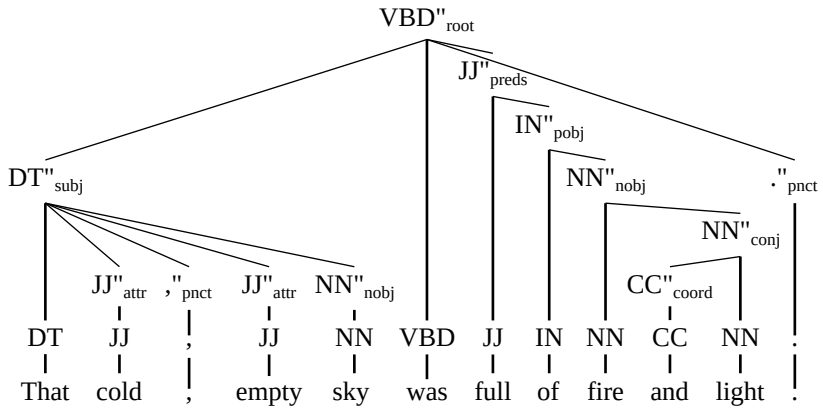

Figure 3: X-bar phrase-structure format for the dependency analysis in Figures 1 and 2.

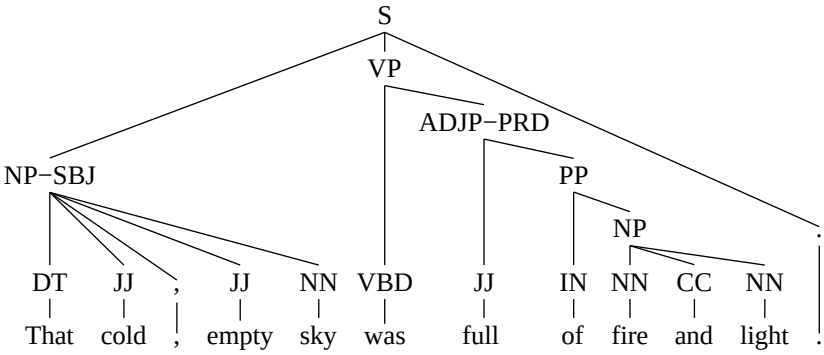

Figure 4: Penn Treebank phrase-structure analysis of sentence (1).

word phrases, but this is a minor difference; (ii) the PTB analysis does not label the phrasal nodes in a way that unambiguously specifies the lexical head associated with each phrase;[2] (iii) the PTB analysis sometimes includes phrasal nodes above the X-bar skeleton (in particular, the sentence node S, which has historically been analyzed as a junction of the subject noun phrase and the verb phrase as equals); and (iv) the PTB analysis does not systematically encode the syntactic role assigned to each phrase — in the example, the syntactic role is only specified for the subject and the predicative.

Although dependency analyses can be viewed as restricted phrase-structure analyses in many respects, there are a few areas where dependency analyses tend to be less restrictive. In particular, Context-Free Grammar and many other (but not all) versions of phrase-structure grammar require the branches in a phrase-structure tree to be projective (non-crossing), ie, the terminal yield of every phrase in the tree must be a sequence of adjacent words without any intervening gaps. In contrast, dependency-based theories often allow these non-projective, crossing branches, an issue we will return to in Section 3. Likewise, dependency frameworks often allow a multi-stratal view of dependency structure where a sentence either has more than one associated dependency graph,[3] or one dependency graph where the edges can be divided into several layers, often corresponding to trees, with nodes shared across layers. The result is that the combined dependency graph may be a general graph rather

[2]PTB is intended to be a theory-neutral treebank (if that is truly possible), and intentionally leaves many aspects of the annotation underspecified to avoid taking sides in the most hotly debated areas of syntactic analysis, such as the internal structure of coordinations and NPs. In practice, the head information is needed by lexicalized parsers and therefore has to be reconstructed using heuristic head-finding algorithms.

[3]For example, the Prague Dependency Treebank employs two trees: an analytical layer that seeks to describe the syntactic structure of the sentence, and a tectogrammatical layer that seeks to describe the semantic structure of the sentence.

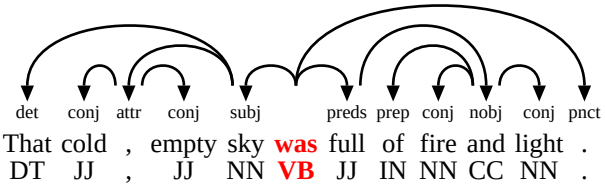

Figure 5: An alternative dependency analysis of (1), which differs from the analysis advocated by the Copenhagen Dependency Treebanks in important respects.

than a tree; in particular, the edge structure in the graph may be allowed to be cyclic and to have more than one incoming edge for each node.

Phrases are a well-defined notion in dependency grammars, although they are encoded implicitly rather than explicitly as in phrase structure grammar: every word heads a phrase consisting of all the words that can be reached from that word by following the arrows (that is, the set of words contained in the subtree associated with the phrasal projection of the word in the X-bar phrase-structure representation of the dependency analysis). In particular, the word *'That'* in Figure 1 heads the phrase *'That cold, empty sky'*, the word *'of'* heads the phrase *'of fire and light'*, the word *'light'* heads the phrase *'and light'*, and the word *'cold'* heads the phrase *'cold'*.

As in phrase-structure grammar, there is no general agreement on how to analyze even the most common linguistic constructions. As an illustration, Figure 5 shows a completely valid dependency analysis of (1) that departs from the CDT guidelines in important respects. Some of the possibilities for alternative analyses of sentence (1) include: analyzing the noun *'sky'* rather than the determiner *'That'* as the head of the noun phrase *'That cold, empty sky'* (NP vs. DP analysis); analyzing the noun *'sky'* rather than the determiner *'That'* as the head of the attributives *'cold'* and *'empty'*; analyzing the attributives *'cold'* and *'empty'* as one coordinated dependent rather than as two separate dependents; or assuming a different dependency structure in coordinations like *'fire and light'*, eg, analyzing *'and'* as the head of either the entire coordination *'fire and light'* or of the second conjunct *'and light'*.

We will return to some of these differences of analysis later, but for now, it suffices to note that although these differences do have important linguistic ramifications, they all fall within a shared abstract dependency framework. Moreover, the same range of differences in analysis can be found in phrase-structure grammars, especially those incorporating some version of X-bar theory (Chomsky 1970), such as Head-driven Phrase Structure Grammar (Pollard and Sag 1994), Lexical-Functional Grammar (Dalrymple et al. 1994), Government and Binding (Chomsky 1982), and Lexicalized Tree-Adjoining Grammar (Joshi and Schabes 1997; Abeillé and Rambow 2000; Joshi and Rambow 2003).

1.3 A brief history of dependency grammar

Dependency grammar has a long history in descriptive linguistics. The earliest forms of dependency grammar can be traced back to the Indian grammarian Pānini 2600 years ago, the Greek grammarians Thrax and Appolonius 2000 years ago, the Latin grammarian Priscian 1500 years ago, the Arabic grammarian Ibn al-Sarrāǧ 1200 years ago, and the medieval grammarians Thomas of Erfurt and Martin of Dacia 700 years ago. Dependency grammar was the dominant paradigm in descriptive linguistics until modern times, where Jespersen (1984/1937) made one of the first attempts at defining a formal notion of dependency grammar, and Tesniere (1959) provided the first modern formalization of dependency grammar.[4]

[4]For a detailed account of the history of dependency grammar, see Kruijff (2006).

Dependency grammar lost ground as the dominant paradigm in theoretical linguistics with the advent of generative phrase-structure grammars introduced by Bloomfield (1933) and formalized by Chomsky (1957). However, important work on dependency grammar continued in the European tradition of descriptive linguistics, especially in the work of the Praguian Functional-Generative Description (Sgall et al. 1986), the Russian Meaning-Text Theory (Melcuk 1988), the German school of valency grammar (Helbig and Schenkel 1971/1969), and the dependency frameworks Lexicase (Starosta 1988) and Word Grammar (Hudson 1990, 2010).

In the 1980s and 1990s, phrase structure grammar and dependency grammar began to converge again. On the phrase structure side, the prominence of lexical heads was reintroduced with the advent of lexicalized phrase structure grammars such as Head-Driven Phrase Structure Grammar (Pollard and Sag 1994), Lexical-Functional Grammar (Dalrymple et al. 1994), Lexicalized Tree Adjoining Grammar (Joshi and Schabes 1997; Joshi and Rambow 2003), and Combinatory Categorial Grammar (Steedman 2000). This development was supported by the realization in computational linguistics that lexicalization was the key to good parsing performance in context-free parsing based on treebanks (cf. Eisner 1996; Collins 1997).

On the dependency side, ideas about constraint-based approaches and unification from phrase-structure grammar were used to formulate new, more formally rigorous dependency-based theories of grammar, including Link Grammar (Sleator and Temperley 1993), Dependency Unification Grammar (Hellwig 2003), Constraint Dependency Grammar (Menzel and Schröder 1998) and Weighted Constraint Dependency Grammar (Schröder 2002), Functional Dependency Grammar (Järvinen and Tapanainen 1998), Discontinuous Grammar (Kromann 1999a; Buch-Kromann 2006)Dependency Grammar Logic (Kruijff 2001), and Extensible Dependency Grammar (Debusmann et al. 2004). Mirroring the developments in phrase structure grammar, where the emergence of the Penn Treebank (Marcus et al. 1994) had spurred the development of corpus-based approaches to natural language processing, the dependency-based paradigm was applied in the analysis of corpora to produce a number of dependency-based treebanks, most prominently the Prague Dependency Treebank (Böhmová et al. 2001), but also treebanks for Turkish (Oflazer et al. 2003), Danish (Kromann 2003), Basque (Aduriz et al. 2003), Italian (Bosco and Lombardo 2004), and Japanese (Kurohashi and Nagao 1998), to mention a few. For a more detailed overview of dependency grammar and dependency parsing, see Nivre (2005).

1.4 The scope of the course

While there are important similarities between the many dependency-based frameworks, there are also many important differences. However, rather than trying to cover their similarities and differences in detail, this course seeks to give an in-depth introduction to one particular dependency framework, Discontinuous Grammar (DG). DG couples a detailed, large-coverage linguistic theory with large-scale treebank annotations for five languages in the Copenhagen Dependency Treebanks. In the course, we will describe how DG deals with a wide range of linguistic phenomena, including primary dependencies, discontinuous word orders, secondary dependencies, and how these notions can be extended to morphology and discourse.

2 Introduction to Discontinuous Grammar

In the previous section, we introduced dependency graphs and their relationship to phrase-structure graphs; we also presented a brief history of dependency grammar and mentioned some of the frameworks that have been proposed within dependency grammar. For the rest of the course, we will focus on one particular dependency formalism, Discontinuous Grammar (DG). We start by providing a brief

introduction to DG, followed by a detailed description of the deep trees that DG uses to account for primary dependencies and to provide an interface to functor-argument structure and compositional semantics. To illustrate the difference between these two levels of analysis, we present three analyses of connectives that are markedly different with respect to their syntax, although they are very similar in terms of their functor-argument structures.

2.1 Overview of Discontinuous Grammar

Discontinuous Grammar (Kromann 1999a, 2001; Buch-Kromann 2006, 2009) is a dependency theory that seeks to give a formal dependency-based account of linguistic structure, with the same level of detail as in phrase-structure based theories like Head-driven Phrase Structure Grammar (Pollard and Sag 1994), Lexical-Functional Grammar (Dalrymple et al. 1994), Government and Binding (Chomsky 1982), and Lexicalized Tree-Adjoining Grammar (Joshi and Schabes 1997; Abeillé and Rambow 2000; Joshi and Rambow 2003). The DG framework includes ideas about machine learning, generative probability models, and computational models of human parsing as well, but in this paper, we will focus on the linguistic aspects of DG.

Like phrase-structure based theories, DG seeks to account for a wide range of linguistic phenomena: how the syntactic structure determines compositional semantics as well as word order, how to account for secondary dependencies where phrases fill more than one syntactic role (known as filler-gap constructions in HPSG), and how to account for complex syntactic phenomena such as elliptic coordinations, parasitic gaps, speech repairs, etc. In its account of these phenomena, DG employs a variety of mechanisms, many of them based on ideas borrowed from other frameworks, including: an unordered deep tree that provides the interface to compositional semantics; a projective surface tree that controls word order; a mechanism for generating phonetically empty copies of phrases (*fillers*) to account for secondary dependencies in a wide range of syntactic constructions (verbal chains, sharing, raising, relatives, parasitic gaps, elliptic coordinations, speech repairs); and an anaphor mechanism that accounts for anaphoric elements that require an antecedent. This formal DG machinery, supplemented with a few mechanisms for handling punctuation marks and lexical transformations, can be used to provide a unified DG account of syntax, discourse, anaphora, and morphology. The current paper seeks to provide a first introduction to this formal machinery.

DG has formed the theoretical basis for the annotations in the Copenhagen Dependency Treebanks (Buch-Kromann et al. 2009, 2010), an ongoing annotation project that seeks to construct a set of parallel dependency treebanks for Danish, English, German, Italian, and Spanish. Each CDT treebank consists of a text corpus of 40-100,000 words annotated with respect to syntax, morphology, discourse, coreference, and translational equivalence.[5] The corpora consist of a Danish source corpus based on a balanced, mixed-genre written corpus (Keson and Norling-Christensen 1998), which has been translated into the other languages by professional translators with native target-language competence. The treebanks provide an important empirical basis for the DG theory, and the examples in this paper are annotated according to the annotation guidelines provided by this framework.

2.2 Primary dependencies: complements and adjuncts

Most syntactic theories assume that humans divide text and speech into segments that represent lexical elements (*lexemes*), and that these lexemes are subsequently grouped into larger units, called *phrases*.

[5]The treebanks and the annotation manual, which provides a detailed description of the relation inventory as well as the current annotation status and statistics related to inter-annotator agreement, can be downloaded from the CDT repository Buch-Kromann et al. (2010).

Complement relations – top 15	Adjunct relations – top 15
nobj: nominal object: *for the*$^{1}_{nobj}$ *child*$^{2}_{nobj}$	**pnct**: punctuation: *It*$^{2}_{subj}$ *is* !$^{2}_{pnct}$
subj: subject: *They*$^{2}_{subj}$ *saw me*$^{2}_{dobj}$	**attr**: attributive: *the white*$^{1}_{attr}$ *cat*$^{1}_{nobj}$
vobj: verbal object: *He*$^{2}_{subj}$ *had left*$^{2}_{vobj}$ *it*$^{3}_{dobj}$	**conj**: conjunct: *John and*$^{3}_{coord}$ *Mary*$^{1}_{conj}$
dobj: direct object: *He*$^{2}_{subj}$ *left us*$^{2}_{dobj}$	**coord**: coordinator: *tea or*$^{3}_{coord}$ *coffee*$^{1}_{conj}$
pobj: prepositional object: *one of*$^{1}_{pobj}$ *them*$^{2}_{nobj}$	**time**: time adverbial: *We*$^{2}_{subj}$ *leave now*$^{2}_{time}$
preds: subject predicative: *It*$^{2}_{subj}$ *was blue*$^{2}_{preds}$	**loc**: location adverbial: *I*$^{2}_{subj}$ *fell here*$^{2}_{loc}$
predo: object pred.: *We*$^{2}_{subj}$ *found it*$^{2}_{dobj}$ *hard*$^{2}_{predo}$	**namef**: first name: *Igor*$^{2}_{namef}$ *Stravinsky*
possd: possessed: *His hat*$^{1}_{possd}$	**man**: manner adverbial: *I*$^{2}_{subj}$ *read slowly*$^{2}_{man}$
possr: possessor: *Peter*$^{2}_{possr}$ *'s hat*$^{2}_{possd}$	**degr**: degree adverbial: *very*$^{2}_{degr}$ *hard*
qobj: quotational obj.: *He*$^{2}_{subj}$ *said* “$^{4}_{pnct}$ *No*$^{2}_{qobj}$ ”$^{4}_{pnct}$	**neg**: negation: *I*$^{2}_{subj}$ *will not*$^{2}_{neg}$ *leave*$^{2}_{vobj}$
expl: expletive: *There*$^{2}_{expl}$ *arises one*$^{2}_{dobj}$ *question*$^{3}_{nobj}$	**relr**: restrictive relative: *the cat*$^{1}_{nobj}$ *that*$^{4}_{subj}$ *died*$^{1}_{relr}$
iobj: indirect object: *We*$^{2}_{subj}$ *gave him*$^{2}_{iobj}$ *flowers*$^{2}_{dobj}$	**relp**: paren. relative: *the cat*$^{1}_{nobj}$ (,$^{1}_{pnct}$ *which*$^{5}_{subj}$ *died*$^{1}_{relp}$
avobj: adverbial object: *as before*$^{1}_{avobj}$	**quant**: degree adverbial: *only*$^{2}_{quant}$ *two cats*$^{2}_{nobj}$
part: verbal particle: *break up*$^{1}_{part}$	**appr**: restrictive apposition: *the genius*$^{1}_{nobj}$ *Einstein*$^{1}_{appr}$
aobj: adjectival object: *he*$^{2}_{subj}$ *lied as*$^{2}_{add}$ *well*$^{3}_{aobj}$	**appa**: paren. appos.: *Einstein* (,$^{1}_{pnct}$ *the*$^{1}_{appa}$ *genius*$^{3}_{nobj}$

Figure 6: The fifteen most important syntactic complement and adjunct relations in the CDT treebanks. In the associated examples, the relation label and the head position is shown in sub- and superscripts after each non-root word.

In dependency-based theories, each phrase is assumed to consist of a *phrasal head* (or *governor*) lexeme that determines the syntactic and semantic type of the phrase, and a set of subphrases whose phrasal heads are called the *dependents* of the governor. The phrasal relationship between a governor and a dependent is called a *dependency*, and these dependencies form a tree structure which is called a *dependency tree* (or a *deep tree*). There are many linguistic tests which can be used to identify the head of a phrase. Eg, it is often possible to eliminate some or all of the dependents in the phrase, insert pronouns in their place, or manipulate the surrounding context, but the tests are sometimes contradictory, and there is no universal agreement in all cases about the best way of identifying phrases and phrasal heads.

As in most other linguistic theories, DG assumes that dependency relations can be subdivided into *complement relations* and *adjunct relations*. Complement relations are assumed to be lexically licensed by the governor, whereas adjunct relations are lexically licensed by the adjunct, ie, the adjunct may combine freely with any governor that matches its lexical requirements. One important consequence of this is that a governor always has a limited number of complements, but may in principle have an unlimited number of adjuncts. Most theories assume that complements are mostly obligatory, and that adjuncts are mostly optional — some theories, including DG, operate with optional complements and obligatory adjuncts. Prototypical complements include subjects, direct objects, indirect objects, and verbal objects, whose presence or absence is strongly dependent on the verb, whereas prototypical adjuncts include time and space adverbials, which may appear with almost any verb. The fifteen most important syntactic complement and adjunct roles in the Copenhagen Dependency Treebanks are shown in Figure 6.

The main function of a dependency tree is to provide an interface to semantics. Most formal semantic theories assume that phrases are assigned meanings according to the principle of compositionality, which states that the meaning of a phrase is computed as a function of the meanings of its parts (possibly supplemented with some kind of representation of the context). We will follow Dowty (1992) and the approach taken in many linguistic theories, including HSPG, by assuming that complements are lexically selected by their governor and function as *semantic arguments* to their governor in the compositional semantics, whereas adjuncts lexically select their governor and function as *modifiers* to

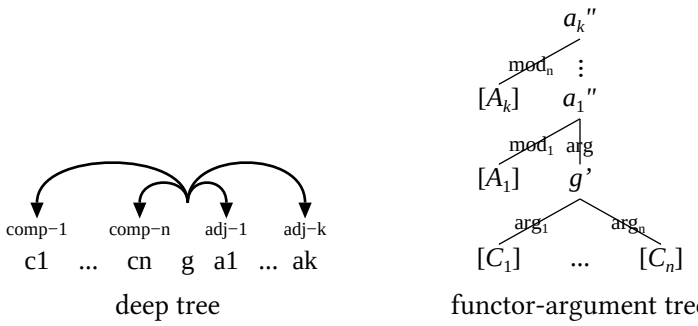

Figure 7: An unordered deep tree G (left) and its associated functor-argument tree $[\![G]\!]$ (right), consisting of a functor g' for governor g, complement trees $[\![C_1]\!], \ldots, [\![C_n]\!]$, and modifiers $a_1'', \ldots, a_k''$ with adjunct trees $[\![A_1]\!], \ldots, [\![A_k]\!]$.

their governor in the compositional semantics. More formally, we will make the following assumption:

Assumption 2.1 (Principle of compositionality) Let G be a phrase with governor g, complements $c_1, \ldots, c_n$, and adjuncts $a_1, \ldots, a_k$ ordered by a *modifier scope* (ie, an ordering from lowest-scoped to highest-scoped adjunct). Then the meaning $[\![G]\!]$ associated with G can be computed by:

$$[\![G]\!] = a_1''([\![A_1]\!]) \circ \cdots \circ a_k''([\![A_k]\!]) \circ g'([\![C_1]\!], \ldots, [\![C_n]\!])$$

where $\circ$ denotes function composition, g' is a functor-argument composition function given by the lexical entry for g (for the given complement frame), and each a_j'' is a modifier composition function given by the lexical entry for the adjunct a_j (for the given adjunct role).

The computation of the meaning of a phrase can be visualized by means of a *functor-argument tree*, as shown in Figure 7. Each node in the tree corresponds to the application of a functor or modifier supplied by a governor or an adjunct. That is, reading the tree bottom-up, g first provides a functor g' which computes a meaning for g and its complement phrases; this meaning is then modified by the lowest-scoped adjunct which integrates its own meaning into the meaning computed by g; the resulting meaning functions as input to the next higher-scoped adjunct, and the process continues until all adjuncts have computed a new meaning from the meaning returned by the previous adjunct, eventually resulting in a meaning for the entire phrase G. Deep trees can be viewed as underspecified functor-argument trees, since the deep tree does not encode the functional order in which the modifiers are applied, the modifier scope; but given a modifier scope, the deep tree determines the functor-argument tree uniquely.

2.3 Examples of dependency analyses

There is a wide range of linguistic tests which can be taken into account when trying to determine whether a particular dependent is best analyzed as a complement or an adjunct (cf. Vater 1978; Helbig 1992; Borsley 1996; Pustejovsky 1995). The tests that have been proposed include checking whether the dependent is obligatory or optional; whether the dependency is restricted to narrowly defined classes of heads or can be used with very few restrictions (in particular, if a dependent can be inserted in a phrase which is known to have no empty complement slots (eg, "do so"), without any change in meaning, it is an adjunct); looking at whether the dependent expresses an entity that is believed to be part of the argument structure of the head or not; and whether the dependent can be rephrased as an independent clause or sentence. However, none of these criteria work in all cases, and there are border-line cases where it is difficult to determine whether a particular dependency is best analyzed

as a complement relation or an adjunct relation (cf. Helbig 1992) — prepositional phrases often fall in this category. In DG, the presence of these border-line cases is not viewed as an inherent problem with the complement-adjunct distinction, but rather as a consequence of a genuinely free choice in the complement-adjunct mechanisms language users employ to decode utterances in their language and encode their possibly redundant perception of the grammar (cf. Buch-Kromann 2006; Kromann 1999b).[6] To illustrate how dependency analysis is performed in practice, we will look at three syntactic constructions: noun phrases with determiners, common nouns, and adjectivals; person names; and connectives analyzed as either combiners, conjunctions, or markers.

Noun phrases

When making a dependency analysis of a standard noun phrase consisting of a determiner, one or more adjectival attributives, and a common noun — like the noun phrase *'the red apple'* in the sentence *'the red apple is delicious'* — we first need to determine the head of the construction. The adjective is obviously a poor candidate since it has the wrong word class and can be omitted in many contexts, as shown in (2).

(2) The apple is delicious.

The determiner and the common noun are therefore the only obvious candidates for head. We continue our analysis by trying to delete one of the other words from the NP:

(3) ?The red is delicious.

(4) ?Red apple is delicious.

Without a supporting context, both deletions are slightly problematic, as indicated with the '?' notation: (3) sounds incomplete without a context, and in (4) the NP changes from a countable reading (one apple) to a generic mass reading (red apples as a generic substance). However, in richer contexts, both deletions are possible:

(5) Red apples are delicious.

(6) The green apple tastes bad, but (the red/*red apple) is delicious.

(7) The yellow lemon and red apple were delicious.

(8) We served two green and three red apples.

(9) The green apple and the red were delicious.

(10) The largest of the apples are particulary delicious.

When the determiner is deleted in (5) and (6), the NP assumes a non-countable, generic reading, which is ungrammatical in the context of (6) here, alternatives have been listed in parentheses, '*' is used to indicate ungrammatical expressions. Coordinations seem to be a special case, where both the determiner

[6]When comprehending a particular sentence at a particular instance in time, an individual language user obviously has to make a choice between a complement and an adjunct reading for each word, but there is nothing in the theory that forces us to assume that he or she will make that choice consistently over time, or that other language users will always make the same choice. On the contrary, it is conceivable that the language user has a redundant grammar that allows some constructions to be analyzed with both complement and adjunct mechanisms, and that language users differ with respect to their disambiguation preferences and degrees of redundancy for particular constructions. For some constructions, one mechanism may be much more economical than the other in terms of processing time and memory requirements, so that one of the mechanisms will tend to be used by all language users at all times; for other constructions, both mechanisms are almost equally economical and perhaps simultaneously encoded in the language user's grammar, so that a much greater variation is to be expected between individuals and over time. This hypothesized redundancy may well play an important role in language change.

and the common noun can be deleted in one of the conjuncts, as is obvious from (7)–(9), which is why we will later argue that special mechanisms are needed to account for elliptic coordinations, regardless of whether one chooses the determiner or the common noun as the head. Finally, (5) and the partitive construction in (10) clearly show that it is possible to construct noun phrases that unambiguously have a common noun and a determiner, respectively, as their head — unless, that is, we assume the presence of an elided head in these constructions, a rather awkward assumption that most dependency theories try to avoid.[7] So whatever our analysis, we will have to assume that both determiners and common nouns may head an NP: the only freedom lies in which one of them we choose as the head when they are both present. In languages with case, it is also possible to look at case and number agreement to identify the head, but the evidence is usually mixed or compatible with both analyses. For example, the number agreement in English exemplified by (11) shows that the common noun and the verb agree in number, without any number-marking on the determiner, which could be used as an argument in favor of an NP analysis; however, proponents of a DP analysis may counter that number agreement is mainly a semantic phenomenon, and that there are other languages (eg, German) where the NP must agree in case with the verb, and where it is the determiner rather than the common noun which receives the case marking.

(11) The friend is here. / The friends are here.

Compositional semantics is sometimes used to argue for one analysis over the other, but since we are very far from having a complete, formal theory of semantics, the validity of these arguments is far from certain; for example, in the analysis of generalized quantifiers in Montague semantics (Montague 1974; Gamut 1991), determiners take the common noun as their semantic argument, but analyses in which the common noun functions as the head are possible as well. In general, the syntactic evidence in many languages tends to lend most support to a DP analysis, whereas the semantic evidence tends to lend more support to an NP analysis. The choice of DP vs. NP analysis within a particular dependency framework therefore tends to be determined by whether the framework places most emphasis on syntax or semantics. In this paper, and in the Copenhagen Dependency Treebanks, we adopt a syntax-centered analysis proposed by Hudson (2004), which argues that determiners are pronouns that subcategorize for a noun, ie, the determiner acts as the head of the construction.

There is also a free choice in how we analyze attributive adjectives. As is evident from (5) and (6), whatever our choice of head in the NP, we can find examples where an attributive adjective unambiguously modifies a determiner and common noun, respectively (in the absence of elided heads). The only question is therefore what happens in examples where there is both a determiner and a common noun. As with the NP vs. DP discussion, the evidence is mixed. For example, the strong/weak agreement in German and Scandinavian exemplified by the German example (12) shows that the adjective agrees with the determiner, which seems to support the determiner as the head.

(12) ein guter Freund / der gute Freund
a good friend / the good friend

However, most examples are compatible with both analyses, especially if we assume that agreement between two words is not necessarily mediated by a direct dependency, but may be slightly more non-

[7] From a computational point of view, one of the advantages of dependency-based parsing is that we know the nodes in the graph before-hand so that we only need to identify the edges between them; with elided heads, this advantage disappears, and we suddenly have to decide whether elided heads are present or not, how many elided heads there are, when they are licensed, how they interact, etc. — a complication in the linguistic theory as well. As we will argue later for gapping coordinations, it is not possible to eliminate elided heads completely, but there is a big difference between allowing them in narrowly specified contexts and allowing them across the board.

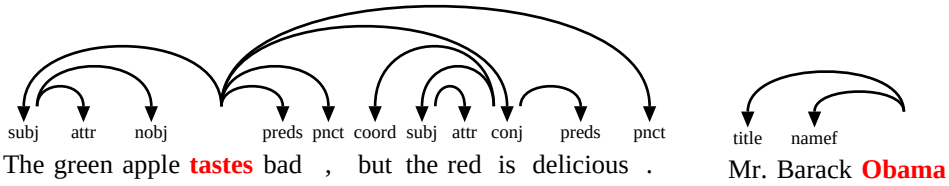

Figure 8: Dependency analysis of (6) on the left and (13) on the right.

	Combiner analysis	**Conjunction analysis**	**Marker analysis**
Syntactic head	C	X	X
Semantic head	C	C	Y
Syntax	comp comp X **C** Y	adj comp **X** C Y	adj adj **X** C Y
Semantics	$c'([\![X]\!], [\![Y]\!])$	$c''(c'([\![Y]\!]), [\![X]\!])$	$y''_C([\![Y]\!], [\![X]\!])$
CDT example	X did Y[8]	X because Y	X and Y

Table 1: Three alternative dependency analyses of connectives.

local. In the CDT analysis, we assume that attributival adjectives modify the determiner if present, and the common noun otherwise. The resulting analysis of (6) is shown in Figure 8.

Person names

The same principles can be applied to constructions that are often assumed to lack an internal dependency structure, such as person names like *'Mr. Barack Obama'*. Again, by deleting words and changing the context we can tease out the dependency structure, as shown in (13)–(15):

(13) Barack / Obama / Barack Obama / Mr. Obama / *Mr. Barack / Mr. Barack Obama arrived.

(14) President Obama / *President Barack arrived.

(15) Sir Paul McCartney / *Sir McCartney / Sir Paul / Paul / McCartney played at the charity concert.

The evidence seems to support an analysis of English proper names where the last name functions as the head, and the first name functions as a dependent (adjunct) of the last name, as shown in Figure 8 (right). Titles like *'Mr.'* and *'President'* normally attach to the last name, whereas honorifics like *'Sir'* attach to the first name.

Connectives as combiners, conjunctions, or markers

Connectives are used to link two phrases X and Y in a construction of the form $X\ C\ Y$, where C is a connective that typically represents a single lexical entity (such as *'and'*, *'because'*, etc.). Connectives are particularly frequent in syntax and discourse, and is the main focus of attention in the Penn Discourse Treebank. It is therefore worth taking a look at the three dependency analyses that naturally suggest themselves for connectives, and discuss their characteristics. The three analyses are summarized in Table 1, which also shows an example of a construction for which this analysis is used in the CDT treebanks.

In the *combiner analysis*, C takes X and Y as its complements, ie, C functions as the syntactic head; semantically, C provides a functor c' which computes a joint meaning from $[\![X]\!]$ and $[\![Y]\!]$, ie, C is the lexico-semantic head. In the *subordinating conjunction analysis*, C takes Y as its complement and functions as an adjunct of X, ie, X is the syntactic head; semantically, C computes a joint meaning

$c'([\![Y]\!])$ from Y, and integrates this meaning into $[\![X]\!]$ using an adjunct composition function c'' given by c, ie, c is the lexico-semantic head. Finally, in the *marker analysis*, C modifies Y which in turn modifies X, ie, X is the syntactic head; semantically, y computes $[\![Y]\!]$, selects an adjunct composition function y''_C using C as a semantically vacuous marker that merely disambiguates the choice of adjunct composition function, and uses y''_C to compute the meaning for the entire phrase from $[\![Y]\!]$ and $[\![X]\!]$, ie, y is the lexico-semantic head.

The three analyses are essentially identical in terms of their semantics (in all three cases, $[\![X]\!]$ and $[\![Y]\!]$ function as the arguments for a composition function), but markedly different in terms of their dependency structure and their syntactic and semantic headedness. That is, we have to distinguish carefully between what functions as the syntactic head (ie, what is the head word in the phrase that determines the word class associated with the phrase and functions as the receiver of incoming dependencies), what functions as the semantic head (ie, who is responsible for the semantic computation of the meaning associated with the phrase as a whole), and the computed meaning (which may well end up being the same in all three analyses). Much confusion about dependency grammar probably arises from the fact that the term 'dependency structure' is used to refer to the syntactic structure in some frameworks, and the semantic structure in others. In Discontinuous Grammar, the syntactic structure is always referred to as dependency structure, and the semantic structure is always referred to as functor-argument structure.

What determines which of the three analyses one should choose? The combiner analysis is most appropriate when the connective is the best candidate for the syntactic head of the phrase, ie, when it is the most central word in the phrase and its word class is most representative for the phrase as a whole. The marker analysis is most appropriate when the connective is optional, since this indicates that Y can handle the semantic composition on its own and therefore functions as the lexico-semantic head some of the time; it is therefore natural to assume that it functions as the semantic head all of the time. Finally, the conjunction analysis is most appropriate when C is obligatory and X is the natural syntactic head of the phrase. It is interesting that many theories of discourse structure, including Penn Discourse Treebank (Miltsakaki et al. 2004) and the dependency-based discourse analysis proposed by (Mladová et al. 2008), seem to analyze connectives as combiners — even in cases where CY is an adverbial clause modifying X, where virtually all mainstream theories of syntax opt for one of the two other analyses, or cases where the connectives are optional (the Penn Discourse Treebank refers to these cases as 'implicit connectives'). In these cases, one of the two other analyses would be far more natural.

The discussion of NP structure, person names, and connectives we have presented here is obviously rather superficial. However, it does give a general idea of how linguistic analyses are performed in a dependency framework, and illustrates that the arguments used to analyze a particular construction are not very different from the arguments that are employed in linguistic theories based on phrase-structure.

2.4 Dependency grammars: a toy example

Figure 9 shows a small lexicon with word classes, complement frames, and adjunct frames for the words in the sentence "John saw a little boy with a telescope today". The word class tags used in the rules are the two first letters of the PAROLE-based word class tags listed in Figure 10 and documented in detail by Keson and Norling-Christensen (1998). The lexicon describes the word class associated with each word, and the kinds of complement and adjunct dependencies that it licenses. In particular, it

[8] An auxiliary like *'did'* is not really a connective, but the CDT treebanks do not contain examples of connectives with the combiner analysis, so auxiliaries are the closest example.

Type	Word class	Complement frame	Adjunct frame
a	PI	$\xrightarrow{\text{nobj}}$: NC	
boy	NC		
John	NP		
little	AN		$\xleftarrow{\text{attr}}$: N
saw	VA	$\xrightarrow{\text{subj}}$: N, $\xrightarrow{\text{dobj}}$: N	
telescope	NC		
today	RG		$\xleftarrow{\text{time}}$: V|N
with	SP	$\xrightarrow{\text{nobj}}$: N	$\xleftarrow{\text{inst}}$: N|V, $\xleftarrow{\text{accom}}$: N|V

Figure 9: A small lexicon with supertypes, complements, and adjuncts.

A: adjective	**P**: pronoun	**V**: verb
AN: normal	**PP**: personal	**VA**: main
AC: cardinal	**PD**: demonstrative	**VE**: medial
AO: ordinal	**PI**: indefinite	**X**: extra-linguistic unit
C: conjunction	**PT**: interrog./relative	**XA**: abbreviation
CC: coordinating	**PC**: reciprocal	**XF**: foreign word
CS: subordinating	**PO**: possessive	**XP**: punctuation
I: interjection	**RG**: adverb	**XR**: formulae
N: noun	**SP**: preposition	**XS**: symbol
NP: proper	**U**: unique	**XX**: other
NC: common		

Figure 10: The main PAROLE word class tags for Danish.

specifies that the word "a" is an indefinite pronoun (PI), which takes any common noun (NC) as its nominal object; that the words "boy" and "telescope" are common nouns, and "John" a proper noun (NP); that "little" is a normal adjective (AN) that can modify any noun as an adjectival attributive (an adjunct); that "saw" is a verb (VA) that takes two nouns as its subject and direct object; that "today" is an adverb (RG) that can modify any noun or verb as a time adverbial (an adjunct); and that "with" is a preposition (SP) that can take any noun as its nominal object, and which can modify any noun or verb as an instrument adverbial (*inst*) or companionship adverbial (*accom*).

In other words, in this simple rule-based notion of a lexicon, the lexicon embodies a specification of what counts as a well-formed dependency graph. Other conceptions of dependency-based lexicons are possible as well. In a generative probabilistic dependency paradigm, the lexicon is responsible for computing the probabilities associated with each step in the generative process associated with a particular dependency analysis; and in a discriminative probabilistic paradigm, the lexicon is merely responsible for computing the features that are fed into some kind of machine learning algorithm. In these other paradigms, the lexicon does not necessarily reject ungrammatical analyses — on the contrary, having a lexicon that overgenerates is an advantage, as long as ungrammatical analyses are assigned a lower probability or score than the alternative analysis we want the parser to come up with.

The lexicon in Figure 9 allows many analyses of the same sentence. Two well-formed analyses of the sentence "John saw a little boy with a telescope today" are shown in Figure 11 and 12. Figure 11 is a sensible linguistic analysis, whereas Figure 12 is a very poor analysis — in canonical word order, the analysis in Figure 12 corresponds to the sentence "A boy today with a telescope saw little John." Many of the dependencies in Figure 12 are discontinuous, ie, they cross other arcs or span over the root node.

More formally, a primary dependency is called *continuous* (*projective*) if all nodes between the head and the dependent in the linear order are dominated by the head in the deep tree; the edge is

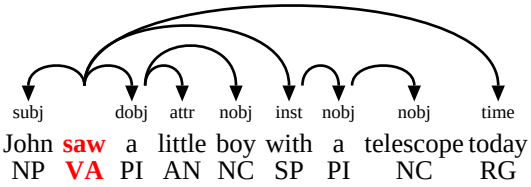

Figure 11: A continuous analysis licensed by the lexicon in Figure 9.

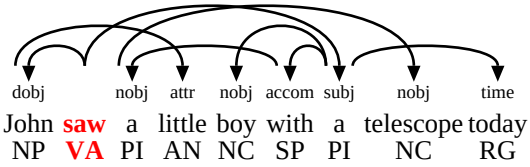

Figure 12: A highly discontinuous analysis licensed by the lexicon in Figure 9, illustrating the need for controlling word order.

called *discontinuous* (*non-projective*) otherwise. A primary dependency tree is said to be *projective* if all the primary dependencies in the tree are projective, and *non-projective* if it contains at least one non-projective edge. In Figure 11, all the dependencies are projective; for example, the *inst* dependency from *'saw'* to *'with'* is projective because *'saw'* dominates all the intermediate words *'a little boy'*, and the *subj* dependency is trivially projective because there are no words between *'John'* and *'saw'*. In Figure 12, there are many non-projective dependencies; for example, the *time* dependency from the second *'a'* to *'today'* is non-projective because *'a'* fails to dominate *'telescope'*. In this particular case, the correct analysis happens to be projective. But as we will see in the next section, there are many examples of well-formed sentences that involve non-projective dependencies. Since complement and adjunct frames place no restrictions on word order, we need a separate mechanism for controlling word order if we want to avoid the overgeneration represented by the analysis in Figure 12. In the following section, we will show how word order can be controlled by means of a surface tree.

Part II
Discontinuous word order

3 Discontinuous word order

As we saw in the previous section, complement and adjunct frames do not place any restrictions on the word order — that is, given any valid analysis, any permutation of the words in the analysis will result in a valid analysis as well, in terms of complement and adjunct frames. We therefore need special mechanisms for controlling word order, which is the topic of this section.

3.1 Surface trees: landing sites and word order

In a dependency theory, it is natural to assume that the word order in a dependency graph must be controlled by lexical rules associated with the nodes in the graph. The simplest idea is to let the word order be controlled by the deep tree, ie, to let each governor be responsible for the relative word order of its complements and adjuncts. Unfortunately, this simple idea is too crude. The reason is that the global word order is not uniquely specified by a local ordering of the dependents of all gover-

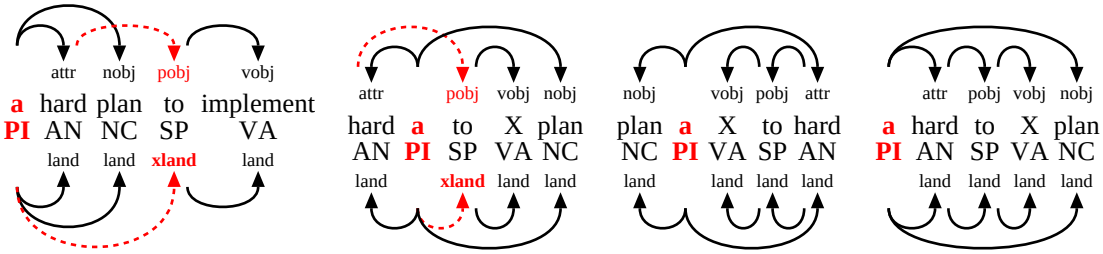

Figure 13: Deep trees with discontinuities (top arcs) and the associated continuous surface trees (bottom arcs) for four word order variants of a dependency tree. Discontinuous dependencies and their associated *xland* edges are shown with dotted lines.

nors because the potential discontinuity of the deep tree allows phrases to be intermingled. Without the discontinuity, the idea would work fine, and for this reason, many dependency parsers solve the word order problem by imposing projectivity on all dependency trees. This is a crude but reasonably effective solution that mirrors the situation in Context-Free Grammar, where phrase-structure trees are required to be continuous (projective), allowing context-free rules to define global word order by specifying both immediate dependency and local linear precedence at the same time. However, the projectivity requirement is problematic from a linguistic point of view because non-projectivity seems to be a universal feature in human languages. For example, the treebanks in the CONLL 2006 shared dependency parsing task include an average of 1.4% non-projective edges and 15% non-projective sentences, with great individual variation between the treebanks (Buchholz and Marsi 2006): eg, 5.4%/36.4% non-projective dependencies/sentences for Dutch, 2.3%/27.8% for German, and 1.0%/15.6% for Danish; some of the treebanks (Chinese, Japanese, Spanish, Bulgarian) impose projectivity on the annotations and therefore have artificially low levels of non-projectivity.

The obvious solution if we want to deal with discontinuous dependencies is to supplement the deep tree with a continuous *surface tree* (or *landing tree*), where all discontinuous edges from the deep tree are replaced with continuous edges by lifting discontinuous dependents upwards until all discontinuous edges have been eliminated, an idea first proposed by Kunze (1968) and refined by Kahane et al. (1998); similar mechanisms have been proposed by Hudson (1998, 2003), Kromann (1999a,b, 2001), and Duchier and Debusmann (2001) and Duchier (2001).[9] It is possible to prove that every non-projective dependency tree has a unique minimal continuous lifting (Buch-Kromann 2006, p. 69-70), which is quite easy to compute and is used in more recent versions of the pseudo-projective parsing algorithm proposed by Nivre and Nilsson (2005).

The node where a dependent lands in the minimal continuous lifting is called its *landing site*, and is defined as the lowest transitive governor in the deep tree that dominates all nodes between the dependent and itself in the linear order. The path in the deep tree from the dependent's governor to its landing site is called its *extraction path*. When the dependency is projective, the landing site coincides with the governor, and the extraction path is empty. When the dependency is non-projective, the extraction path is non-empty, and can be used to express linguistically motivated constraints on non-projectivity, such as island constraints (Ross 1967).

Figure 13 shows a dependency analysis of four word-order variants of the phrase "a hard plan to implement", consisting of discontinuous deep trees (top) and associated continuous surface trees (bottom). The four permutations have the same underlying dependency tree, but the surface trees

[9] (Buch-Kromann 2006, p. 74-75) gives a detailed description of the similarities and differences between these approaches, as well as a comparison with domain unions (Reape 1994) and the word order theory in HPSG (Bouma et al. 2001, §3.1).

differ because of the differences in word order. For example, in the leftmost graph, the surface tree is the result of moving up the dependent "to" from its governor "hard" to its landing site "a". The discontinuous edge and its replacement are shown with dotted arcs. The label *land* is used for surface edges where the governor and the landing site coincide, whereas the label *xland* (external landed node) is used for surface edges where they do not coincide, ie, where the node has been moved upwards.

Like with complement and adjunct dependencies, we will assume that the landing relation is lexically licensed. In particular, we will assume that the relationship between a landed node and its landing site is controlled by means of the following three mechanisms (expressed as rules in a rule-based framework, or as features in a discriminative framework):

(a) *landing rules*: the landing site must license its landed nodes, ie, it must look at the landed node and make a decision as to whether it will allow the node to land or not, based on information about the landed node, such as the word class, the word class of its governor, its primary dependency role, and whether the landing site also functions as the word's governor;

(b) *word order rules*: the landing site must license the local word order of its landed nodes, ie, it must control their relative word order;

(c) *extraction rules*: each node on the extraction path other than the landing site must give permission for the extraction to proceed, ie, it must license its primary dependency edge as a valid extraction edge for the extracted node.

In the following, we will assume that these rules are implemented as weighted constraints or features that assign a penalty score (*cost*) to each node in the graph (ie, lower scores are preferred). As a simple example of how this mechanism can be used to explain the grammaticality of the first word order in Figure 13 and the ungrammaticality of the three others, we will postulate the six weighted constraints (a)–(f) below. For simplicity, we will assume that a local violation cost 100 indicates a completely forbidden, ungrammatical configuration, and that a local cost 50 indicates a dispreferred, unnatural configuration (DG assumes that speakers are often capable of analysing even highly ungrammatical analyses).

- **(a) 100**: pronouns forbid landed nodes on the left
- **(b) 100**: prepositions forbid landed nodes on the left
- **(c) 100**: adjectives forbid landed prepositions on the left
- **(d) 100**: pronouns forbid landed prepositions before landed nouns or adjectives
- **(e) 50**: noun-modifying adjectives dislike landed nodes on the right

Figure 14 shows the violation costs associated with each of the four analyses; from the total cost, we see that the phrase "a hard plan to implement" (cost 0) is more well-formed than the phrase "a hard to implement plan" (cost 50), which is in turn significantly more well-formed than the phrases "hard a to implement plan" (cost 200) and "plan a implement to hard" (cost 300). The violation costs are somewhat arbitrary in this simplified example, and the rules themselves are obviously only a crude approximation to English word order, since there are constructions where these rules are too restrictive (eg, "How good an idea is it?"). But this is not the important point here. What is important is that all the weighted constraints we have formulated above fall out as special instances of a more general word order feature scheme, where we:

- split the landed nodes at a landing site into a left and right landing field, and insert a dummy stop node at the outer periphery of each field (to account for obligatory landing);

	left	middle left	middle right	right
(a)	0	100	100	0
(b)	0	0	100	0
(c)	0	0	100	0
(d)	0	100	0	0
(e)	0	0	0	50
Total	0	200	300	50

Figure 14: The violation costs for the four analyses in Figure 13 computed using the weighted constraints (a)–(e).

L	*Field*	C_l	S_l	N	S_r	C_r
TOP	right	–	–	$\text{a}^{\text{land}}_{\text{root}}$	–	–
TOP	right	–	$\text{a}^{\text{land}}_{\text{root}}$	STOP	–	–
$\text{a}^{\text{land}}_{\text{root}}$	left	–	–	STOP	–	–
$\text{a}^{\text{land}}_{\text{root}}$	right	–	–	$\text{hard}^{\text{land}}_{\text{attr}}$	–	$\text{plan}^{\text{land}}_{\text{nobj}}$
$\text{a}^{\text{land}}_{\text{root}}$	right	–	$\text{hard}^{\text{land}}_{\text{attr}}$	$\text{plan}^{\text{land}}_{\text{nobj}}$	$\text{to}^{\text{xland}}_{\text{pobj}}$	–
$\text{a}^{\text{land}}_{\text{root}}$	right	$\text{plan}^{\text{land}}_{\text{nobj}}$	–	$\text{to}^{\text{xland}}_{\text{pobj}}$	–	–
$\text{a}^{\text{land}}_{\text{root}}$	right	$\text{plan}^{\text{land}}_{\text{nobj}}$	$\text{to}^{\text{xland}}_{\text{pobj}}$	STOP	–	–
$\text{hard}^{\text{land}}_{\text{attr}}$	left	–	–	STOP	–	–
$\text{hard}^{\text{land}}_{\text{attr}}$	right	–	–	STOP	–	–
$\text{plan}^{\text{land}}_{\text{nobj}}$	left	–	–	STOP	–	–
$\text{plan}^{\text{land}}_{\text{nobj}}$	right	–	–	STOP	–	–
$\text{to}^{\text{xland}}_{\text{pobj}}$	left	–	–	STOP	–	–
$\text{to}^{\text{xland}}_{\text{pobj}}$	right	–	–	$\text{implement}^{\text{land}}_{\text{vobj}}$	–	–
$\text{to}^{\text{xland}}_{\text{pobj}}$	right	$\text{implement}^{\text{land}}_{\text{vobj}}$	–	STOP	–	–
$\text{implement}^{\text{land}}_{\text{vobj}}$	left	–	–	STOP	–	–
$\text{implement}^{\text{land}}_{\text{vobj}}$	right	–	–	STOP	–	–

Figure 15: Context lists for the nodes in Figure 15 (left).

- generate a list of context nodes for each landed node, which includes the landing site L, landed node N, the closest locally landed complements C_l, C_r to the left and right, as well as the landed node's immediately adjacent landed siblings S_l, S_r if they are not already on the list;
- generate a list of features for each landed node, using information about the landing field (left/right), as well as the word class, dependency role, and local landing (is it local or extracted?) for each node on the context list.

This results in a total of 3 context lists for each node — one for the node as a landed node, and two for its left and right dummy stop nodes, respectively — as well as a context list for the right stop node at the formal top node for the entire sentence. As an illustration, the 16 context lists for the left analysis in Figure 13 are shown in Figure 15. Context lists capture most, if not all, the information that is needed from a linguistic perspective to determine whether a node is licensed as a landed node, and whether its local word order at the landing site is acceptable or not. In particular, the five word order constraints (a)–(e) can be rephrased as context list patterns, as illustrated in Figure 16. Since it is easy to extract a set of features from the context lists, any reasonable machine learning algorithm should be able to learn weighted constraints like (a)–(e) from the context lists, allowing these features to be used in, say,

Rule	*L*	*Field*	C_l	S_l	*N*	S_r	C_r
(a)–(c)	PI|SP|AN	left	*	*	*	*	*
(d_1)	PI	both	*	*	SP	NC|AN	*
(d_2)	PI	both	*	*	SP	*	NC|AN
(e)	PI_{attr}	right	*	*	*	*	*
(f)	VA	left	–	–	$*^{land}_{subj}$	–	–

Figure 16: Reformulation of weighted constraints (a)–(e) as context list patterns.

	Borsley formulation	**DG formulation**
subject condition	A wh-dependency cannot cross the boundary of a subject.	An extraction path cannot contain a subject edge.
complex NP constraint	A wh-dependency cannot cross the boundary of a clause and an NP that contains it.	An extraction path cannot contain a verb dominated by a (common) noun.[a]
wh-island condition	A wh-dependency cannot cross the boundary of a subordinate wh-question.	An extraction path cannot contain a verb that participates in a wh-construction.[b]
adjunct island constraint	A wh-dependency cannot cross the boundary of an adverbial expression.	An extraction path cannot contain an adjunct edge.
complementizer gap constraint	A wh-dependency gap cannot appear in subject position in a clause introduced by a complementizer.	A verb governed by a complementizer requires a locally landed subject.

Figure 17: A set of island constraints from Borsley (1991, 181–187), and their corresponding formulation in DG.

[a] In our DG analysis, the complementizer "that" is analyzed as a pronoun with an obligatory verbal object, rather than as a complementizer, because of its ability to appear as a subject. For this reason, such complementizer pronouns must be excluded from the DG formulation of the complex NP constraint.

[b] In our DG analysis, subordinate wh-questions are analyzed as relative clauses where the wh-phrase has been relativized (because some Danish subordinate wh-questions include an obligatory relative pronoun, and because the relative clause is usually optional given an appropriate context). Since relative clauses are analyzed as adjuncts, the wh-island condition is a consequence of the adjunct island constraint.

a discriminative parsing or translation system.

3.2 Island constraints

Island constraints (Ross 1967) place restrictions on the possible extraction paths from a word's governor to its landing site. Island constraints are important because they limit how far a node can be extracted, thereby providing indirect restrictions on the word order, and reducing the time complexity of parsing and generation by limiting the number of potential governor-landing site combinations a node can have. All languages are believed to have island constraints which are very similar across languages, although there are slight variations from language to language.

Figure 17 shows a list of island constraints from Borsley (1991, 181–187), and their corresponding formulation in DG as constraints on extraction paths and landing rules. That is, the standard island constraints can be formulated as restrictions on the individual edges in an extraction path. It is not clear that the island constraints will hold universally as they are formulated above (eg, there may be a subset of adjunct roles that should be excepted from the adjunct island constraint). But island constraints in some form probably play a significant role in extraction, and we can expect to get good constraints

on the extraction from treebanks with machine learning techniques that are capable of learning island constraints of this kind.

Since the extraction is set up so that single edge on the extraction path can act as a barrier for the extraction, ie, the extraction can be effectively vetoed by each edge on the extraction path, we have to be careful about how we implement this constraint as a feature. In particular, in a discriminative system, we do not want a negative score for one extraction edge to be compensated by a positive score for another extraction edge on the extraction path. When computing the total score for the extraction path, it might therefore be better to score the extraction path using the worst-scoring edge rather than summing all the edges.

3.3 Examples: topicalizations, scramblings, partial verb phrases

There is a wide range of linguistic constructions where the dependency structure is unremarkable, but where the word order is so complicated that the constructions are viewed as a challenge in linguistic theories. For example, topicalizations, extrapositions, scramblings, and partial verb phrases have received a great deal of attention because of their difficult word order. In this section, we will describe how DG accounts for these phenomena.

Topicalizations

Topicalization has historically been seen as a central phenomenon in virtually all formal syntactic theories, perhaps because it is a relatively frequent word order phenomenon in most languages. In Danish, topicalizations can be explained by assuming that a finite verb allows all nodes other than finite verbs to land in its left field, provided the left field does not contain any other landed node.[10] This rule captures a wide range of topicalizations in Danish, including cases where the topicalized phrase is headed by a preposition or an infinite verb; it excludes all topicalizations where the left field contains more than one normal complement or topicalized node. A similar rule would work for English. Since rules like these can be formulated entirely in terms of restrictions on word order context lists, they should be possible to learn with any reasonable machine learning algorithm.

As an example, consider the dependency structure expressed by the Danish sentence "Jeg har givet Alfred slik" ("I have given Alfred candy"). Danish allows the dependency structure of this sentence to be realized in many different ways which express more or less the same meaning, but with slightly different topic-focus structures. Apart from the canonical word order (Figure 18 left), Danish also allows topicalization of the verb phrase (Figure 18 right), and topicalization of the indirect and direct object (Figure 19). In Figure 18 and 19, the dependency trees are shown above the sentence, and the surface trees below the sentence (fillers are omitted for brevity). Surface edges for extraced nodes are

[10] The landing rule is obviously a crude approximation. For example, vocatives and certain parentheticals are allowed to land in the frontal field, even in the presence of a topicalized phrase, as in examples (1) and (2) below. In (1), the vocative phrase "min elskede" ("my dear") is analyzed as a *voc* adjunct of the finite verb "har" ("have"), and in (2), the parenthetical sentence "og det indrømmer jeg" ("and that I admit") is analyzed as a *modp* adjunct of the finite verb "har" ("have").

(1) Én is, min elskede, har jeg givet Alfred idag.
One ice-cream, my dear, have I given Alfred today.
'I have given Alfred *one* ice-cream today, my dear.'

(2) Én is, og det indrømmer jeg, har jeg givet Alfred idag.
One ice-cream, and that admit I, have I given Alfred today.
'I have given Alfred *one* ice-cream today, and that I admit.'

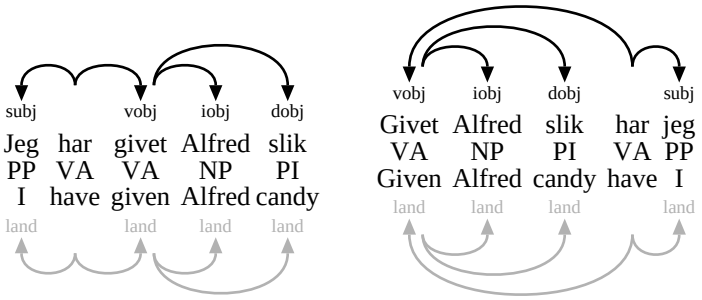

Figure 18: Canonical word order (left) and VP topicalization (right).

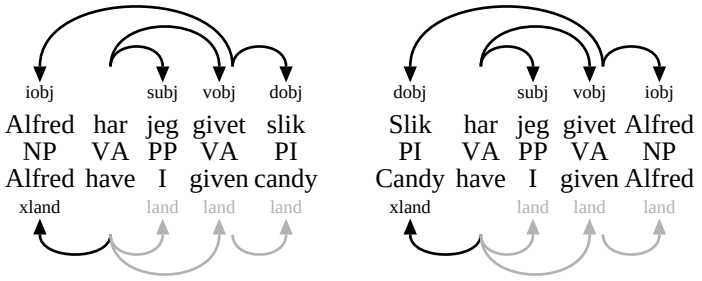

Figure 19: Topicalization of indirect object (left) and direct object (right).

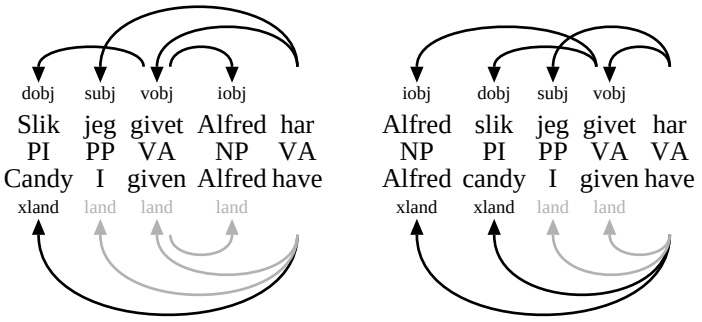

Figure 20: More complicated topicalizations and scramblings that could be used in poetry and songs because of their trochaic metric.

black, whereas surface edges for non-extracted nodes (which are usually omitted in DG visualizations) are grey. The four graphs have the same dependency structure, and only differ with respect to their surface trees: the two surface trees in Figure 18 have the same tree structure, but differ with respect to the ordering of the landed nodes of the finite verb, and in the two surface trees in Figure 19, the topicalized object lands on the finite verb rather than its governor. In Danish, there is an even greater word order variation in poetry and songs, since speakers can relax the usual word order constraints to ensure rhymes and metric. Figure 20 shows two word orders for our original dependency tree, which would be acceptable in songs or poetry. In both examples, the requirement that the frontal field must contain only one landed node is relaxed, and the second example even contains two extracted nodes.

Partial verb phrases

Partial verb phrases (PVPs) in German (Nerbonne et al. 1994, pp. 109–150) is another word order phenomenon where a dependency structure may have several different surface realizations with approximately the same meaning, but slightly different topic-focus structures. DG handles it in the same way as topicalization: by assuming that certain words may function as landing sites for the phrases that have

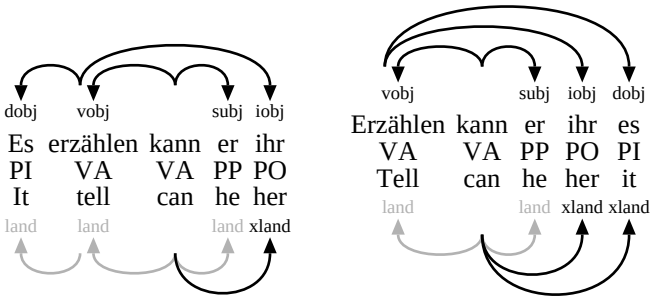

Figure 21: DG analyses of the partial verb phrases (17) and (19).

been extracted from the partial verb phrase. Consider the German sentences below (from Nerbonne 1994, p. 112).

(16) Er kann seiner Tochter ein Märchen erzählen.
He can his daughter a fairy-tale tell.
'He can tell his daughter a fairy tale.'

German allows the words in this sentence to be reordered by topicalizing the embedded verb phrase, but letting some of the dependents in the embedded verb phrase remain in their original position, as in (17)–(19) below.

(17) Ein Märchen erzählen kann er seiner Tochter.
A fairy-tale tell can he his daughter.

(18) Seiner Tochter erzählen kann er ein Märchen.
His daughter tell can he a fairy-tale.

(19) Erzählen kann er seiner Tochter ein Märchen.
Tell can he his daughter a fairy-tale.

We can explain PVPs by assuming that the finite verb does not only allow a topicalized constituent (in this case, the inner VP) to land on it, but also allows one or more extracted NPs to land in its middle field. Like with the other cases of word order, this can be expressed by means of the word order context features described previously. A rule of this kind will allow the analyses of (17) and (19) shown in Figure 21 (to save space, we have replaced all nouns with pronouns). In the analyses, the verbal object is topicalized, and one or more dependents of the verbal object are extracted from the topicalized VP so that they land in the middle field of the finite verb. A DG analysis of (20), a more complicated example of a partial verb phrase, is shown in Figure 22. The VP headed by "erzählen" ("tell") is topicalized, and the indirect object "seiner Tochter" ("his daughter") is extracted from the topicalized VP to the middle field of the finite verb "wird" ("will").

(20) Erzählen wird er seiner Tochter ein Märchen können.
Tell will he his daughter a fairy-tale be-able-to.

Extraposition, scrambling, and cross-serial dependencies

Extraposition, scrambling, and cross-serial dependencies are other examples of word order phenomena where a dependency structure may have several surface realizations, with slightly different topic-focus structures. In extraposition in German (cf. Rambow and Joshi (1994) and Reape (1994)), a "zu" infinitive governed by a verb with verb-last order can be extracted from its canonical position before its governing verb to a position after the verb. For example, in German, the phrase "die Walzer zu spielen" ("to play

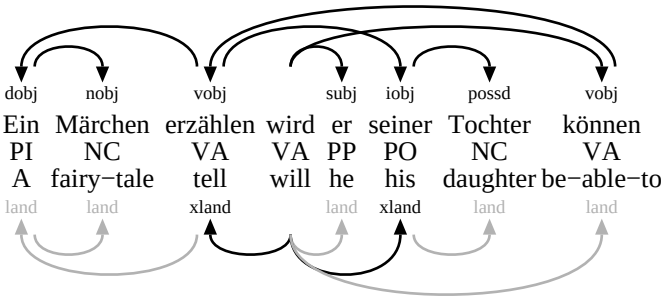

Figure 22: DG analysis of the partial verb phrase (20).

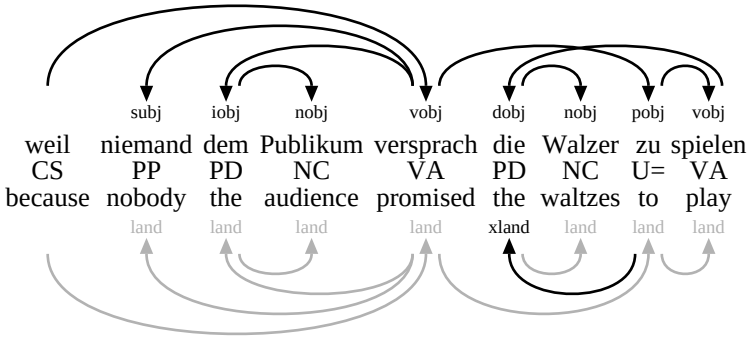

Figure 23: DG analysis of the extraposition (22).

the waltzes") can be moved from a position before "versprochen" to a position after "hat", as shown in (21) and (22).

(21) weil niemand dem Publikum die Walzer zu spielen versprach
because nobody the audience the waltzes to play promised
'because nobody promised the audience to play the waltzes'

(22) weil niemand dem Publikum versprach die Walzer zu spielen
because nobody the audience promised the waltzes to play

These examples can be explained by assuming that German verbs allow "zu" infinitives to land in their right field when they have verb-last order (ie, when they function as verbal objects). The resulting analysis is shown in Figure 23 (with fillers omitted). Note that the infinitive marker "zu" allows all the dependents of its verbal object as non-locally landed nodes, and that an infinitive verb does not allow any left landed nodes when acting as the verbal object of "zu". To control the word order in the left and right field of the verb, we will assume that the left field cannot contain extracted "zu" infinitives, and that the right field cannot contain anything but "zu" infinitives, when the verb is in verb-last order.

Scrambling (cf. Rambow and Joshi 1994) is characterized by a deviation from the canonical word order in the middle field. As an example of scrambling, consider (23) which has been derived from (22) by interchanging the order in which the subject "niemand" ("nobody") and the indirect object "dem Publikum" ("the audience") appears. (24) is identical to (23), except that we have inserted the verb "hat" ("has") into the verbal complex, to demonstrate the non-locality of the extraction.

(23) weil dem Publikum niemand versprach die Walzer zu spielen
because nobody the audience promised the waltzes to play

(24) weil dem Publikum niemand versprochen hat die Walzer zu spielen
because the audience nobody promised has the waltzes to play

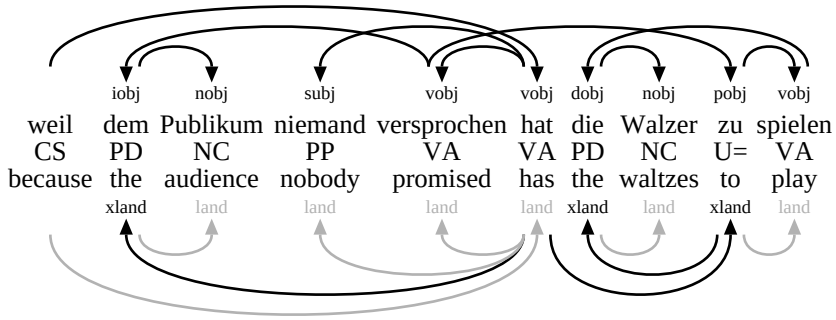

Figure 24: DG analysis of the extraposition and scrambling (24).

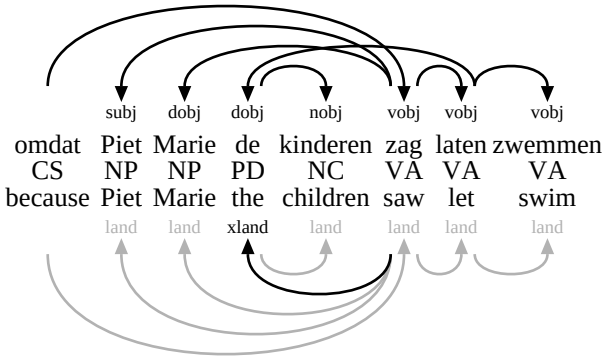

Figure 25: DG analysis of the cross-serial dependency (25).

Scrambling is handled in the same way as topicalization and extraction: by positing a landing rule that allows scrambled phrases to land on a certain word. In a probabilistic language model, scrambling is detected as the possibility that an extracted node can be placed in a non-canonical position relative to the other landed nodes. In the analysis of (24) shown in Figure 24, we assume that the finite verb "hat" ("has") allows the extracted indirect object "dem Publikum" ("the audience") to land on it, and that the scrambled word order is licensed by the finite verb.

Cross-serial dependencies

The DG analysis of cross-serial dependencies follows the same pattern as the DG analysis of other word order phenomena. As an example, consider the Dutch cross-serial dependency (25) (cf. Steedman 1984; Rambow and Joshi 1994). The example is analyzed like other word order phenomena: a dependent (the phrase "de kinderen" ("the children") in (25)) is moved from a subordinate verb ("laten") to the finite verb ("zag"). The resulting DG analysis is shown in Figure 25. The verb "zag" is assumed to be a control verb that passes on a filler for "Marie" to "laten", and "laten" is assumed to be a control verb that passes on a filler for "de kinderen" to "zwemmen"; for simplicity, the fillers generated by the control verbs have been omitted from the graph (we will return to fillers in the following section).

(25) omdat Piet Marie de kinderen zag laten zwemmen
because Piet Marie the children saw let swim
'because Piet saw Marie let the children swim'

Obligatory extraction

Obligatory extraction in Danish is another interesting word order phenomenon. For example, in Danish, objects that have been negated with the determiner "ingen" ("no") are treated in the word order as

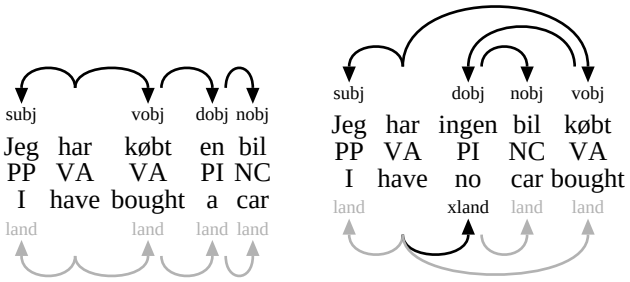

Figure 26: DG analysis of (27b) and the obligatory extraction (26a).

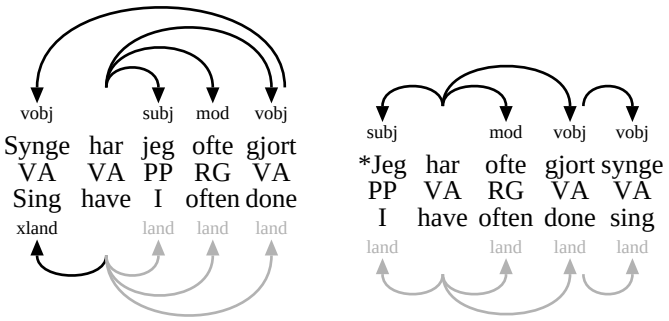

Figure 27: DG analysis of the topicalization (28a) and the ungrammatical canonical word order (28b).

if they were sentence adverbials, ie, they cannot remain in the ordinary object position, but must be obligatorily fronted to sentence adverbial position (where objects are not normally permitted).

(26) a. Jeg har købt en bil idag.
I have bought a car today.

b. *Jeg har en bil købt idag.
I have a car bought today.

(27) a. *Jeg har købt ingen bil idag.
I have bought no car today.

b. Jeg har ingen bil købt idag.
I have no car bought today.
'I have not bought any car today.'

A DG analysis of (26a) and (27b) is shown in Figure 26. To encode this analysis in the DG lexicon, we must ensure that finite verbs such as "har" ("have") license direct and indirect objects headed by negated quantifier pronouns like "ingen" ("no") as external landed nodes, and that the finite verbs are equipped with word order cost functions that ensure that the extracted object ends up in sentence adverbial position.

Another example of obligatory extraction is provided by the Danish verb "gøre" ("do"), which subcategorizes for a direct object or an obligatorily topicalized verbal object. Thus, in the example below, the topicalized word order in (28a) is grammatical, whereas the canonical word order (28b) is ungrammatical. The corresponding DG analyses are shown in Figure 27.

(28) a. Synge har jeg ofte gjort.
Sing have I often done.
'I have often been singing.'

b. *Jeg har ofte gjort synge.
I have often done sing.

'I have often been singing.'

We can explain these phenomena by assuming that the verb "gøre" ("do") is equipped with a cost function that induces a cost whenever the verbal object is non-topicalized, ie, whenever it lands in the right field of "gøre".

In this section, we have sketched how DG can be used to explain a wide range of word order phenomena, such as topicalizations, partial verb phrases, extrapositions, scramblings, cross-serial dependencies, and obligatory extractions. The analyses of these phenomena have followed a general pattern: using the mechanism for identifying landing sites to construct a projective surface tree which can be used to order the words by means of local word order rules, defined in terms of word order features and restrictions on extraction paths.

Part III
Secondary dependencies and simple fillers

4 Secondary dependencies and simple fillers

4.1 Secondary dependencies

In section 2.2, we saw that lexemes are equipped with complement and adjunct frames that license the dependency relations among the lexemes in a dependency graph, and that a lexeme cannot have more than one governor in a well-formed deep tree. In this section, we will show how to uphold this assumption while providing analyses for constructions such as verbal complexes, relatives, and control constructions, where a lexeme with a primary governor seems to satisfy a complement or adjunct role of one or more secondary governors.

For example, in the verbal complex (29) below, the word "truth" seems to satisfy the subject roles of both "will" and "prevail". Similarly, in the control construction (30), "the opera" seems to satisfy both the direct object role of "want" and the subject role of "stop". Finally, in the relative construction (31), "the painting" seems to satisfy both the subject role of "was" and the direct object role of "bought".

(29) Truth will always prevail. (verbal complex)

(30) We want the opera to stop right now. (control)

(31) The painting we bought was a fake. (relative)

A lexeme can even satisfy a dependent role in arbitrarily many governors, as shown by (32), where "they" satisfies the subject role of all verbs in the sentence: "should", "have", "been", "permitted" (analyzed as a passive form), "seek", and "agree".

(32) They should have been permitted to seek to agree on a compromise.

A simple idea for dealing with these examples would be to assume that lexemes can have arbitrarily many governors; but if we did that, "John likes" could be interpreted to mean "John likes John", where "John" acts as the simultaneous subject and object of "likes", and this is not what we want. As in other syntactic frameworks, the key to explaining secondary dependencies is to note that a secondary dependency is not licensed by some particular property of the dependent or its secondary governor, but is lexically licensed by a third lexeme, called the *filler licensor*, that acts as an intermediary between the secondary dependent and its secondary governor. For example, it is a property of a modal verb like "will" that it passes on its subject to its verbal complement; it is a property of a control verb like "want"

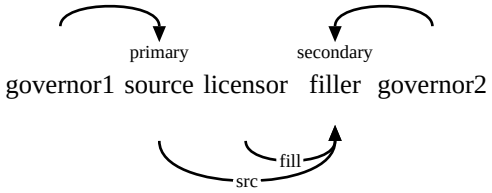

Figure 28: The schematic dependency encoding of the relationship between a filler, filler licensor, and filler source in a dependency graph.

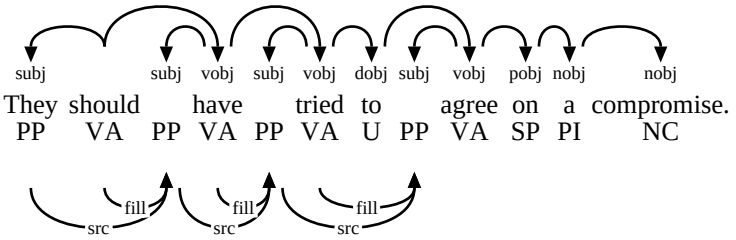

Figure 29: The use of fillers in constructions with verbal complexes and control.

that it passes on its direct object as the subject of the infinitival verb within its "to" complement; and it is a property of any verb that attaches itself as a relative modifier to a noun that it passes on the relativized noun as a secondary dependent somewhere within the relative clause. That is, whenever a word satisfies several dependency roles simultaneously, one of the roles will be a primary role identifying the governor in the deep tree, and the other roles are secondary roles licensed by one or more filler licensors.

4.2 Simple fillers

In DG, we will assume that a filler licensor acts as a special landing site that establishes a secondary dependency by creating a phonetically empty placeholder (called a *filler*) for another node (called the *filler source*). The governor of the filler is sometimes called a *secondary governor* of the filler source, and the filler source is called its *secondary dependent*. Fillers are optional by default, ie, the presence of a filler frame does not mean that the lexeme has to generate a filler, only that it can do so. The obligatory presence of a filler must therefore be controlled by means of a weighted feature that can impose a cost if the lexeme fails to generate a filler, just like the obligatory absence of a filler can be controlled with weighted features. The interpretation of the filler licensor as a landing site has several important consequences. First of all, the filler licensor must dominate (or coincide with) the secondary governor in the deep tree; secondly, the filler's extraction path, from the secondary governor to the filler licensor, is subject to most (if not all) the island constraints that apply to extractions in general. The relationship between the filler, filler licensor, and filler source can be encoded by means of two edges: a surface edge with edge type *fill* from the filler licensor to the filler, and a source edge with edge type *src* from the filler source to the filler, as shown schematically in Figure 28. Figure 29 shows an example of a verbal complex with subject control which is annotated in this way. The intuition behind this analysis is that *'should'*, *'have'*, and *'tried'* each create a phonetically empty filler from their subject that is free to function as a primary dependent for the secondary governor (*'have'*, *'tried'*, and *'agree'*, respectively). Note that in this setup, a filler may well function as filler source for a new filler.

In practice, this annotation is somewhat confusing and cumbersome, especially in treebank annotation where explicit annotation of fillers would require the insertion of a node every time we wanted to make a secondary dependency. In the CDT treebanks, we therefore use a simplified, more intuitive annotation scheme where we eliminate filler nodes entirely: instead, we annotate the filler as a sec-

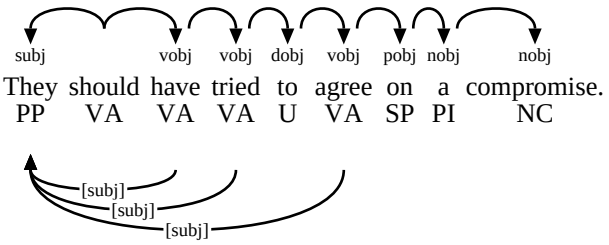

Figure 30: The simplified annotation corresponding to Figure 29.

ondary dependency from the filler governor to the filler source with secondary relation role $[r]$ where r is the corresponding primary dependency role, as shown in Figure 30. This annotation allows us to identify the filler source and the secondary governor, but does not directly identify the filler licensor. However, like with landing sites, the annotation allows the filler licensor to be recovered quite reliably. We will now describe how.

The DG theory requires the filler licensor to dominate the secondary governor, but the filler source is not necessarily a transitive governor of the filler. Since the deep tree is connected, there is always a unique shortest path $P = P_g^s$ from the secondary governor g to the filler source s (the *filler path*). We will assume that the filler licensor always lies on this path as a transitive governor of the secondary governor. With this assumption, we can divide P into two subpaths $P = U + D$ consisting of an upwards *filler up-path* U from the secondary governor to the highest node on the path (the *path root*), and a downwards *filler down-path* D from the path root to the filler source. The filler-up path can in turn be divided into two subpaths $U = E + B$: a *filler extraction path* E from the secondary governor to the filler licensor, and a *barrier path* B from the filler licensor to the path root. The path $S = B + D$ from the filler licensor to the filler source is called the *filler source path*.

The main purpose of subpath B is to allow the secondary dependency to bypass an island constraint that would otherwise act as a barrier for the secondary dependency, something that is needed for the treatment of parasitic gaps (Engdahl 1983). The path E is an extraction path which is subject to island constraints and may be arbitrarily long, whereas the paths B and D are unregulated by island constraints and typically very short (usually no more than two dependencies, and often zero, with the exception of parasitic gaps where the filler source may be farther down if its extraction path includes the path root).

Figure 31 shows our proposed algorithm for reconstructing fillers and filler edges from the simplified annotation. We are assuming that each edge on an extraction path can be classified as a *barrier* or *non-barrier* for extraction, according to whether it blocks extractions or not (eg, the subject island constraint implies that subject edges are barriers in this sense). Moreover, we assume that each edge has an associated positive *barrier cost*; the sum of the barrier costs for an extraction path is called the *extraction cost*, and is defined so that a path always has larger cost that any proper subpath. For simplicity, we will henceforth assume that only adjunct and subject edges act as barriers, with barrier cost 1, while all other edges are non-barriers with a very small barrier cost (say, 0.01) — this provides a very simple implementation of the subject condition and the adjunct island constraint. The reader should verify that with these assumptions, the algorithm produces the filler structure in Figure 29 for the simplified annotation in Figure 30.

DG distinguishes between two kinds of fillers. A *simple filler* is a filler that cannot take any complements or adjuncts, but which can satisfy all dependency roles that could have been satisfied by the filler source. Simple fillers are used in most of our analyses involving fillers, including our analyses of verbal complexes, raising and control, relative clauses, parasitic gaps, and elliptic coordinations with

Input: deep dependency tree T, secondary dependent d, secondary edges E for d.
Output: T augmented with reconstructed fillers and filler edges for E.

$T' := T$ used as initial augmented tree;
$S := \{d\}$ used as initial set of filler sources;
Define $e_g :=$ governor for secondary edge $e \in E$;
Define $E_i :=$ set of $e \in E$ where filler down-path for e has length i ;
Decompose E as $E_0 + \cdots + E_k$;
foreach $i := 0$ to k **do**
  while E_i is non-empty **do**
    $(e, s) :=$ any pair (e', s') in $E_i \times S$ with minimal extraction cost for $U^{s'}_{e'_g}$;
    $B :=$ maximal suffix subpath of $U^{s}_{e_g}$ consisting only of barrier edges;
    $l :=$ first node in B used as filler licensor;
    $r :=$ primary dependency role corresponding to secondary role $[r]$ for e;
    $T' := T +$ new filler $f +$ edges $f \xleftarrow{\text{fill}} l + f \xleftarrow{\text{src}} s + f \xleftarrow{r} e_g$;
    $S := S + \{f\}$;
    $G_i := G_i - \{g\}$;
return T';

Figure 31: Algorithm ReconstructFillers(T, d, E) for augmenting a dependency tree T with reconstructed fillers and filler edges corresponding to the secondary dependencies in E for secondary dependent d.

peripheral sharing. While simple fillers can account for the secondary dependencies in verbal complexes, control constructions, relatives, parasitic gaps, and coordinations with peripheral sharing, they are not powerful enough to deal with cases where the same head has more than one set of dependents, as in gapping coordinations (eg, "We must tutor John, you Alice" or "I had tea and then coffee") and comparatives (eg, "I love tea more than you coffee"). In these cases, we need to use *complex fillers* that represent copies of entire subgraphs in which parts of the copied subgraph can be replaced with new material.

In the following, we will demonstrate how DG handles a wide range of constructions with secondary dependencies that can be handled with simple fillers, including relatives, elliptic coordinations, and parasitic gaps. For each construction, we will explain how it is analyzed in the CDT treebanks, and identify the reconstructed filler licensors. In section 5, we will then describe how DG deals with complex fillers.

4.3 Examples: verbal chains, control constructions, relatives, parasitic gaps

Verbal complexes

In a verbal complex, a modal or auxiliary verb creates a filler that is used to satisfy the subject role of the verbal object. The resulting analysis of verbal complexes with modal and auxiliary verbs is shown in Figure 32 using CDT annotation. The superscripts are used to indicate the filler licensor for each secondary dependency, as computed by the filler reconstruction algorithm in Figure 31, and is not part of the CDT annotation. Ie, the *[subj]*^{1} has reconstructed filler licensor *'should'* and filler source *'He'*, and *[subj]*^{2} has reconstructed filler licensor *'have'* and the filler for *[subj]*^{1} as its filler source.

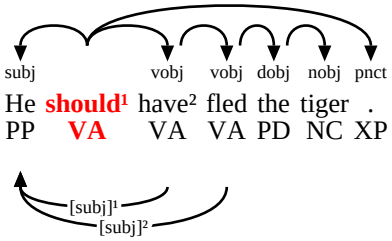

Figure 32: CDT analysis of verbal complexes, with super scripts indicating reconstructed filler licensors.

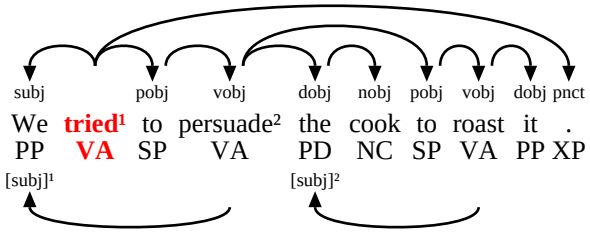

Figure 33: CDT analysis of subject and object control.

Raising and control

Raising and control constructions are analyzed like verbal complexes, ie, the verb is assumed to pass on a filler for one of its dependents to a subordinate verb. An example with subject and object control is shown in Figure 33. Here the *[subj]*^{1} dependency has reconstructed filler licensor *'tried'* and filler source *'We'*, and *[subj]*^{2} has filler licensor *'persuade'* and the filler for *[subj]*^{1} as its filler source. Raising is analyzed in the same way as control constructions. For example, in the raising construction "He seems to lie", we analyze "He" as the subject of "seems" (since there is person and number agreement between the subject and "seems"), and assume that "seems" acts like any other control verb by passing on a filler to the infinitival verb "lie".[11]

Relatives

Fillers can also be used to account for relatives. We assume that the finite verb heading the relative clause always attaches itself as a *rel* adjunct of the relativized constituent and creates a filler that has the relativized constituent as its filler source. Unlike auxiliaries, modals, and control verbs, the relative verb does not place any restrictions on the governor of the filler, which allows the filler to be analyzed as a dependent of the relative verb itself or of any other word within the relative clause. The resulting analysis is illustrated by Figure 34. Note that in relatives, it is not the path root *'The'* which functions as the reconstructed filler source, but the relative verb; this is because the *rel* edge is an adjunct edge, hence a barrier that is identified as part of the barrier subpath.

This analysis is maintained for relatives in which relative pronouns such as "which", "that", and "whose" are present. To avoid a competition between the relative pronoun and the filler with respect to satisfying the secondary dependency, we assume that the relative pronoun obligatorily consumes the filler by taking the filler as its so-called *filler object* (*fobj*). With this analysis, the antecedent of the rel-

[11] It is often argued that the subject of "seems" is not an argument of "seems" from a semantic point of view — in DG terms, this means that in the compositional semantics, the functor associated with "seems" does not consume the semantics associated with "He" as if it was a normal argument. However, from a syntactic point of view, the subject of "seems" behaves as any other subject, so even if it is not an argument of "seems" from a semantic point of view, it is convenient to analyze it as a subject of "seems" from a syntactic point of view.

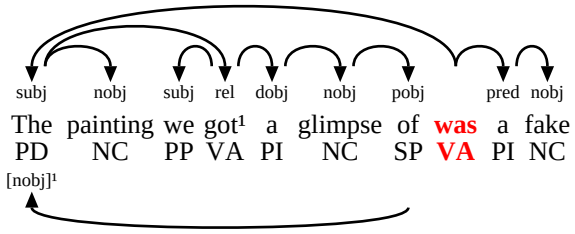

Figure 34: CDT analysis of a relative without a relative pronoun.

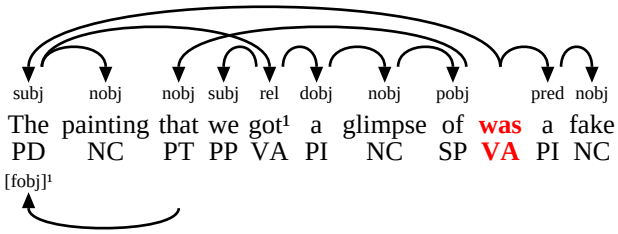

Figure 35: CDT analysis of a relative with a relative pronoun (with *[fobj]* instead of *ref*).

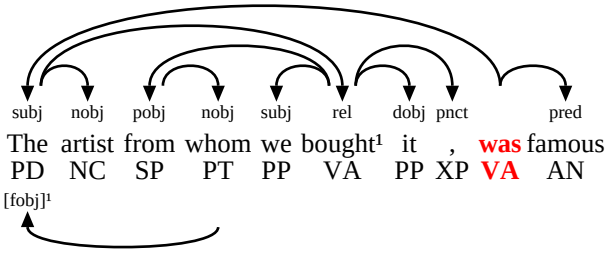

Figure 36: CDT analysis of a relative with an embedded relative pronoun.

ative pronoun is syntactically constrained to be identical to the filler object's filler source. Thus, in this analysis, the relativized noun should be analyzed as a *[fobj]* secondary dependent of the relative pronoun. In the CDT treebanks, we use an inverse *ref* edge (syntactic coreference) as a more immediately intuitive shorthand for the *[fobj]* edge, but in this article, we use the *[fobj]* notation.

Free relatives (or *wh-questions*) are relatives where the relativized phrase is a wh-phrase. Free relatives are analyzed just like other relatives, ie, the relative verb is assumed to create a filler that satisfies some dependency within the relative clause. Three Danish examples of free relatives are shown in (33)–(35) below. The corresponding DG analyses are shown in Figure 37 and 38.

(33) Hvem de vælger er afgørende.
Who they elect is decisive.
'It is decisive who they elect.'

(34) Han spurgte hvem der kom.
He asked who that came.
'He asked about who was coming.'

(35) Alle ved hvor svært det er.
Everybody knows how difficult it is.

The analysis of wh-questions as relatives is supported by the observation that the relative pronoun "der" is explicitly present in (34).

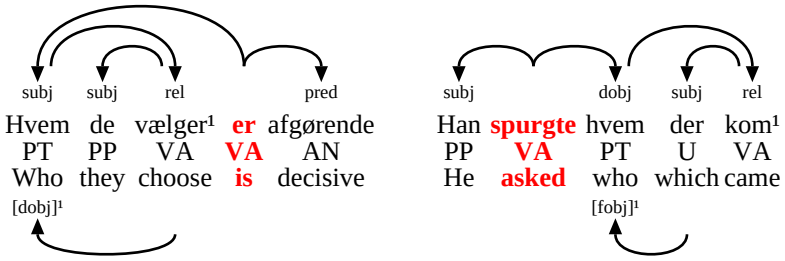

Figure 37: An analysis of the free relatives (33) and (34).

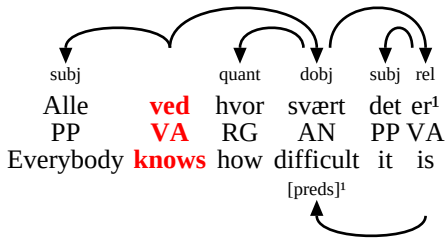

Figure 38: An analysis of the free relative (35).

Elliptic coordination: peripheral sharing

Peripheral sharing (or *right node raising*) is a phenomenon where one or more dependents are shared by several conjuncts in a coordination. It is a relatively frequent construction. For example, approximately 0.66% of all words in the Copenhagen Dependency Treebank for Danish have been marked as heads of a secondary conjunct with peripheral sharing. To account for coordination and sharing in DG, we will assume that a coordination is headed by its first conjunct, and that the remaining conjuncts (called *secondary conjuncts*) attach themselves as *conj* adjuncts of the first conjunct; following Schachter (1977, p. 88) (cf. Hartmann 2000, p. 24), we will assume that coordinators are analyzed as *coord* adjuncts of the heads of secondary conjuncts. Thus, in our analysis, every word is capable of attaching itself as a *conj* adjunct to a preceding word, if the two conjuncts are sufficiently parallel with respect to word class, syntactic structure, and semantics. In this analysis, a shared dependent must be governed by some word within the first conjunct, and the head of the first conjunct must be contained in the shared dependent's extraction path (possibly as the shared dependent's landing site). The heads of all other conjuncts must create a filler with the shared dependent as the filler source; this filler is then used to establish the secondary dependency between the shared dependent and its secondary governors within the other conjuncts.

In the peripheral sharing (36) below, the word "She" functions as the subject of both "was" and "listened", and the word "opera" functions as the nominal object of both "of" and "to" — ie, the words "She" and "opera" are *shared* by the words "was" and "listened", and "of" and "to", respectively. The DG analysis of this example is shown in Figure 39.

(36) She was very fond of, and often listened to, opera.

The head of the second conjunct (*'listened'*) functions as the reconstructed filler licensor for both shared dependents, because the conjunct relation *conj* is an adjunct relation.

Peripheral sharing may involve any number of conjuncts, as shown by (37), where the word "He" functions as the subject of "spotted", "aimed", and "listened", and the phrase "the elephant" functions as the direct object of "spotted" and "shot", and as nominal object of "at". The DG analysis of this example is shown in Figure 40.

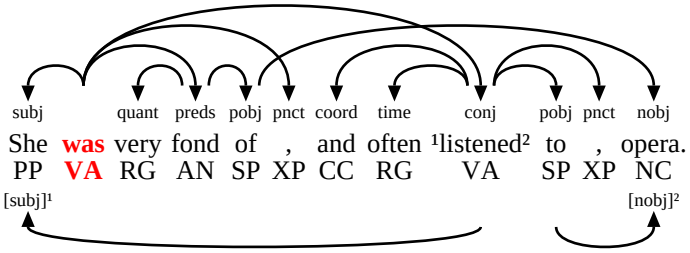

Figure 39: CDT analysis of the peripheral sharing (36).

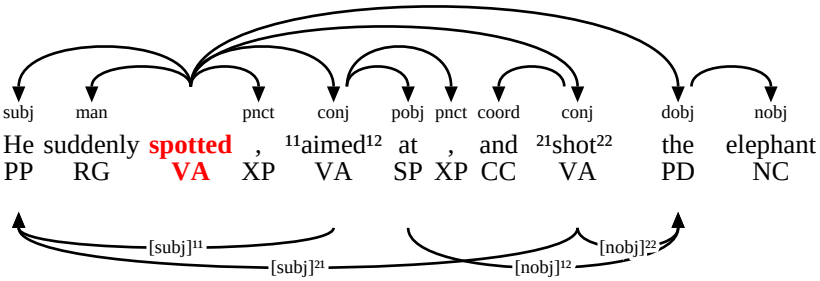

Figure 40: CDT analysis of the peripheral sharing (37).

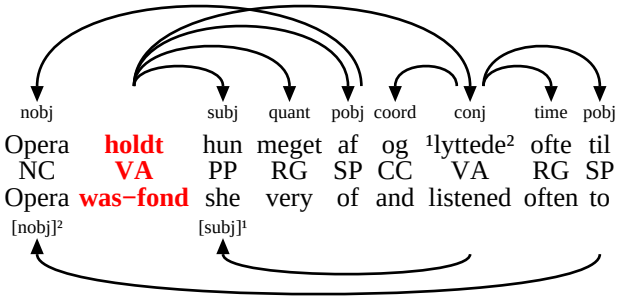

Figure 41: CDT analysis of the peripheral sharing (38) with non-peripheral shared subject.

(37) He suddenly spotted, aimed at, and shot the elephant.

What is characteristic about these examples of sharing is that the shared dependents are located at the periphery of the phrase, ie, the shared phrases are placed either before all non-shared words within the coordination, or after them. In both Danish and English (and many other European languages, we presume), sharing seems to be overwhelmingly peripheral, with one exception in Danish and German, known as *empty subject coordination* in some theories (cf. Hartmann 2000, p. 43ff): non-peripheral subjects can sometimes be shared as well, as illustrated by (38), where the shared subject "hun" ("she") is placed between the heads of the two conjuncts. The DG analysis of this example is shown in Figure 41.

(38) Opera holdt hun meget af og lyttede ofte til.
Opera was-fond she very of and listened often to.
'She was very fond of opera and often listened to it.'

In Figure 39, 40, and 41, all shared dependents land on the head of the first conjunct. Figure 42 (an example due to Kahane 1997) shows our analysis of a peripheral coordination where one of the two shared dependents ("whose mother") has been extracted from the first conjunct. The example is complicated by the presence of fillers for relatives and control.

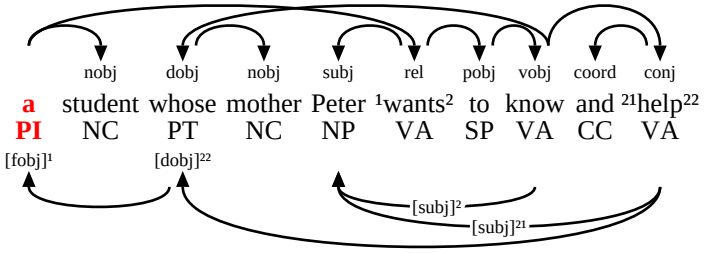

Figure 42: CDT analysis of peripheral sharing with an extracted shared dependent.

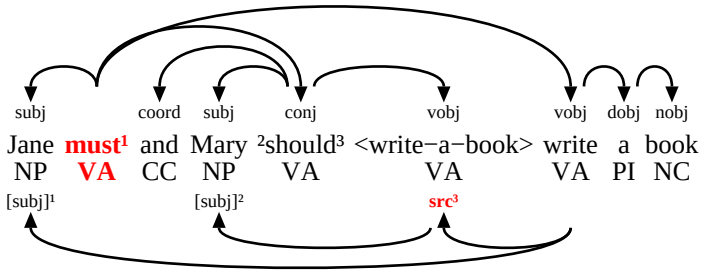

Figure 43: CDT analysis of the peripheral sharing (39) based on gapping fillers rather than simple fillers.

The account of peripheral sharing in this section may be somewhat simplistic, since it cannot handle examples where the shared dependent must receive different filler arguments from different conjuncts, as in (39) where the shared verbal object "write a book" receives the subject "Jane" from the first conjunct, and the subject "Mary" from the second conjunct. The example (40) from Hartmann (2000, p. 104) shows the same phenomenon in German.

(39) Jane must and Mary should write a book.

(40) Peter sollte es und Maria wollte es Hans erzählen.
Peter should it and Maria wanted it Hans to-tell
'Peter should tell it to Hans, and Maria wanted to tell it to Hans.'

It is possible to account for such examples by assuming that secondary conjuncts copy the secondary dependent by means of a gapping filler rather than a simple filler. In (39), the gapping mechanism allows the secondary conjunct to replace the default subject (a filler for "Jane") with the new subject (a filler for "Mary"). Figure 43 shows the resulting DG analysis: instead of encoding the filler for the verbal object *'write a book'* by means of a normal *[vobj]* secondary dependency, we have inserted a complex filler explicitly into the analysis as *'<write-a-book>'*. However, this is a technical complication that we will not go further into in this course.

Parasitic gaps

Simple fillers can also be used to account for parasitic gaps. There is a vast literature on parasitic gaps, summarized by Culicover and Postal (2001) and Parker (1999). Many linguistic formalisms provide an account of them, including GB (Chomsky 1982; Engdahl 1982, 1983), Combinatory Categorial Grammar (Steedman 1987), and HPSG (Pollard and Sag 1994). Parasitic gaps resemble other simple fillers in many respects, except that the filler source path may be much longer — indeed, in all the filler examples we have seen so far, the filler source is directly connected to the filler licensor by just one dependency (ie, the filler source path has length 1). This is typically not the case for parasitic gaps, where the filler source may be far removed from the filler licensor; however, locality is preserved to some degree, because the

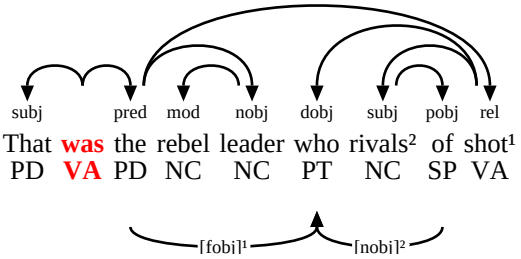

Figure 44: CDT analysis of the parasitic gap (41).

filler licensor is always very close to the landing site or extraction path for the filler source. It should be noted that parasitic gaps are not interesting because of their frequency, which is rather low in written English (although nontrivial in spoken Danish); but parasitic gaps are challenging, which makes them a good testing ground for the mechanisms offered by a linguistic theory.

The examples (41)–(46) below are taken from Pollard and Sag (1994, p. 182–200) (parasitic gaps are indicated with $\__p$, normal gaps with _).

(41) That was the rebel leader who rivals of $\__p$ shot _.

(42) Those boring old reports, Kim filed _ without reading $\__p$.

(43) Which of our relatives should we send snapshots of $\__p$ to_?

(44) Who did my talking to $\__p$ bother _?

(45) Who did you consider friends of $\__p$ angry at _?

(46) Here's the jerk that I expected my pictures of $\__p$ to bother _.

The CDT analysis of (41) is shown in Figure 44, which consists of a normal relative (the *[fobj]*^{1} dependency) and a parasitic gap (the *[nobj]*^{2} dependency). Note that the reconstructed filler licensor for the parasitic gap is *'rivals'* rather than *'shot'*, because subject relations act as barriers and are hence analyzed as part of the barrier path. The filler source for the parasitic gap may be either the relative pronoun *'who'*, or the filler for the relativized noun; both analyses result in the same semantic interpretation.

The examples (47)–(50) below are taken from Haegeman (1994/1991, p. 473–478):

(47) Poirot is a man whom you distrust _ when you meet $\__p$.

(48) Poirot is a man that anyone that talks to $\__p$ usually likes _.

(49) Poirot is a man who I interviewed _ before hiring$__p$.

(50) Poirot is a man who I interviewed _ before wondering when to hire $\__p$.

Example (48) is particularly interesting. The corresponding DG analysis is shown in Figure 45. In direct and indirect object relatives, such as (48), there is an ambiguity with respect to how we select the filler source for the parasitic gap: we can either select the relative pronoun as filler source, or its filler object, which is a filler licensed by the relative verb. The two readings have the same semantic interpretation when the relative pronoun and the filler are coreferent, but it is also possible to find examples where the two readings differ with respect to their interpretation, as in (51) and (52) below, whose analyses are shown in Figure 46. Here, "whose" takes the relative filler for "a man" as its filler object, but "whose motives" is not coreferential with "a man". (51) only allows the interpretation where the direct object of "meet" is coreferential with "Poirot", and (52) only allows the interpretation where the direct object of "meet" is coreferential with Poirot's superiors.

(51) Poirot^1 is a man whose motives you distrust _ when you meet $\__p^1$.

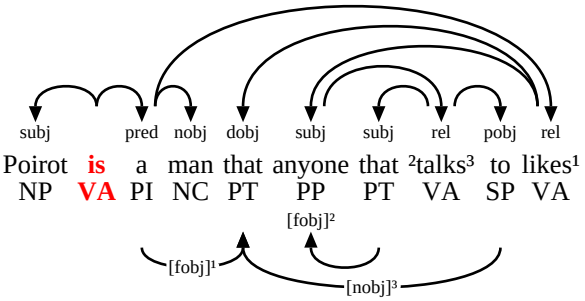

Figure 45: CDT analysis of the parasitic gap (48).

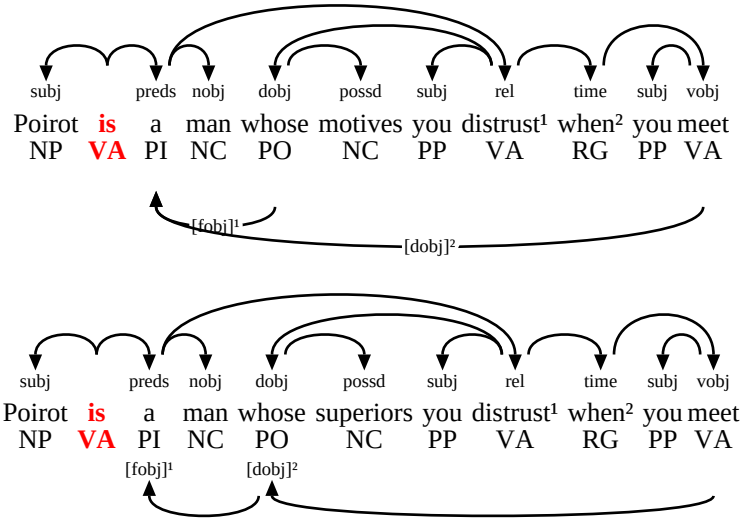

Figure 46: A relative clause in which the parasitic filler may have two possible sources: the relativized noun, and the fronted direct object in the relative clause.

(52) Poirot is a man whose^{2} superiors you distrust _ when you meet $_{-p}^{2}$.

In this section, we have shown how simple fillers can be used to account for constructions that involve secondary dependencies, such as verbal complexes, raising and control, relatives, and parasitic gaps. In the following section, we will demonstrate how complex fillers can be used to account for gapping coordinations where the filler needs to replicate an entire subgraph.

Part IV
Secondary dependencies and complex fillers

5 Secondary dependencies and complex fillers

This part of the course materials will be handed out during the course.

5.1 Complex fillers

5.2 Gapping coordinations

5.3 Speech repairs

Part V
Beyond syntactic dependencies

6 Beyond syntactic dependencies

.

This part of the course materials will be handed out during the course.

6.1 Discourse structure

6.2 Anaphora

6.3 Lexical transformations: morphology

6.4 Periphery and absorption: punctuation

7 Conclusion

In this article, we have sketched how DG analyzes a wide range of natural language phenomena, including word order phenomena such as topicalizations, extrapositions, and scramblings; filler phenomena such as control constructions, relatives, and parasitic gaps; coordination phenomena such as sharing and gapping; punctuation phenomena; morphological phenomena such as inflections, derivations, and lexical alternations; and discourse structure and anaphora. We have described the mechanisms DG provides for identifying the features that can be used to model these phenomena in a natural language system.

Bibliography

Abeillé, Anne and Owen Rambow, eds. 2000. *Tree Adjoining Grammars. Formalisms, linguistic analysis, and processing*. CSLI Publications.

Aduriz, I., M.J. Aranzabe, J.M. Arriola, A. Atutxa, A.D. de Ilarraza, A. Garmendia, and M. Oronoz. 2003. Construction of a Basque dependency treebank. In *Proc. of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 201–204.

Bloomfield, Leonard. 1933. *Language*. Holt, Rinehart and Winston.

Böhmová, Alena, Jan Hajič, Eva Hajičová, and Barbora Hladká. 2001. The Prague Dependency Treebank: three-level annotation scenario. In A. Abeillé, ed., *Treebanks: Building and using syntactically annotated corpora*. Kluwer Academic Publishers.

Borsley, Robert D. 1991. *Syntactic theory*. Edward Arnold.

Borsley, Robert D. 1996. *Modern phrase structure grammar*. Blackwell.

Bosco, C. and V. Lombardo. 2004. Dependency and relational structure in treebank annotation. In *Proceedings of Workshop on Recent Advances in Dependency Grammar at COLING'04*. Geneve, Switzerland.

Bouma, Gosse, Robert Malouf, and Ivan Sag. 2001. Satisfying constraints on extraction and adjunction. *Natural Language and Linguistic Theory* .

Buch-Kromann, Matthias. 2006. Discontinuous Grammar. A dependency-based model of human parsing and language learning. Dr.ling.merc. dissertation, Copenhagen Business School. http://www.id.cbs.dk/∼mbk/thesis.

Buch-Kromann, Matthias. 2009. *Discontinuous Grammar. A dependency-based model of human parsing and language learning*. VDM Verlag.

Buch-Kromann, Matthias, Morten Gylling-Jørgensen, Lotte Jelsbech Knudsen, Iørn Korzen, and Henrik Høeg Müller. 2010. The Copenhagen Dependency Treebank repository. http://code.google.com/p/copenhagen-dependency-treebank.

Buch-Kromann, Matthias, Iørn Korzen, and Henrik Høeg Müller. 2009. Uncovering the 'lost' structure of translations with parallel treebanks. *Copenhagen Studies in Language* 38:199–224.

Buchholz, Sabine and Erwin Marsi. 2006. CoNLL-X shared task on Multilingual Dependency Parsing. In *Proc. CoNLL-2006*.

Chomsky, Noam. 1957. *Syntactic structures.* Mouton, The Hague.

Chomsky, Noam. 1970. Remarks on nominalization. In R. Jacobs and P. Rosenbaum, eds., *Reading in English Transformational Grammar*, pages 184–221. Waltham: Ginn.

Chomsky, Noam. 1982. *Some consequences of the theory of Government and Binding*. Cambridge, Mass.: MIT Press.

Collins, Michael. 1997. Three generative, lexicalized models for statistical parsing. In *Proc. ACL-1997*, pages 16–23.

Culicover, Peter W. and Paul M. Postal. 2001. *Parasitic gaps.* MIT Press.

Dalrymple, M., R. M. Kaplan, J. T. Maxwell III, and A. Zaenen, eds. 1994. *Formal issues in Lexical-Functional Grammar*. CSLI Lecture Notes, no. 47.

Debusmann, Ralph, Denys Duchier, and Geert-Jan Kruijff. 2004. Extensible Dependency Grammar: A new methodology. In *Proc. Recent Advances in Dependency Grammar, COLING-2004*.

Dowty, David R. 1992. Towards a minimalist theory of syntactic structure. In W. Sijtsma and A. van Horck, eds., *Discontinuous constituency*. Mouton de Gruyter.

Duchier, Denys. 2001. Lexicalized syntax and topology for non-projective dependency grammar. In *Joint Conference on Formal Grammars and Mathematics of Language FGMOL-01, Helsinki, August 2001.*

Duchier, Denys and Ralph Debusmann. 2001. Topological dependency trees: A constraint-based account of linear precedence. In *39th Annual Meeting of the Association for Computational Linguistics (ACL 2001), Toulouse, France.*

Eisner, Jason M. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. COLING-96*, pages 340–345.

Engdahl, Elisabeth. 1982. Restrictions on unbounded dependencies in Swedish. In E. Engdahl and E. Ejerhed, eds., *Readings on unbounded dependencies in Scandinavian languages.* Stockholm: Almquist and Wiksell.

Engdahl, Elisabeth. 1983. Parasitic gaps. vol. 6, pages 5–34. Springer.

Gamut, L. T. F. 1991. *Logic, language, and meaning I+II*. University of Chicago Press.

Haegeman, Liliane. 1994/1991. *Introduction to Government and Binding theory.* Blackwell, 2nd edn.

Hartmann, Katharina. 2000. *Right node raising and gapping: interface conditions on prosodic deletion.* John Benjamins.

Helbig, Gerhard. 1992. *Probleme der Valenz- und Kasustheorie.* Max Niemeyer Verlag.

Helbig, Gerhard and Wolfgang Schenkel. 1971/1969. *Wörterbuch zur Valenz und Distribution deutscher*

Verben. Max Niemeyer Verlag.

Hellwig, Peter. 2003. Dependency unification grammar. In V. Agel, L. M. Eichinger, H. Eroms, P. Hellwig, H. Heringer, and H. Lobin, eds., *Dependency and valency. An international handbook of contemporary research*. Mouton.

Hudson, Richard. 1990. *English Word Grammar*. Blackwell.

Hudson, Richard. 1998. Trouble on the left periphery. Unpublished manuscript.

Hudson, Richard. 2003. An encyclopedia of English grammar and Word Grammar. http://www.phon.ucl.ac.uk/home/dick/enc-gen.htm.

Hudson, Richard. 2004. Are determiners heads? *Functions of Language* 11(1).

Hudson, Richard. 2010. *An Introduction to Word Grammar*. Cambridge University Press.

Järvinen, Timo and Pasi Tapanainen. 1998. Towards an implementable dependency grammar. In *COLING-ACL'98, Montreal*, pages 646–52.

Jespersen, Otto. 1984/1937. *Analytic syntax*. University of Chicago Press.

Joshi, Aravind and Owen Rambow. 2003. A formalism for dependency grammar based on Tree Adjoining Grammar. In *Proc. of the Conference on Meaning-Text Theory*.

Joshi, Aravind and Yves Schabes. 1997. Tree-adjoining grammars. In *Handbook of Formal Languages. Beyond Words*. Springer-Verlag.

Kahane, Sylvain. 1997. Bubble trees and syntactic representations. In Becker and Krieger, eds., *Proc. MOL-1997*. DFKI.

Kahane, Sylvain, Alexis Nasr, and Owen Rambow. 1998. Pseudo-projectivity: a polynomially parsable non-projective dependency grammar. In *COLING-ACL'98, Montreal*, pages 646–52.

Keson, B. and O. Norling-Christensen. 1998. PAROLE-DK. Det Danske Sprog- og Litteraturselskab.

Kromann, Matthias T. 1999a. Towards Discontinuous Grammar. In A. Todirascu, ed., *Proceedings of the ESSLLI 1999 Student Session*, pages 145–156.

Kromann, Matthias Trautner. 1999b. Towards discontinuous grammar formalisms with lexical notions of valency. Master's thesis. Department of Computational Linguistics, Copenhagen Business School.

Kromann, Matthias T. 2001. Optimality parsing and local cost functions in Discontinuous Grammar. In *Proceedings of the Joint Conference on Formal Grammar and Mathematics of Language (FGMOL-01), Helsinki, August 10-12, 2001. Electronic Notes in Theoretical Computer Science 53*.

Kromann, Matthias T. 2003. The Danish Dependency Treebank and the DTAG treebank tool. In *Proc. Treebanks and Linguistic Theories 2003*.

Kruijff, Geert-Jan. 2001. *A categorial-modal logical architecture of informativity: Dependency Grammar Logic and information structure*. Ph.D. thesis, Charles University.

Kruijff, G.-J.M. 2006. Dependency grammar. In K. Brown, ed., *Encyclopedia of Language & Linguistics*, pages 444–450. Oxford: Elsevier. ISBN 978-0-08-044854-1.

Kunze, Jürgen. 1968. The treatment of nonprojective structures in the syntactic analysis and synthesis of English and German. *Computational Linguistics* 7:67–77.

Kurohashi, Sadao and Makoto Nagao. 1998. Building a Japanese parsed corpus while improving the parsing system. In *Proc. of The First International Conference on Language Resources and Evaluation*, pages 719–724.

Marcus, M., G. Kim, M. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *ARPA Human Language Technology Workshop*.

Melcuk, Igor A. 1988. *Dependency syntax: Theory and practice*. State University of New York Press.

Menzel, Wolfgang and Ingo Schröder. 1998. Decision procedures for dependency parsing using graded constraints. In S. Kahane and A. Polguère, eds., *Proc. Coling-ACL Workshop on Processing of*

Dependency-based Grammars, Montreal, Canada, pages 78–87.

Miltsakaki, Eleni, Rashmi Prasad, Aravind Joshi, and Bonnie Webber. 2004. The Penn Discourse TreeBank. In *Proc. of LREC*.

Mladová, Lucie, Šarka Zikánová, and Eva Hajičová. 2008. From sentence to discourse: Building an annotation scheme for discourse based on Prague Dependency Treebank. In *Proc. 6th International Conference on Language Resources and Evaluation (LREC 2008)*.

Montague, Richard. 1974. *Formal philosophy: selected papers of Richard Montague*. Yale University Press.

Nerbonne, John. 1994. Partial verb phrases and spurious ambiguities. In Nerbonne et al. (1994), pages 109–150.

Nerbonne, John, Klaus Netter, and Carl Pollard, eds. 1994. *German in Head-driven Phrase Structure Grammar*. University of Chicago Press.

Nivre, Joakim. 2005. Dependency grammar and dependency parsing. MSI report 05133, Växjö University: School of Mathematics and Systems Engineering.

Nivre, Joakim and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of ACL 2005*.

Oflazer, Kemal, Bilge Say, Dilek Zeynep Hakkani-Tür, and Gökhan Tür. 2003. Building a Turkish treebank. In A. Abeillé, ed., *Treebanks: Building and Using Parsed Corpora (Text, Speech, and Language Technology)*, pages 261–277. Springer.

Parker, Anna R. 1999. *Parasitic gaps in the Germanic languages*. Master's thesis, Department of Linguistics, National University of Ireland, Dublin.

Pollard, Carl and Ivan A. Sag. 1994. *Head-driven Phrase Structure Grammar*. University of Chicago Press.

Pustejovsky, James. 1995. *The generative lexicon*. MIT Press.

Rambow, Owen and Aravind K. Joshi. 1994. A processing model for free word order languages. In C. Clifton, L. Frazier, and K. Rayner, eds., *Perspectives on sentence processing*, pages 267–301. Lawrence Earlbaum Associates.

Reape, Mike. 1994. Domain union and word order variation in German. In Nerbonne et al. (1994), pages 151–197.

Ross, John Robert. 1967. *Constraints on variables in syntax*. Ph.D. thesis, MIT.

Schachter, P. 1977. Constraints on coordination. *Language* 53:86–103.

Schröder, Ingo. 2002. *Natural language parsing with graded constraints*. Ph.D. thesis, Univ. of Hamburg.

Sgall, Petr, Eva Hajičová, and Jarmila Panevová. 1986. *The meaning of the sentence in its semantic and pragmatic aspects*. Reidel, Dordrecht.

Sleator, Daniel and Davy Temperley. 1993. Parsing English with a Link Grammar. In *Third International Workshop on Parsing Technologies*.

Starosta, Stanley. 1988. *The case for Lexicase: An outline of Lexicase grammatical theory*. Pinter Publishers.

Steedman, Mark. 1984. On the generality of the nested dependency constraint and the reason for an exception in Dutch. In B. Butterworth, B. Comrie, and O. Dahl, eds., *Explanations for language universals*. Mouton.

Steedman, Mark. 1987. Combinatory grammars and parasitic gaps. *Natural Language and Linguistic Theory* 5:403–439.

Steedman, Mark. 2000. *The syntactic process*. A Bradford Book, The MIT Press.

Tesniere, Lucien. 1959. *Éléments de syntaxe structurale*. Klincksieck.

Vater, Heinz. 1978. On the possibility of distinguishing between complements and adjuncts. In W. Abra-

ham, ed., *Valence, semantic case and grammatical relations*, pages 21–45. John Benjamins, Amsterdam.