Institut für Informatik
der Universität Zürich

Sommer 1998
- Summer 1998

Lizentiatsarbeit - Diploma Paper
Supervisor: Prof. Dr. Michael Hess

ExtrAns Research Report: Dependency vs. Constituency

A Linguistic Comparison of Constituency, Dependency and Link Grammar

Gerold Schneider

Büro 27-K-46
Institut für Informatik
Winterthurerstr. 190
CH - 8057 Zürich
+41 1 / 635 67 24
gschneid@ifi.unizh.ch

Lerchenhalde 43
CH - 8046 Zürich
+41 1 / 371 85 06

Version 1.1
Juli – July 1998

Notes on the Present Version 1.1

The revisions between version 1.0 and 1.1 of this paper were based on the supervisor's comments. They consist in the correction of about two dozen small errors, some printing mistakes, some stylistic improvements and the addition of some explanatory footnotes. No fundamental changes were made.

It has to be pointed out, however, that chapter 6 is not well enough researched and contains a fundamental mistake, which would necessitate a major revision. The supervisor, however, preferred to accept the paper without such a revision. The fundamental mistake I have made is that the topic-focus articulation (TFA) dependency structures presented there have nothing to do with a logical representation and are not suitable as an intermediate step towards them, either. TFA dependency structures should rather be treated and therefore also presented and defended as a discourse representation structure (DRS). Please bear this in mind when reading chapter 6 and section 2.3.2.2, which introduces the same mistake.

Comments and criticisms are always welcome, although I will no longer be able to integrate any of them into this paper. I hope, nevertheless, that it will be possible for me to come back to some of the topics dealt with at a later stage.

I hope that you will enjoy reading this paper

Gerold Schneider

O. Table of contents

0.1 Overview

The detailed table of contents follows on the next page:

0.2 Detailed Contents

1. Introduction

The goal of this paper is to compare the linguistic adequacy of constituent analysis versus dependency analysis, with special focus on the dependency-related Link Grammar Parser described in [Sleator & Temperley 1993], and to show how far Link Grammar, dependency and constituent structures can be converted into each other.

This paper does not intend to be a comprehensive introduction to Link Grammar, nor to dependency grammar. Readers are referred to [Sleator & Temperley 1993, 1998b] for an introduction to Link Grammar and to [Fraser 1996], [Weber 1997] and [Tarvainen 1981] for an introduction to dependency grammar. Some basic concepts of dependency grammar will be discussed in chapter 2, however. Chapter 5 introduces some of the link types used in Link Grammar.

1.1 Link Grammar and Its Linguistic Background

At the start of a University project on passage retrieval, the ExtrAns project at the University of Zürich, a broad-coverage grammar was sought after. The two candidates were the Link grammar system [Sleator & Temperley 1991, 1993, 1995, 1998a, 1998b, 1998c] and the Alvey Natural Language Tools [Grover 1993]. Mollá [1997] discusses the pros and cons of the two systems. The disadvantage of Link Grammar is its "unusual" formalism, the disadvantage of the Alvey NL Tools its slow speed and the fact that it often reports hundreds of ambiguous parses. It was decided to use Link Grammar.

It is theoretically possible to write one's own grammar in the formalism of Link Grammar, but Link Grammar is delivered together with a well-developed broad-coverage English grammar, which exhausts all the facilities of the grammar and very closely interacts with the grammar formalism. I will therefore not make a distinction between the grammar formalism and the English Link Grammar, which is an instance of this formalism.

The questions about Link Grammar's *linguistic performance* and its unusual, if not unique *formalism* became a focus of our attention and hence the first topic of this paper. By "unusual" I mean the fact that it is not based on a PSG grammar of some sort, that the reported sentence structures are not constituent tree structures as is the case in the vast majority of parsers. Link Grammar structures are – as the name suggests – links between the words of a sentence. Link Grammar is very closely

related to dependency grammars, which were first formally expressed by Gaifman [1965], as also the authors of Link Grammar point out:

> ... there is a very close relationship between [Dependency Systems] and link grammars. ... It is easy to take a dependency grammar in Gaifman's notation and generate a link grammar that accepts the same language.
>
> (Sleator & Temperley 1993:12)

The *theoretical background of dependency grammar* has therefore become the second topic of this paper (the one I have dealt with most thoroughly), the *relationship between dependency and "usual" PSG grammars* a third, and *the relationship between dependency and Link Grammar* a fourth topic. When this paper was almost finished, the new version 3.0 of Link Grammar became available and I did not have enough time left to carry out all the necessary revisions, but at least some. As for the fourth topic, I was no longer surprised to find the authors suggest a more careful formulation of the relationship between dependency and Link Grammar.

> The structure assigned to a sentence by a link grammar is rather unlike any other grammatical system that we know of (although it is certainly related to dependency grammar).
>
> (Sleator & Temperley 1998b)

1.2 Relations and Questions

As mentioned, the paper aims to address the following questions:

- **Link Grammar vs. Dependency Grammars:** Is the relationship between Link Grammar and dependency grammars as close as we are led to believe by the (Sleator & Temperley 1993:12) quotation above? What are the consequences of the differences? These questions are addressed in chapter 4.
- **Constituency vs. Dependency:** What are the linguistic differences between constituency and dependency? Are there expressions that dependency formalisms or constituency formalisms fail to express? Up to which point are constituency and dependency linguistically equivalent? Chapter 2 takes up these questions in detail.
- **A description of the grammar provided with Link Grammar:** Link Grammar is delivered together with a broad-coverage English grammar. I will take a look at this grammar in chapter 5.
- **Syntactic vs. Semantic: The Semantic Interface:** Because of its use of syntactic relations as primitives, dependency systems have often been said to be particularly close to semantics. Is this true? What are the problems likely to be encountered? I will only briefly touch on this topic in chapter 6.

- **Practical Experiments:** Although this paper is mainly theoretical, I could not resist the temptation to add some practical experiments with dependency to this paper. I believe that it is linguistically more rewarding to experiment with dependency rather than in the framework of Link Grammar, which is, first, limited by the formalism and, secondly, exhausted by the well-developed English grammar provided with Link Grammar. Two very simple experimental dependency parsers are presented in chapter 3.
- **Practical Performance of Link Grammar:** Two articles about precisely this topic exist already [Sutcliffe et al. 1995, 1997], therefore no further elaboration on the topic seems necessary. Using the first 70 sentences from a technical manual (the *Lotus (R) AmiPro for Windows User's Guide 3.0*) Sutcliffe et al. [1995: 6-7] report: "the overall success rate is surprisingly high considering the complexity of the document and the selectivity of the parser – a grammatical analysis could be produced for 86% of utterances overall". The performance of Link Grammar has increased in the new version 3.0, which has become available too recently to allow me to make extensive tests. For these reasons I have decided to excluded this chapter. A test conducted by the authors of Link Grammar themselves [Sleator & Temperley 1998c] reports the following results:

> In March of 1998 we did a test of the speed and coverage of version 3.0 of the parser, using a random block of 100 sentences from the Penn Treebank corpus. ... The sentence had an average length of 25 words; the text was of the "financial news" type.
>
> The parser took 276 seconds total, a mean of 2.76 seconds per sentence, on a 266 MHz Pentium II. The parser correctly identified 82.1% of the constituents. On 37 of the 100 sentences, the parser's preferred linkage (the first one outputted) was the correct one.
>
> (Sleator & Temperley 1998c)

1.3 The Scientific Status of Linguistics

1.3.1 To Speak or to Silence, That Is the Question

Before starting to make or refute any statement or "scientific" claim, it is important to consider the fragile status of linguistics as a science.

Any philosophical system, any science has to start with assumptions, axioms which cannot be really proved or disproved, which are fundamentally arbitrary but hopefully convincing. In his *Tractatus Logico-Philosophicus*, Wittgenstein [1918] writes that the only true philosophy would be to utter proven scientific facts, to use nothing but defined symbols of a defined formalism – i.e. to renounce on metaphysics and thus on philosophy:

> Die richtige Methode der Philosophie wäre eigentlich die: Nichts zu sagen, als was sich sagen lässt, also Sätze der Naturwissenschaft – also etwas, was mit Philosophie nichts zu tun hat–, und dann immer, wenn ein anderer etwas Metaphysisches sagen wollte, ihm nachzuweisen, dass er gewissen Zeichen in seinen Sätzen keine Bedeutung gegeben hat. ...
>
> (Wittgenstein 1918: 85, § 6.53)

Wittgenstein is aware that the problems with this suggestion are, however, that every definition necessitates a definition of the defining terms until we reach the unprovable maxims. If we refuse to accept these fundamental maxims, the cornerstones of meaning, we cannot state anything and are condemned to remain silent.

> Wovon man nicht sprechen kann, darüber muss man schweigen.
>
> (Wittgenstein 1918: 85, § 7)

These maxims have transcendental, metaphysical quality, only they make any meaning possible and can thus instantiate our questions and answers in life.

> Wir fühlen, dass, selbst wenn alle *möglichen* wissenschaftlichen Fragen beantwortet sind, unsere Lebensprobleme noch gar nicht berührt sind. Freilich bleibt dann eben keine Frage mehr; und eben dies ist die Antwort.
>
> (Wittgenstein 1918: 85, § 6.52)

Only the transcendental character of metaphysical philosophy can really give answers and assert meaning. If we only utter scientific proven facts we can only replace meaningless utterances with one another. E.g. in semantics we can step from language to metalanguage to meta-meta-etc.-langauge, but this does not bring us an inch closer to real meaning.

On the other hand, because we cannot define the maxims we use, we remain incompetent about them nevertheless. Again, Wittgenstein's famous quote applies:

> Wovon man nicht sprechen kann, darüber muss man schweigen.
>
> (Wittgenstein 1918: 85, § 7)

We are therefore in principle disqualified from speaking, from stating anything meaningful or even "scientific".

If we accept a minimal set of maxims on which everybody agrees science seems to be possible nevertheless, as long as we can base everything on these maxims.

1.3.2 There Is no Linguistic Structure

Yngve [1996] criticizes that linguistics is still lacking a proper scientific foundation. Linguistics needs to accept more maxims than those accepted in natural science. Even basic linguistic concepts like word classes or wordhood itself remain unprovable and, in Yngve's view, unscientific.

> The conventional wisdom that a language like English has a real existence led to the idea that we only have to discover the structure of *the* language. The question than became: *How* could we discover its structure and the structure of language in general? In later years it was thought, optimistically, that it was

> simply a problem of the blind men, each examining a different part of an elephant. It was thought that more work was needed on devising better test criteria and on how to interpret the results of the tests, and that these results would eventually converge on a consistent overall description of the elephant.
>
> But, as one of my students gasped when realizing the implications of the domain confusions, "There is no elephant!"
>
> (Yngve 1996: 46)

On the one hand this is true, on the other hand linguistics wavers between natural science and human science, in the latter nothing can be taken for granted, and we have to assume additional maxims if we want to be able to infer anything. Until neurology – which is a natural science - progresses considerably, hypotheses is all we can have in linguistics.

But it would be illusory and unscientific to forget that this entails that every linguistic classification we make is ultimately a conjecture (to a higher degree than in natural science), a metaphor (which may, nevertheless, work nicely in many cases), but for which we can never claim evidence of any sort. Linguistic categories and structures are at best tools but in principle unable to express any underlying truth.

The second danger linguists tend to succumb to is to base arguments on evidence internal to the theory. The more internal arguments a theory uses the more it becomes an ideological end in itself and the less reliable it grows, because theory-internal arguments are unreliable themselves already.

1.3.3 Arbitrary Grammatical Decisions

Some linguistic decisions are indeed arbitrary in nature. It is important to admit that they are. It can be postulated that linguistic structure exists in a certain way, but other contradicting postulations may be as valid. Examples are easy to find:

- In many cases, the grammatical class of a word form cannot be disambiguated from the context. Quirk et al. [1985] discuss examples of ambiguity between verb or noun (“I hate lying “[15.13: 1065]), adjective or participle (“people involved” [7.21:419]) .
- Even the set of admissible word classes is not clear-cut. E.g., while Quirk et al. [1985] treat *ago* as a postposed adverb, it is just as convincing to see it as a postposition – the only English postposition.
- As we will see in this paper, the debate if the subject or the verb should be the “core” of a sentence is still undecided, perhaps undecidable.
- It is equally debatable for many structures which word form should be regarded as its head, as we will see in 2.3.2.

Yngve [1985: 43] builds up his arguments to point out that there is no necessity to accept Chomsky’s transformations or indeed any of his other postulations. What I

hope to show in the following will rather be that dependency or Link Grammar or constituency are all metaphors that allow us to construct structures that often could be underlying structures of language, as if such an underlying "elephant" existed. For certain phenomena it is even possible to suggest which of these fundamentally different approaches to grammar could be less linguistically equivalent than the others.

1.4 Acknowledgements

I would like to thank the following persons for fruitful linguistic discussions and suggestions (in alphabetical order): Dr. Diego Mollá Aliod, Lumme Erilt, Prof. Michael Hess, Timo Järvinen, Hans-Martin Lehmann, Dr. Martin Volk.

2. Constituency versus Dependency

Dependency, although less known among linguists than constituent analysis, is an intuitive concept. In constituency, a sentence *consists of* certain elements which in turn consist of other elements or words. In dependency, one word form *depends* on the other. Especially morphological dependencies, so-called agreement, are self-evident. If we are ready to accept that, on the morphological level, a masculine article evidently depends on a masculine noun, we can start to develop dependency to a concept.

In other words, dependency is a grammar in which individual words both act as terminal nodes and as non-terminal nodes. They are terminal because they directly access the lexicon, because in its purest form, dependency only knows words; and they are non-terminal because they "require", they "subcategorize for" other words, so-called dependents. Because dependency is an intuitive concept, it is an old concept:

> 'Dependency analysis' is an ancient grammatical tradition which can be traced back in Europe at least as far as the Modistic grammarians of the Middle Ages, and which makes use of notions such as 'government' and 'modification'. In America the Bloomfieldian tradition (which in this respect includes the Chomskyan tradition), assumed constituency analysis to the virtual exclusion of dependency analysis, but this tradition was preserved in Europe, particularly in Eastern Europe, to the extent of grammar teaching in schools. However, there has been very little theoretical development of dependency analysis, in contrast with the enormous amount of formal, theoretical, and descriptive work on constituent structure.
>
> (Hudson 1996: 369-70)

Despite its tradition, dependency seems to have been overshadowed by constituency more recently, especially since the start of 'modern' grammar theory.

> [P]hrase structure representation in syntax was strongly promoted by the Structuralist school during the thirties, forties and fifties (...). It became the only syntactic representation ever seriously discussed in the work of Noam Chomsky and the Transformational-Generative School he founded in the late fifties. As a result of the triumphal offensive of the transformational-generative approach throughout the world, phrase-structure syntax forced dependency syntax into relative obscurity.
>
> (Mel'čuk 1988: 3-4)

Let us first take a look at the some of the basic concepts of dependency, to get used to its way of thinking.

2.1 Tesnière

For an introduction to Tesnière see Weber [1997] and Gréciano [1996]. This subchapter is mainly based on them. Lucien Tesnière, Professor for Comparative Linguistics at the University of Montpellier from 1937 to his death in 1954, is the undoubted father of

dependency [Weber 1997: 11, Mel'čuk 1988: 3]. Already working on his theory in World War I, a number of tragic incidences postponed the publication of his major work [Tesnière 1959] until after his death. It appeared posthumously, almost unnoticed in a time moulded by Chomsky's constituent syntactic structures and the prevalence of phrase structure grammar.

> Als er seine tragenden Ideen in den 30er Jahren entwickelte, war er seiner Zeit voraus; als aber sein immer wieder verzögertes Hauptwerk [Tesnière 1959] 1959 postum erschien, blieb es teilweise hinter dem Stand der damaligen Forschung zurück und entsprach nicht mehr ganz dem Erwartungshorizont der damaligen Sprachwissenschaft. In der Folge standen meist nicht mehr die Ideen Tesnières, sondern die der generativen Grammatik im Zentrum der linguistischen Aufmerksamkeit.
>
> (Helbig 1996: 41)

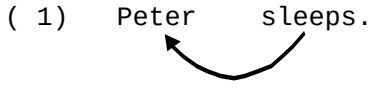
Tesnière makes the distinction between the *outer form*, the 'ordre linéaire', the surface string of words in the text, the 'chaîne parlée' in linear order, and the *inner form*, 'ordre structurale', which contains a net of relations conveying the grammatical relations in the sentence on an abstract level independent of the linear precedence in the surface text [ibid.: 19]. He stresses that the domain of syntax is to describe the *inner form*, i.e. the structural order, while he wants to delegate the *outer form*, i.e. word order to morphology and phonology. In dependency theory, word order plays no primary role (it may help as a secondary role to disambiguate on the outer form if necessary), but it is not conserved in the inner form.

For languages with freer word-orders than English, such a suggestion seems promising indeed and more suitable and 'modern' than GB descriptions, in which complex topicalisation movements have to take account for different word orders. I have described such a movement system for a subset of German, Danish and English in an earlier paper and implemented it in Lexical Functional Grammar (LFG) [Schneider 1996]. In LFG terms, put very loosely, Tesnière's suggestion allows to parse surface text directly for f-structures, i.e. functional relations, without the need for transformations. We will come back to f-structures in 2.3.8.2

Dependency forms only a part (although the core) of Tesnière's linguistic theory. In addition to dependency, which Tesnière calls *connection*, his theory knows two more types of syntactic relations, *junction*, and *translation*.

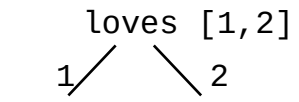
2.1.1 Connection

Connection, which corresponds to dependency in dependency grammar, is the most basic relation between words [Weber 1997: 21 ff.]. The simple sentence (1) *Peter sleeps* consists of the elements (i) *Peter*, (ii) *sleeps* and (iii) the connection between them:



(1) shows one common way to express dependency. The arrow points from he head to the dependent. (The direction of dependencies will be discussed in 2.3.2)

Tesnière favours to represent his syntactic relations using another common way to express dependency, so-called *stemmas,* which visualise dependencies (introduced in 2.1.1.2) and are reminiscent of constituent analyses. (The numbers in square brackets show the verb's subcategorisation frame). (2) is a representation of *Peter loves Mary*:



(2)

In this representation, the head is placed above its dependent(s). The number in square brackets refer to the number of dependents, or *arguments* in a logical representation.

2.1.1.1 Syntax and Semantics

It is not a coincidence, anyway, that the above stemma structure is so close to a logical, semantic representation (3):

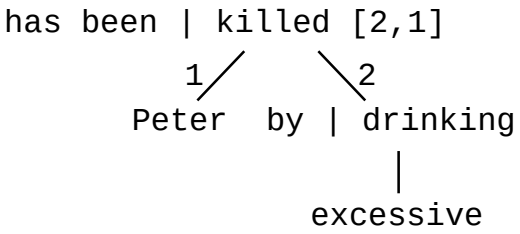
```
( 3) loves(Peter, Mary).
```

Connection has a syntactic and a semantic component, which usually run in parallel. Without syntactic connection there is no semantic connection but not always vice versa (see below) [Weber 1997: 21]. Unlike in other dependency-based grammars such as Link Grammar [Sleator & Temperley 1991, 1993] or Word Grammar [Hudson 1984, 1990] not every word is connected. *Function words* [Finegan 1989: 175] only signal grammatical relationship, they are only a part of a nucleus. Tesnière calls the *content words*, which have connections to other words, *nuclei* ('nœud'). For Tesnière, not words (as in Link Grammar or Word Grammar), neither constituents as in GB, but the nuclei are the basic elements of his theory [Weber 1997: 21]. As we will see in chapters 2 and 6, this means that Tesnière is much closer to semantics than many of his successors and more *functional* (cf. 2.3.2.5.6 and 2.3.2.7). The sentences

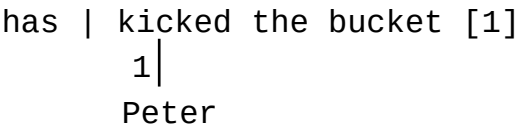
```
( 4) Peter has been killed by excessive drinking.
```

and (5) Peter has kicked the bucket.

show how e.g. passivisation, PPs, auxiliaries and idioms can be elegantly analyzed. The (functional) words before the vertical bar '|' are taken up into the nucleus of the content word or word group after the vertical bar:



(4)

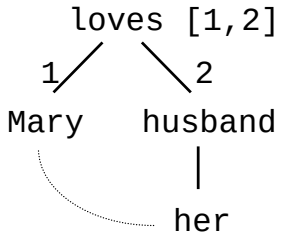


and (5)

Articles remain functional words in the text, according to Tesnière they only signal grammatical relation and are therefore a part of the nucleus. Articles and pronouns, however, clearly illustrate that the distinction between function and content words is a continuum:

There are a number of special word classes, in which syntax and semantics cannot run in parallel. They are mainly *deictic* categories like anaphora [ibid.: 23,44] and *reference* articles [ibid.: 24]. Reference and deixis are of course related semantic concepts, both dealing with real-world identification of linguistic constructs.

Anaphora are function words in the lexicon on the one hand, on the other hand they are also referent to other nuclei and therefore have to be nuclei themselves in the text . The stemma representation of e.g. (6) 'Mary loves her husband' is



(6)

As we will see in 2.3.2.5.4, it is also a serious problem to decide if a determiner is a head to its noun or vice versa. Analyzing it as a functional component of its noun is a possible solution, or at least it elegantly permits to leave the dependency direction problem underspecified.

The relationship between syntax, grammatical functions and semantics may not be as simple as Tesnière liked to present it. He has often been criticized for mixing up these three distinct levels; e.g. Helbig [1993] points out that

- on the one hand, Tesnière postulates that syntax and semantics be completely independent, on the other hand he thinks they usually run in parallel. As seen above, he recognizes that anaphora are semantic

connections without syntactic realization, that syntactic and semantic dependency often run in opposite directions, and that there are dissociations between them (e.g. the auxiliary verb carries the syntactical function, but the infinite verb the semantic function), but for him this does not suffice to indicate that the postulate of their being parallel needs revision. [ibid.: 43]

- the definition of word classes in Tesnière is partly based on semantic and partly on functional considerations. While nouns describe semantic objects and adjectives describe semantic attributes, *nobody* is defined as a noun for functional reasons, or *in the mountains* is an adverb because it is subject to a translation (cf. 2.1.4) [ibid.: 44]

2.1.1.2 Dependency

Connections between nuclei are *directed*, i.e. one participant of the connection depends on the other. Except in verbless clauses, the root dependence is from the verb. Weber writes that, unlike all other dependencies, it is difficult to prove this root dependency.:

> Damit haben wir die **Basis-Dependenz** in der DG [=Dependenzgrammatik] eingeführt. Sie ist empirisch nicht einfach zu begründen, sondern muss als Setzung betrachtet werden. Die Abhängigkeitsrelationen zwischen den anderen Wortklassen können experimentell überprüft werden.
>
> (Weber 1997: 22)

It is possible to suggest that *PRO-drop languages* and *expletive subjects*, according to the above elimination test, predict verb supremacy anyway. But verbless sentences in which the subject functions as root-head are also very frequent. The main predicate becomes the root head, then, e.g. *sentence* in (7):

(7) What a lousy example sentence !

The philosophical question whether the subject or the main verb should be the semantically most important element and therefore the root is a difficult, perhaps undecidable problem. While Tesnière and recent GB analysis (which places the subject within the verb phrase, cf. 2.3.9) regard the verb as root, Link Grammar (cf. chapter 5) and possibly older versions of GB (in which VP is a part of the subject NP) often take the Subject as the root of a sentence.

All the other dependencies can be more easily identified by an *elimination test*: If eliminating a nucleus leads to an ungrammatical sentence, then other nuclei depend on this nucleus:

> Akzeptiert man die grundlegende Abhängigkeit zwischen dem Verb einerseits und seinen Aktanten bzw. Circumstanten andererseits, die Tesnière postulirert, dann können Abhängigkeiten zwischen den übrigen knoten relativ leicht identifiziert werden ... Zur ‹berprüfung hat sich "unter Dependenz-Grammatikern" ein Test als nützlich erwiesen: Er besagt, dass ein Knoten nicht eliminiert werden kann, wenn noch andere Knoten von ihm abhängen.
>
> (Weber 1997: 45)

Chapter 2.3.2 will discuss dependency and the suggested elimination test more closely. Let us now explain the terms "Aktanten" and "Circumstanten" from the above quotation:

2.1.2 Complements (*actants*) and adjuncts (*circonstants*)

Tesnière distinguishes between elements that are subcategorized for, so-called *actants*, i.e. arguments filling valencies in predicates, and adjuncts or modifiers that can be attached quite freely, so-called *circonstants*. These terms are central in *valency theory*, which was originated by Tesnière, but developed by many others (cf. 2.3.1 and 2.2.3.1). Actants are usually compulsory, they are always semantically present, but under certain conditions they may be syntactically absent, e.g. the actor in passive sentences, the subject in pro-drop languages etc.

> Tesnière war jedoch mitnichten der Ansicht, dass ein Verb immer mit allen seinen Aktanten vorkommen müsse; vielmehr baut er seine Valenzmetapher weiter aus, indem er zwischen gesättigten und ungesättigten Valenzstellen unterscheidet:
>
> > "Notons d'ailleurs qu'il n'est jamais nécessaire que les valences d'un verbe soient toutes pourvues de leur actant et que le verbe soit, pour ainsi dire, saturé. Certaines valences peuvent rester inemployées ou libres"
>
> [(Tesnière 1959: 238)]
>
> (Storrer 1996: 225)

Tesnière himself does not elaborate a theory addressing the question when actants can be absent. See [Storrer 1996] for such a theory or [Feuillet 1996] for a criticism of Tesnière's distinction, which is considered too simple there.

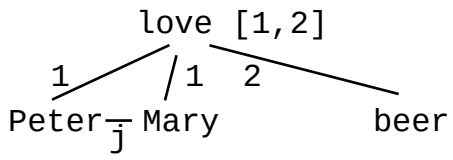
2.1.3 Junction

Junction is employed to relate elements on the same level, i.e. non-dependently [Weber 1997: 55ff]. Junction is mainly used for non-subordinating conjunction, which poses a major problem to dependency. In a sentence like

(8) Peter and Mary love beer.

it is hard to think of *Mary* to depend on *Peter*, or the vice versa, neither semantically nor syntactically. Thus, they need to appear at the same level (*j* stands for junction):

(8)



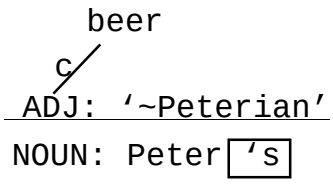
For current dependency theories, coordination remains a very serious problem. In the debate between advocates of dependency and those of constituency in 2.2.6.1 we will see that coordination is the one and only phenomenon where even hard-line

dependency linguists are forced to use some kind of constituency or a non-dependent relation.

2.1.4 Translation

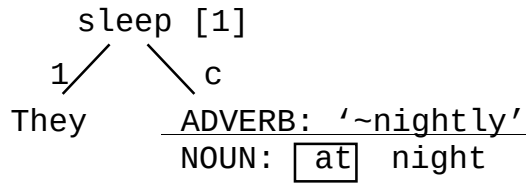
Translation is used to allow words to appear in syntactosemantic positions and functions usually occupied by words of other word-classes [Weber 1997: 77ff]. For a detailed introduction to Tesnière's translation, refer to [Werner 1993].

A noun-premodifying genitive can e.g. be conceived of as occupying an adjective position and function, as in (9) *Peter's beer*. The bar symbolizes the translation, the squared element is the so-called *translative* which triggers the translation, and the quoted element is a virtual pseudo-natural linguistic element I have allowed myself to insert for explanatory purposes:



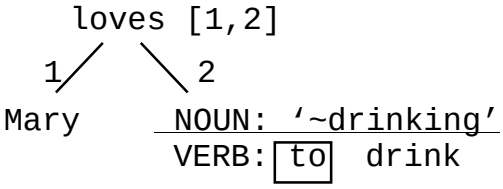
(9)

Similarly, in (10) *They sleep at night*, the preposition *at* triggers translation to adverbial:



(10)

The noun-like character of English gerunds also lends itself to be dealt with by translation, as in (11) *Mary loves to drink*.:



(11)

Such translations are reminiscent of LFG lexical rules (cf. 2.3.8.2), and especially the nominalization translation in (11) seems to have been taken up by [Chomsky 1965]'s nominalization transformation, one of his most debated transformations:

> Clearly, the words destruction, refusal etc., will not be entered into the lexicon as such. Rather, destroy and refuse will be entered into the lexicon ... [and] a nominalization transformation will apply at the appropriate stage in the derivation.
>
> (Chomsky 1965)

Today, the immense problems such transformations pose are well-known. While they seem linguistically elegant, they massively overgenerate when applied blindly and automatically. Derivations are never fully productive, and semantic shifts

frequently occur. The effects of the complex interplay between these translations or transformations are also hard to predict.

But is this the only possible answer to Tesnière's "théorie de la translation, sans doute la partie la plus contestable de l'œuvre de Tesnière" (Feuillet 1996: 130) ? Because translations are not part of most modern dependency theories, and because we will only refer to them cursorily in the rest of this paper, we may well take a closer look at them here.

To begin with, we have to remember that Tesnière's theory always intended to analyze only, therefore overgeneration is a lesser problem: "in der Tat war Tesnières Anliegen primär auf die Analyse sprachlicher Äusserungen, auf Analysemethoden, nicht auf die Synthese und folglich auch nicht auf eine Erzeugungsgrammatik ausgerichtet" (Helbig 1996: 41).

Tesnière has been criticized for confounding categories (in the word-class sense), grammatical relational functions and grammatical case (cf. 2.3.8) in his translation.

> La théorie de la translation, ... , fait apparaître une confusion très grave entre catégories (dans le sens de «types de mots») et fonctions. En ce qui concerne les actants, Tesnière renouvelle la même erreur en assimilant cas et fonction ...
>
> (Feuillet 1996: 130)

2.1.4.1 A Functional Concept of Word Classes

Addressing these criticisms, we can first note that our conception of word-classes is too much form-based (cf. [Weber 1996: 250]). While e.g. the italicised part of (12b,c) are *grammatically* not objects, they *functionally* serve the same purpose. Translations allow us to take a more open and functional perspective.

(12a) She knows *the art of singing*.
(12b) She knows *how to sing*.
(12c) She knows *that she can sing*.

Because Tesnière is not interested in generation, we do not have to worry if all the sentences such a conception allows are grammatical, but we are able to easily and economically (parsing-speed!) recognize all the correct ones and subsume them to the same class, This does not mean that we could not divide them into additional subclasses later on if needed. *(b) how to sing*, e.g., is rarely acceptable because it depends on the verb semantics, which means that the sentences can be rejected at a later stage, if necessary.

(13a) She loves the art of singing
(13b)* She loves how to sing.
(13c) She loves that she can sing.
(13d) She loves to sing.
(13e) She loves singing.

or

(14a) She approves of the art of singing.
(14b)* She approves of how to sing.
(14c)? She approves of that she can sing.
(14d)* She approves of to sing.
(14e) She approves of singing.

or perhaps even

(15a) She wants the art of singing.
(15b)* She wants how to sing.
(15c)? She wants that she can sing.
(15d) She wants to sing.
(15e)? She wants singing.

Accepting ungrammatical sentences has the clear advantage for computational analyzers that we easily understand them and that we do not have to worry about unnecessary standards of acceptability, which are undefinable anyway. If two related constructions have similar semantics we may freely choose whether we really need a model that distinguishes between them, and whether we really need a model that pays respect to historical arbitrary decisions to favour one form over the other. There is e.g. absolutely no need for a semantically oriented parser to reject * *I like sing* or * *I can singing.* While this distinction is important for a grammar-checker or a generation system, an efficient text-understanding or information-retrieval system completely fails its goals, loses a lot of precious parsing time and requires an unnecessary complex grammar if it tries to be a pedantic language teacher. Here, a *functionally* based parser makes much more sense than a *grammatical* word-class based parser. Grammatical features should then only be taken into consideration as far as they can reduce ambiguity.

Weber [1996] portrays translation as an important instrument of functionalism:

> Mit Einführung der Translation ist es Tesnière gelungen, die Satzstrukturbeschreibung einerseits auf wenige Konnexions-Relationen zu beschränken und sie andererseits offenzuhalten, und zwar sowohl im Hinblick auf ihre hierarchische Gliederung als auch in Hinblick auf ihre Variabilität ... Gegenüber den Konnexionen und der Valenz, die als weitgehend konstante

> Elemente der Satzstrukturbeschreibung anzusehen sind, bildet die Translation ein Element der Offenheit ... Von Tesnière selbst wurde die Translation als unverzichtbarer Teil seines Entwurfs angesehen. Als vorteilhaft hebt er hervor, dass die Ausbaumöglichkeiten des einfachen Satzes unter einer Rubrik beschrieben werden und dass dabei syntaktische Aspekte im Vordergrund stehen und weniger die morphologischen Eigenschaften einzelner Wortarten oder ihre Position in der Chaîne parlée.
>
> (Weber 1996: 249)

2.1.4.2 Open Valencies

Addressing the criticisms raised against translations (cf. [Feuillet 1996] above) , Weber [1996:250-1] explicates that the impression of confounded word-classes also partly stems from our interpretations of their usual stemma notation. E.g. in (10) above the noun literally seems to be translated to an adverb(ial). But the apparent change of category is only a metaphor. The connection potential, the characteristics of the element under transformation as a head to its dependents is not affected. Weber [ibid.] stresses that translation rather enables the element under translation to fill a valency it otherwise could not. Translation is therefore a concept to extend valency classes.

> Die Wortarten Tesnières haben einen Doppelcharakter als Form- und Funktionsklassen, auch im Rahmen der Translation. Dabei kann die Translation Auswirkungen auf die Form, also auf das "Aussehen", und das Stellungsverhalten der entsprechenden Wörter haben. Das Konnexionspotential eines Wortes wird jedoch nicht ausgetauscht. Denn bei Translation liegt strenggenommen keine Vermischung von Kategorien und Funktionen vor, auch kein Kategorien-*Wechsel* (...), sondern eine Anreicherung ... Es reicht demnach nicht aus, von einer Translation zu sprechen, die ein Substantiv [cf. (10)] in ein Adverb oder die die Wortart Substantiv in die Funktion eines Adverbs *überführt*. Angemessener ist die Sehweise, dass ein Knoten sowohl für einen substantiv-spezifischen als auch für einen adverb-spezifischen Strukturzusammenhang kompatibel gemacht wird. Diese (Form- und Funktions-) Anreicherung ist erforderlich, um auch komplexe sprachliche Konstruktionen – auf der Grundlage bestehender Konnexionsregeln – in den Satz einzufügen, ist damit eine wesentliche Vorbedingung für die strukturelle Offenheit des Satzes.
>
> (Weber 1996: 250-1)

2.1.4.3 Parsing Efficiency

If we really treated translations as transformations, in the sense that one word form is transformed into another, they would be as very time-intense and unwieldy to parse as Chomskyan transformations: We only know the "surface" element after the transformation (or translation: I will only use the term transformation here) from the *chaîne parlée* (which Chomsky calls numeration) and have to find a corresponding "deep" structure before the translation or transformation. We have to perform a "backwards" transformation. Because of the complexity of transformations, because TG and GB are generative we are forced to more or less blindly generate zillions of surface structures and see if any of them fits the input string. This is a major reason why Chomskyan grammars are unparsable in practical terms.

If we think of translations along the same lines we are also forced to “backward” generate, to more or less blindly transform possible initial candidates according to the established transformation rules until one of them happens to match, which makes parsing forbiddingly inefficient. If we assume Weber’s [1996] above conception, however, i.e. that translations extend the valencies to admit more classes, parsing remains an efficient “forward” activity. Weber’s clarifications are more than welcome to permit efficient and fast parsing.

While translations may seriously overgenerate, when used to analyze they permit to efficiently parse for very compact structures that cover a big variety of semantically related structures with the same syntactic functions. While they may blur semantic details, they are an ideal tool for semantic shallow parsing as aimed for in e.g. information retrieval.

2.1.5 Conclusions

Tesnière's work is only about to be rediscovered again, as e.g. Gréciano [1996] shows. While the Chomskyan theories drown in their increasing complexity, while computational unification-based constituent theories still suffer from very serious efficiency problems, alternatives are sought after. Dependency grammars could be such an alternative.

Research on valency (cf. 2.3.1 and 2.2.3.1) has continued, but especially Tesnière's translations have been almost neglected. Although the valency aspect of Tesnière's dependency has attracted many researchers (e.g. the German School, cf. 2.2.3), many aspects of dependency have not been developed much further. Dependency cannot be reduced to valency only. "Dies ist eine unzulässige Reduktion des dependenzgrammatischen Prinzips auf einen, wenn auch wichtigen, syntaktischen Ausschnitt." (Eroms 1987: 80). "Gegenüber der raschen Entwicklung der Valenzlehre wurde die Forschung auf dem Gebiet des Dependenzprinzips jedoch in den letzten Jehren ziemlich vernachlässigt." (Jung 1995: 11).

[Sleator & Temperley 1991, 1993] do not address Tesnière's theories, nor the differences between their approaches, perhaps because Link Grammar is too far away from the original Tesnière, but probably also due to the facts explained in [Weber 1997]:

> Tesnières Motive lagen nicht darin, eine formale Grammatiktheorie zu stiften. Obwohl in Tesnières Hauptwerk, den "Eléments", häufig auf Verhältnisse in unterschiedlichen (meist indoeuropäischen) Sprachen eingegangen wird, kann man es auch nicht als eine umfassende Sprachtheorie bezeichnen. Vielmehr ist den "Eléments" eine didaktische Grundhaltung eigen. ... gerade in der Praxis des **Sprachunterrichts** hat sich die Tesnièresche Methode der syntaktischen Beschreibung früh bewährt.
>
> (Weber 1997: 11)

2.2 Advocates of Dependency after Tesnière

Hays [1964] and Gaifman [1965] were the first to give dependency a rule formalism, Mel'čuk [1988] was the first to introduce the large Russian dependency tradition in America and is still one of the basic textbooks on dependency. In (partly the former Eastern) Germany, people like Helbig [1988, 1992] or Engel [1994,1996] continued and extended the Russian tradition and Tesnière's work. Fraser [1996] is one of the most prominent representatives of dependency theory today. The list of names included in the following subchapters is by no means complete.

2.2.1 Hays and Gaifman

"A number of different dependency rule formalisms have been developed and described in the literature. The oldest and most widely used of these is due to Gaifman [1965]" (Fraser 1996: 72). "Gaifman [1965] was the first to actually give a formal method of expressing a dependency grammar. He shows that his model is context-free." (Sleator & Temperley 1993: 12). "The earliest formalisations of dependency grammar are often attributed to Gaifman and Hays [1964]" (Järvinen & Tapanainen 1997: 3).

Järvinen & Tapanainen [ibid.] stress, however, that only the model used by Gaifman and Hays is context-free, and that this does not apply for all formalisms. E.g. their non-projective dependency parser is context-sensitive.

> Unfortunately, those formalisations were basically constituent grammars where the head of the constituent is marked. Although Gaifman and Hays studied only properties of one given formalisation and showed that it is "weakly equivalent" to the constituent grammars, many linguists have taken their formalisation for granted.
>
> (Järvinen & Tapanainen 1997: 3)

We will come back to this point in chapter 2.4.

2.2.2 Mel'čuk and the Russian School

> The view that dependency is the basis of most of syntax is commonplace in the Slavic-speaking world, and this tradition is now easily accessible to those of us who do not read Russian, thanks to Mel'čuk and his colleagues in Moscow and Montreal ...
>
> (Hudson 1990: 107)

While Gaifman wanted to formally define dependency, Mel'čuk rather conceives of dependency as a tool. According to Sleator & Temperley [1993], Mel'čuk [1998] offers a formal definition of dependency structures: "This structure, as defined by Mel'čuk [1988], consists of a set of planar directed arcs among the words that form a tree. Each word (except for the *root word*) has an arc out to exactly one other word, and no arc may pass over the root word." (Sleator & Temperley 1993: 11-12). Checking in

Mel'čuk's book, however, reveals that Mel'čuk does not intend to give a proper definition:

> All I intend to do is to suggest an artificial FORMAL LANGUAGE, or a formalism, for describing natural sentences at the syntactic level. ... By its logical nature, dependency formalism cannot be "proved" or "falsified." Leaving aside simple errors and inconsistencies, it can be evaluated solely in terms of expediency or naturalness, not in terms of truth or falsity. Dependency formalism is a tool proposed for representing linguistic reality, and, like any tool, it may not prove sufficiently useful, flexible or appropriate for the task for which it has been devised; but it cannot be true or false.
>
> (Mel'čuk 1988: 12)

Covington [1992] stresses this argument: "As Mel'čuk (1988) has emphasized, dependency grammar is not a **theory** of language, but rather a **notation** for describing structure. Theories of grammar can of course be built upon it, as Mel'čuk, Tesnière, Hudson, Starosta, and others have done." (Covington 1992: 2)

2.2.2.1 Types of Dependency

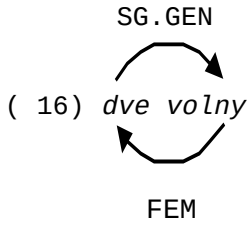
But although dependency is a fairly intuitive concept, the picture becomes much more complex when looking at the gory details. Mel'čuk [1988:106-149] distinguishes three kinds of dependency: *morphological*, *syntactical* and *semantic*. He mentions that possibly more dependency types could be recognized, e.g. anaphoric links.

2.2.2.1.1 Morphological Dependencies

When a word form determines the word form of another word, such as in agreement, we speak of morphological dependency. Mel'čuk [ibid.: 108-112] lines out three main properties of morphological dependency.

First, "all languages have words that are morphologically invariable and that because of this invariability, are never morphologically dependent on another word." (ibid.: 108). Therefore, morphological dependencies do not create a *connected* structure for a given sentence, there are many discontinuities in the chain of morphological dependencies, for every language. Morphological dependency can thus not be a main criterion in the definition of dependency: "All this suggests that ... morphological dependency is a marginal type of syntagmatic dependency".

Second, "a morphological dependency can be bilateral" (ibid.: 109), which means that the word forms of two words can mutually depend on each other. Mel'čuk [ibid.: 109] gives a Russian (16) and a Georgian ergative (17) example [ibid.: 109-10]:



In (16) 'dve volny' (='two waves'), the numeral *dv+e* is morphologically dependent on the noun *volny* according to gender ('volna'='wave' is feminine), but *voln+y* is dependent on *dve* according to number (sg.) and case (genitive).

(17a) singular subject:

Is *amb + ob + s* *rom* ...
he-SG.NOM say PRES 3SG that
'He says that ...'

versus

Man *tkv + a* *rom* ...
he-SG.ERG say AOR.3SG that
'He said that ...'

or

Mas *u + tkv + am + s* *rom* ...
he-SG.DAT say PERF 3SG that
'He has said that ...'

(b) plural subject:

Isini *amb + ob + en* *rom* ...
they-PL.NOM say PRES 3PL that
'They say that ...'

versus

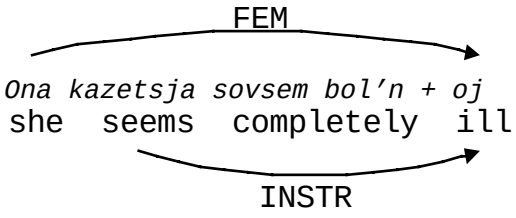
Mat *tkv + es* *rom* ...
they-PL.ERG say AOR.3PL that
'They said that ...'

or

Mat *u + tkv + am + (s)t* *rom* ...
they-PL.DAT say PERF 3PL that
'They have said that ...'

As usually in split-ergative languages, the grammatical case of the subject is determined by the tense of the main verb (*nominative* for the present, *ergative* for the aorist and *dative* for the perfect). At the same time, the form of the verb depends on the number of the subject (*singular* in (17a) and *plural* in (17b)).

Third, "since a word can have several morphological variables, it can be morphologically dependent on several word forms at once." [ibid.: 111]. Mel'čuk gives a Russian example (18) :



(18)

The adjective *bol'n+oj* ('ill') depends in gender and number on *Ona* ('she'), but at the same time it receives instrumental case from *kazetsja* ('to seem').

2.2.2.1.2 Semantic Dependencies

In a predicate-argument structure, which is a common form for expressing semantic structure, the arguments depend on the predicate. Semantic and e.g. morphological dependency do not necessarily go into the same direction. Articles are a good example. A noun morphologically determines the number, gender and case of its article, but semantically an article determines its noun - as the grammatical term *determiner* expresses: the article is a one-place predicate that attributes the semantic property of being definite or non-definite to its noun argument. Mel'čuk [ibid.: 116-118] notes the following characteristics of semantic properties:

First, unlike morphological dependency, semantic dependency is *universal*, which means that it applies to all words in a given sentence, except for some syntactical markers like e.g. adverbial markers (which are part of the verb semantics) or some prepositions (which clarify the semantic role and argument order).

Second, again unlike morphological dependency, semantic dependency is *unilateral*, which means that there are no mutual dependencies.

Third, like morphological dependency, a term can have several governors, i.e. it can be the argument of several predicates. In a prepositional phrase like 'on the screen', *screen* simultaneously depends on the preposition *on* and on the article *the*.

2.2.2.1.3 Syntactic Dependencies

When introducing the distinction between morphological, syntactical and semantic dependency, Mel'čuk [ibid.: 106] calls morphological dependency a strictly formal kind of dependency, and semantic a strictly conceptual dependency. Syntactic dependency is at the uneasy and unclear position between these two extremes, half formal and half conceptual. Yet this intermediate position is necessary. From the one extreme, we cannot extend morphological dependency to replace it completely, because as mentioned above it does not form connected structures and many

languages even lack it completely. From the other extreme, as mentioned, also semantic dependency leaves some words unconnected.

Word form A syntactically depends on word form B if, according to Mel'čuk [ibid.: 113], the syntactic role of the combination of A+B has usually the syntactic role of B when it occurs alone. Syntactic roles, however, are themselves defined by the type of dependency, e.g. on what word C this word form D depends, and in which way. Because the definition of syntactic dependency is therefore itself based on dependency, we seem to run into a vicious circle, as Mel'čuk himself stresses [ibid.: 115]. This vicious circle can only be overcome if one defines a top-head, a unit which only has dependents but cannot depend on anything. As usual in dependency grammar, Mel'čuk defines the finite verb to be the top-head of a sentence. Once this postulation has been accepted, the dependencies in a sentence can be recursively found out by stepping down from dependency to dependency.

About the characteristics of syntactic dependency, Mel'čuk [116] notes:

First, like semantic dependency, syntactical dependency is universal. By definition, syntactic dependencies link all word forms, because sentences with unlinked elements are supposed to be ungrammatical.

Second, Mel'čuk defines dependency not to be bilateral. There can be no mutual links if one wants to represent dependency structures with trees or Tesnière's stemmas.

Third, in Mel'čuk's theory, any word form can syntactically depend on only one other word form. Cf. Word Grammar, 4.2.1, for a different assumption.

In its awkward position between formal morphology and conceptual semantics, Mel'čuk is "unable to propose a rigorous definition of syntactic dependency" (ibid.:129), but he attempts to get a clearer picture. In order to define dependency, we need three criteria:

- **connectedness** criteria, establishing if two word forms are connected by a dependency relation, i.e. if w_1—w_2 or *w_1—w_2.
- **direction** criteria, establishing which of the connected word forms is dependent and which is governor, i.e. given that w_1—w_2, if w_1—>w_2 or w_1<—w_2.
- **dependency type** criteria, establishing the type of dependency. Note that only some dependency theories group the dependencies into different types.

2.2.2.1.3.1 Connectedness Criteria

In order to find out if two word forms in a given sentence are connected, Mel'čuk [ibid.:130-2] suggests two criteria: *linear correspondence and prosodic correspondence.*

Two word forms have **linear correspondence** if the linear position of one cannot be determined without reference to the other. Let us recall that for Tesnière dependency explicitly does not want to express linear oreder. This criterion does therefore not mean that one will always follow or precede the other, but that in order to describe the position of one in a sentence it is necessary to mention the other. In order to describe the position of a preposition, e.g., it is necessary to mention that it occurs before the noun phrase to which it is linked.

Although linear order is not primarily relevant for dependency (cf. 2.1), referring to linear order turns out to be one of its key definitions for Mel'čuk. This means that in an indirect way, word order is nevertheless central, and that because linear order is a constituency relation, this dependency criterion inherently builds on a constituency concept. It is to be suspected, however, that this criterion does not hold in languages with completely free word order and non-projectivity (cf. 2.4.7). On the other hand, freeness of word order and non-projectivity are limited in all languages, one may even want to argue that this dependency criterion is an argument for this limitation, because otherwise human parsers would no longer be able to understand the dependencies in a sentence.

Two word forms have **prosodic correspondence** if they can form a prosodic unit together, or if they can form a prosodic unit together with the dependents of one of them. Since dependency structures are built up iteratively, it is possible to refer to the dependents of a word form at this stage, i.e. when the governor or additional dependents are not yet recognised. Mel'čuk deliberately leaves the concept of *prosodic unit* undefined, but he adds that "our prosodic unit is (roughly) what is currently called a **phrase** or **constituent**" (ibid.: 131).

Again, dependency explicitly refers to a constituency concept for its definition. Surprisingly enough, Mel'čuk does not mention *valency* (cf. 2.3.1 and 2.2.3.1) as a defining factor, but builds on constituency which he introduces through the term *prosodic unit*. By basing his definition on constituents, Mel'čuk's definition builds up on the results of a linkage, while valency would start with the lexical need of words to link, which would allow a definition that does not have to resort to constituency.

Perhaps he does not want to introduce semantic arguments (e.g. subcategorisation), which depend on the particular lexical item.

2.2.2.1.3.2 Dependency Direction Criteria

In order to establish the dependency direction, Mel'čuk [ibid.: 132-138] suggests two criteria: *syntactic role* and *morphological contact point*.

The **syntactic role** criterion is a more elaborate version of the definition stated above: Word form A syntactically depends on word form B if the syntactic role of the combination of A+B has the syntactic role of B when it occurs alone, or B at least determines the syntactic role of A+B to a greater extent. Mel'čuk himself [ibid.:132] calls this criterion *imposition of passive SS-valencies*, but he basically means syntactic roles by that [ibid.:112].

Morphological contact point is a deliberate reference to another type of dependency, morphology, confirming that these types are not fully autonomous. The head of two linked word forms is the word form whose morphological links to other word forms external to this link are more important. This is basically the head-criterion used by Generalized Phrase-Structure Grammar (GPSG) or Head-Driven Phrase-Structure Grammar (HPSG) [Pollard & Sag 1994], in which usually only head-features are passed on to the subordinate syntactic node, according to the *Head Feature Principle* (HFP) (cf. 0).

Mel'čuk [ibid.: 139-4] mentions two additional and frequently used criteria, which in his opinion fail too often to be included:

The **omissibility criterion** says that a dependent can usually be omitted to produce a syntactically correct sentence with similar meaning, while the head cannot. This criterion is e.g. suggested as a useful test in Weber [1997: 45].

The **predictability criterion** says that the dependent is more likely to predict the word class of its head than vice versa. This is true for e.g. adjectives, which usually depend on nouns, but in e.g. a conjunction + verb construction it is rather the conjunction as head, which predicts the dependent verb.

2.2.2.1.3.3 Dependency Type Criteria

If one employs labelled dependencies, as many dependency theories do, it is necessary to decide which dependencies belong to the same type, i.e. receive the same label. I only want to give a very short overview of Mel'čuk's criteria here [ibid.: 140-4].

- **semantic contrast in minimal pairs**: Two otherwise identical sentences have semantically different meaning if one of their dependency types is different. E.g. In German (my own examples) (19a+b) constitute such examples. In (a),

the verb-object dependency is of syntactic type ACCUSATIVE or of deep Case PATIENT, in (b) the dependency is of syntactic type DATIVE or deep Case BENEFACTOR. The change in semantics is so fundamental that the verb has a different lexical meaning in the two cases. In (a), it means *distribute,* in (b) it means *forgive*.

(19a) Ich vergebe ihn.
(b) Ich vergebe ihm.

- **reciprocal substitutability of trees**: this argument is well-known form constituency. Phrases of the same class can mutually substitute each other, analogically elements introduced by the same dependency type do.
- **repeatability of dependency**: If two dependencies from a governor are of the same type, then there may also be more of the same type. An example for this are adjectives, which may be recursively attached to nouns.

2.2.2.2 Mel'čuk's Defense of Dependency

Mel'čuk's clear distinction between different levels of dependency, *morphological*, *syntactical* and *semantic*, allow him to confront common dependency criticisms. "In the literature one finds a number of unjustified criticisms leveled at dependency formalisms that claim its unsufficiency or inadequacy" (1988: 24). They are, according to Mel'čuk:

2.2.2.2.1 Double Dependency

In sentences like (20)

(20) Wash the dish **clean**

the predicative adjective *clean* can be said to be dependent on both the verb and the noun. Mel'čuk [ibid.:25] argues that the adjective is syntactically dependent on the verb only, despite the semantic relation to the noun. In languages like Russian, French, Italian, Danish or Swedish predicative adjectives or past participles also morphologically depend on the noun by agreement, but this does not affect the syntactic dependency, as also [Tarvainen 1981: 11] stresses.

(21) *Les tâches* sont achevé*es* - *L'exercise* est achevé_ (French)

(22) *Jakken* er rød_ - *Huset* er rød*t* (Danish)

In Italian there can even be such a morphological dependency between object and verb in impersonal constructions, probably due to their proximity to passive constructions:

(23) Si sono comprat*i due libri* - Si è comprat*o un libro* (Ital.)

The confusion between the different layers partly comes from Tesnière's conception, because he tries to treat syntax and semantics as equal and does not recognise a clear distinction between them (cf. 2.2.3.1.4) or between word-classes and grammatical functions (2.1.4). Hudson [1980a] still perceives of constructions like (20) as a major problem for dependency: "The main outstanding problem for the definition of dependency ... is that certain facts require the verb and the predicative adjective to be treated as modifiers of the subject, even if other facts require the reverse relation" (Hudson 1980a: 190).

2.2.2.2.2 Mutual Dependency

Many linguists, e.g. Hudson [1980a: 185-7] point out that the direction of the dependency is often unclear. (cf. chapter 2.3.2). For Mel'čuk [1988: 26] this is only one more confusion between syntactic and morphological dependency. E.g. the main verb and the grammatical subject can be said to mutually depend on each other. But in Mel'čuk's framework, only the main verb is the root governor, on which the subject syntactically depends. The dependency of the main verb on the subject is purely morphological.

2.2.2.2.3 No Dependency: Coordination

Mel'čuk does not even fully accept coordination as an argument for the necessity of constituency. He argues that coordinations expressing temporal sequence or causal relations are dependency relations [ibid.:26-7]. He admits, however, that dependency is insufficient in other cases of conjunctions [ibid.:28-33].

2.2.2.3 Conclusion

Mel'čuk carefully distinguishes morphological, semantic, and syntactical dependency, outlines their characteristics and shows that they cannot be mapped in a 1:1 relation onto each other. This distinction allows him to solve a number of problems and address criticisms raised against dependency. He is, however, admittedly unable to give a rigorous definition of dependency, and he leaves the dependency problem of coordinations unsolved. He presents syntactical dependency rather as a tool for analysis, as a working hypothesis with no ontological claims rather than the last wisdom.

2.2.3 The German School

> Die Konzeption der Valenz, die Tesnière aus der Chemie entlehnt hat, ist nach der Veröffentlicheung von Tesnières Arbeit im deutschsprachigen Raum auf grosse Aufnahmebereitschaft gestossen ... Sie hat eine ganze Generation von Lexikographen fasziniert.
>
> (Weber 1997: 34)

Järvinen and Tapanainen [1997:3] write that "much of the linguistic work within applied dependency theory, especially the development of valency theory, has been done in Germany." Among the names he mentions are Engel and Helbig, whose contributions to [Gréciano 1996] we have already met in 2.1. I would like to introduce their ideas in the following subchapters.

2.2.3.1 Helbig

Helbig's research is centered on *valency* (cf. 2.3.1). As we will see, valency is central to dependency grammar. Originally reserved for verbs, valency is a term whose use has been extended to increasingly more phenomena, as Helbig [1992] summarises:

> In den letzten Jahren sind in der Entwicklung der Valenztheorie zwei Grundtendenzen nicht zu verkennen:
>
> (a) die Ausweitung des Valenzbegriffs von der syntaktischen auf die semantische und schliesslich auf die pragmatische Ebene (...);
>
> (b) die Ausweitung des Valenzbegriffs vom Verb auch auf andere Wortklassen
>
> (Helbig 1992: 108)

Helbig [1992: 7-18] distinguishes several layers of valency and shows that there is not a simple 1:1 mapping between them.

> Obwohl die meisten Linguisten in der Annahme mehrerer Valenzebenen übereinstimmen – unabhängig davon, ob sie Termini wie "syntaktische Valenz" und "semantische Valenz" dafür benutzen –, bleiben auch gegenwärtig noch viele Fragen offen zu den Problemen, *was* unter der Valenz auf den Ebenen genau zu verstehen ist, *in welcher Weise* die die Valenz auf den verschiedenen Ebenen zu lokalisieren und *wie* der Zusammenhang (die Zuordnung) zwischen der Valenz auf den verschiedenen Ebenen zu beschreiben sei.
>
> (Helbig 1992: 7)

As a starting point it is common to distinguish between logical, syntactical and semantic valency [Helbig: ibid.]

2.2.3.1.1 Logical Valency

Logical valency is claimed to hold universally. This postulation is a typical case of an unprovable linguistic axiom we have discussed in 1.3, which we have to accept if we want to do semantics. It also constitues a good example to show that, according to Yngve [1996], linguistics is not a natural science.

The axiom we have to accept – or refute – here is that at some 'logical' level the predicates for a given action have the same number of arguments, the same *arity* across all languages. There has to be an internal language-independent concept for every speaker of any language in which e.g. to love has two arguments, or to sleep has only one:

```
( 24) sleep (x).
( 25) love (x,y).
```

On the one hand, such an axiom – like any unprovable axiom – is arbitrary and naïve, on the other hand most linguists tend to refute at least the strong version of the Sapir-Whorf hypothesis now [Fasold 1990:50-3]. Because there are grounds for believing that a native language does not strongly influence one's perception of the world, there are grounds for accepting this axiom, by the unavoidable lack of hard scientific evidence there is at least democratic evidence.

Helbig himself does not refer to Wittgenstein or arbitrary axioms, like many others he rather believes that the 'facts of reality' are somehow reflected in our consciousness:

> Die im Bewusstsein widerspiegelten Sachverhalte der Wirklichkeit sind formulierbar als Aussagestrukuren, d.h. als logische Prädikate ... Es hängt jeweils vom Begriffsinhalt des (logischen) Prädikats ab, ob es *ein* Argument (...) oder *mehrere* Argumente (...) hat.
>
> (Helbig 1992: 7)

2.2.3.1.2 Syntactical Valency

Unlike the ideally universal logical valency, syntactical valency is language-specific. Therefore syntactical and logical valency cannot be mapped isomorphically onto each other. Helbig [1992] first gives a German example,

> Dass diese verschiedenen Valenzebenen nicht identisch und auch nicht einfach isomorph aufeinender abbildbar sind, zeigen schon solche deutschen Verben wie *helfen* und *unterstützen*, die beide eine ähnliche logische Valenz haben (...R(x,y)...) und auch ähnliche Kontextpartner als Variable erfordern (AGENS, ADRESSAT), d.h. eine ähnliche (oder gleiche) semantische Valenz haben, die sich jedoch in der morphosyntaktischen Realisierung von y unterscheiden (bei *helfen* Sd [DATIVOBJEKT], bei *unterstützen* Sa [AKKUSATIVOBJEKT]).
>
> (Helbig 1992: 9)

then he shows different syntactic valencies across different languages with translations of the verb 'succeed':

([26] a) Es gelingt *ihm*, dass ...
(b) *He* succeeds in ...
(c) Il réussit à ...
(d) Udeatsja ...

2.2.3.1.3 Semantic Valency

Semantic valency is closely related to logical valency, some linguists use the term logico-semantic structure [ibid.: 10]. Like logical valency, it is often expressed in predicate-argument strucures. Semantic valency expresses the functional roles of the arguments, e.g. for the verb *give*:

(27) give(AGENT,PATIENT,GOAL) or
give(AGENT,PATIENT,BENEFACTOR) or even
give(AGENT,PATIENT,ADDRESSEE)

Helbig stresses that semantic valency only indirectly leads to a meaning structure [ibid.:10-11]. The terms in (27) do e.g. not express the change of POSSESSOR from the AGENT to the GOAL (or whatever it is called). Depending on the meaning model used, a big deal of abstraction and inference is still necessary.

A main influence to semantic valency theory was Fillmore's [1968] Case Grammar: "In der Tat ist die Valenztheorie - da die semantische Valenz ... zunehmend mit Hilfe der Inventarien der Kasustheorien beschrieben worden ist - eine Verbindung mit der Kasustheorie eingegangen."(Helbig 1992:18). Valency theory originates from Tesnière's dependency grammar, in its development several levels of valency have been recognised (as described above). Case Grammar has a very different origin: it is an answer to Chomsky's standard theory [Chomsky 1965]. Fillmore discovered that Chomsky's deep syntactic structure did not suffice to derive semantic structures, and he therefore developed a theory with a more sufficient deep structure - Case theory. Deep Case was planned to be uniform and universal, but this claim had to be relaxed later [Helbig 1992: 20]. The different Case names suggested in (27) e.g. indicate that there is no agreement about the names and number of cases. Today, the developments of Case theory are so fragmentary and differ from each other to such an extent that they meet more scepticism than approval. "Unbestritten ist auch, dass die Kasustheorie bei dieser Weiterentwicklung schon dem oberflächlichen Beobachter ein verwirrend uneinheitliches, nahezu desolates Bild bietet" (Helbig 1992: 26).

2.2.3.1.4 Extending Valency from Verbs to Other Word Classes

For verbs it was always most obvious that the choice of the number and the kind of complements they take is *lexicalised*. Even the earliest PSG variants recognised that it was not enough to offer a selection of alternative VP rewrite rules like:

```
( 28)  VP -> V           % intransitive
       VP -> V NP        % transitive
       VP -> V NP NP     % ditransitive
       VP -> V COMP      % subordinate clause
     etc.
```

It is necessary to group the verbs into different subclasses, to subcategorise them, to ensure that each gets the correct number and kind of arguments, e.g. by the use of argument features:

```
( 29)  VP[VTYPE:tr0] -> V[VTYPE:tr0]         % intransitive
       VP[VTYPE:tr1] -> V[VTYPE:tr1] NP      % transitive
       VP[VTYPE:tr2] -> V[VTYPE:tr2] NP NP   % ditransitive
       VP[VTYPE:comp]-> V[VTYPE:comp] COMP   % subordinate clause
     etc.
```

Predicative adjectives have in common with verbs that different subclasses of them take different complements,. usually certain PPs and complementizers. E.g. in (30), *afraid* requires a PP introduced by of, *ready* needs a PP introduced by *for*:

(30a) I am *afraid of* dogs.
(b) *I am *ready of* dogs.
(c) *I am *afraid for* action.
(d) I am *ready for* action.

Similarily for complementizers, in (31), *happy* needs e.g. a *that* complementizer while *curious* may take *whether*:

(31a) I am *happy that* you will come.
(b) *I am *curious that* you will come.
(c) *I am *happy whether* you will come.
(d) I am *curious whether* you will come.

Some adjectives need certain complements - analogous to transitive verb, others cannot take certain complements - analogous to intransitive verbs. In (32), *able* needs e.g. an *infinitival* complement, but *predicative* cannot take such a complement.

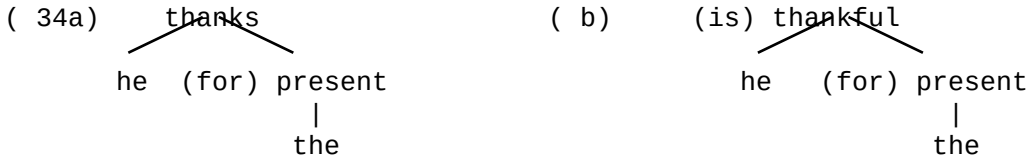
(32a) This adjective is *able to* modify nouns.
(b) *This adjective is *predicative to* modify nouns.
(c) *This adjective is *able.*
(d) This adjective is *predicative.*

In addition to these syntactical indications for adjective subcategorisation or valency, there is semantic evidence: Depending on its semantics, an adjective needs a certain number and kind of arguments, and accordingly opens up valencies for them. E.g. in (31), because one is *happy* about facts but *curious* about questions the adjective valency is semantic. Similarily, the adjective *similar* needs two arguments, otherwise their can be no comparison, or e.g. *guilty* needs at least intrinsically an actor and an action to be guilty of. For Helbig [1992: 109], adjectives have valencies like verbs, and these valencies are lexical.

In the discussion of *auxiliary verb+verb participle* constructions in 2.3.2.5.5 we will see that *functionally*, the auxiliary is only a marker on the verb and thus part of the verb nucleus in Tesnière's approach. There are very close parallels between predicative adjectives and verb participles (cf. Quirk et al. [1985: 413-6, § 7.15-19 and 1325-30, § 17.98-103]). In Tesnière's dependency grammar, copulae are a functional part of the predication nucleus. He analyses *copula+predicative adjective* and *auxiliary+participle* constructions in the same way: The whole construction forms only one nucleus. Weber [1997: 29-33] explains the serious problems this entails. Above all, attributive and predicative adjectives find a fundamentally different analysis, as Tesnière's examples show [Tesnière 1959: 181]:

Helbig [1992: 110-11] follows Mel'čuk's [1988: 26] argument that the discussion if predicative adjectives are part of the verb nucleus or not stems from a level confusion (cf. 2.2.2.2.1). Semantically, the predicative adjective is the head nucleus together with the copula (which he thinks should be cancelled at the sematic level), but syntactically, the copula is the head with the subject and the predicative adjective as arguments. We can see again that Tesnière did not distinguish between syntax and semantics, and that syntax and semantics do not always run in parallel, that Tesnière's aim to use near-semantic representations has to fail sometimes because semantics and syntax cannot be mapped 1:1 onto each other.

Helbig's suggested semantic dependency structure for predicative adjectives shows their close relation to verbs. The semantic dependencies for (34a) *He thanks for the present* and (34b) *He is thankful for the present* are:



Using nominalisations, Helbig [ibid.:122-25] employs the same arguments as above to show that nouns can also have valencies. It turns out, however, that noun valency only applies to nominalisations and relational nouns like *connection to* and *brother of*.

In Valency theory, usually only the word classes verbs, nouns, and adjectives are discussed. The suggestion that e.g. a preposition has a noun valency is only a dependency grammar concept and usually absent from valency theory.

2.2.3.2 Engel

Engel [1996] has written a practical reference grammar based on dependency grammar for the German language and many theoretical dependency books, [Engel 1994] among others.

Engel does not only distinguish between constituency (which is based on a part-whole relationship) and dependency (which is based on governors and dependents), he introduces what he calls *Concomitance* [Engel 1994: 25]. *Webster's New Universal Unabridged Dictionary* describes the adjective *concomitant*:

> **1**. existing or occuring with something else, often in a lesser way; accompanying; concurrent: *an event and its concomitant circumstances.*

In Comcomitance representations, connected word forms appear on the same level. The represenations for (35) Mary loves Peter is:

```
( 35) Mary _______ loves __________ Peter
      SUBJ         V<SUBJ,OBJ>         OBJ
```

Concomitance is based on valency, but it does not recognise one of the connected word forms as a head and the other as dependent; both are on the same level.

Although most dependency theorists would agree that some of the decisions about which of two connected word forms is the head are arbitrary, Engel takes the extreme position of claiming that all of these decisions are completely arbitrary:

> Dependenz ist gerichtete Konkomitanz. Aber diese Ausrichtung ist keineswegs naturgewachsen oder irgendwie durch "die Sprache" vorgegeben, sie ist ein Artefakt, hervorgegangen aus einer willkürlichen Entscheidung des Grammatikers.
>
> (Engel 1994: 28)

2.2.4 The Finnish School

Finno-ugric languages are highly inflectional and have relatively free word order. They are therefore much easier and more intuitive to describe by dependency than constituency. No wonder that dependency theory is wide-spread in Finland. In the 70s and 80s, the Finnish School closely worked together with the German School. Korhonen [1977] is a review of the development of Dependency in Germany up to then. Tarvainen [1981] is an easy-to-read introduction to dependency from the viewpoint of the German school.

More recently, however, the Finns have been going their own ways, first with the Functional Dependency Parsing Language system [Valkonen et al. 1987]. Partly based on the constraint grammars initiated by Karlsson [1990], the most recent development, *Dependency Parser for English* [Järvinen & Tapanainen 1997], looks very promising because it partly goes back to the roots of Tesnière [1959] himself and tries to be functionally oriented. We will take a closer look at this system in 4.2.4.

2.2.5 The Prague School

The Prague School has contributed mainly to semantic research. Some of its representatives are Petr Sgall or Eva Hajičová, but already in the 50's and 60's linguists like Daneš or Skalička contributed fundamental research. For an introduction to the Prague school of dependency refer to [Sgall,Hajičová,Panevová 1986], [Hajičová, Skoumalová, Sgall 1995] and [Partee, Sgall 1996: therein esp. [Peregrin 1996]]. The Prague school is so far the only dependency shool to address Montague semantics and his formal semantics. An announced book with Barbara H. Partee as co-

athour has, to my knowledge, not yet been published, however; [Partee,Sgall 1996] indicates increased co-operation.

The Prague school is also unique among dependency schools in their explicit choice of a multistratal system. Of course every dependency system postulates an additional 'deep' level for semantics, but most dependency sytems use only one system for syntactic and functional representations. But the Prague school uses two syntacto-functional levels: Functional generative description (FGD) at a surface functional level, and tectogrammatical representation (TR) at the deep-functional level, which tries to approach the semantic level. One major reason why the surface FGD level is postulated is that the Prague framework is intended to be a generative grammar (we have seen in 2.1 that Tesnière is not interested in generation), the other is that they are sceptical: "However, it is not clear that a semantically oriented treatment of syntax is able to to handle syntactic ambiguities appropriately." (Sgall, Hajičová, Panevová 1986: 8).

I will not deal with FGD here, but I would like take a quick glance at TR, and one if its methods, topic-focus articulation (TFA).

2.2.5.1 Tectogrammatical Representation (TR)

In a nutshell, a tectogrammatical representation (TR) is a deep-functional dependency structure. It fully takes up functional categories into the semantic nucleus (cf. 2.3.2.5.6 and 2.3.8.3). It also uses a reduced theta-theory for verb valencies.

The theta-theory employed is reduced following typical criticisms:

> However sound the basis of such a differentation, the specification of the individual cases, as it is found in Fillmore's writings, differs from one study to another and does not offer more than vague characterizations in terms of semantic (cognitive) notions.
>
> (Sgall,Hajičová,Panevová 1986: 112)

They only accept *Actor* as a uniform semantic theta role [ibid.: 111ff], and – at least for English and Czech – they suggest four more types of functional roles for what Tesnière calls *actants*, namely *objective*, *addressee-dative*, *origin* and *effected* [ibid.: 134].

A TR for the sentence *John was in Prague yesterday* is e.g.:

(36)

$\mathrm{Declar}^{\mathrm{I,you,here,now}}(be^{t}_{\mathrm{Anter}}(\mathrm{Act}:John,\mathrm{Time\text{-}when}:yesterday,\mathrm{Loc}:Prague))$

TRs are admittedly rather more deep-functional than semantic structures:

> TR's are often thought of as being too close to the surface syntax to be able to render the meanings of sentences. However, it should be understood that what we describe here is literal meaning as structured by language itself, not the conceptual content (...)
>
> (Sgall,Hajičová,Panevová 1986: 154)

I will not deal with TR in more detail in this paper.

2.2.5.2 Topic-Focus Articulation (TFA)

I will give an introduction to TFA and show some of its advantages in 6.2, as described in Peregrin [1996].

2.2.6 Fraser and Hudson

The fact that it was Fraser [1996] to contribute a description of dependency in [Brown 1996] is enough to distinguish him as a key figure in today's dependency discussion. There he offers a historic description of dependency, tracing it back to the Greek scholars Thrax and Appollonius, medieval European grammarians and, of course, Tesnière. He also stresses the recent renaissance of dependency:

> Much of the formal work in DG [Dependency Grammar] carried out in the 1960's was directed towards solving problems in 'machine translation'(MT). After a gap of twenty years, during which very few 'natural language processing systems' employed DG, the 1980's saw a resurgence of interest, with several very large MT projects using DG [e.g. EUROTRA]. DG is now used in a wide variety of natural language processing systems such as parsers, natural language interfaces to databases, and, particularly, speech understanding systems. (Fraser 1996: 75)

Fraser has implemented a part of Hudson's dependency-based Word Grammar. For Word Grammar, cf. Hudson [1984, 1990] and 4.2.1. For Fraser's implementation, cf. [Fraser 1989].

Before developing Word Grammar, Hudson has defended dependency as fiercely as Mel'čuk:

2.2.6.1 Hudson versus Dahl

Hudson [1980a] argues that constituency can be completely abolished in favour of dependency, which in opposition could not be completely replaced by constituency. "My view is that constituency is *not* needed as well as dependency (...), and to support this view I shall now consider a number of arguments which have been used to, or could be used, in favour of constituency" (Hudson 1980a: 191). These arguments are:

- Linear sequence: Word order is relatively free in dependency grammars, just like in immediate dominance (ID) rules we know in certain PSGs. But Hudson stresses that "there is no problem in writing linearisation rules for dependency structures" (ibid.: 192). Link Grammar, e.g. uses strict linearisation by means of indicating whether links depart to the right (+) or to the left (-).
- Boundaries: In a clauseless representation as dependency is, boundaries need to be explicitly derived. Hudson [ibid.: 192-3] is positive that this will not pose major problems. The last word has probably not been said about this point. Just before finishing this paper I have discovered that Link

Grammar post-processing (see 4.3) tries to identify clause-boundaries and discards parses in which certain links cross clause-boundaries.

- Features on higher nodes: Some features are difficult to attach to single words. Hudson refutes the idea that features like 'mood' should be attached to a syntactical analysis at all, because he deems them to be semantic rather than syntactic in nature. [ibid.: 193-4]
- Categories of higher nodes: Hudson argues against the need for phrases and clauses, but he admits that his arguments are at present no more than "hand-waving" (ibid.: 194). Higher nodes are also available in dependency, as a derived concept (cf. 2.3.13).
- Headless constructions: In headless constructions, the concept of what a head can be needs to be discussed. This is a problem which which does not only arise for dependency, as head-driven PS theories face the same problems. [ibid.: 195]
- Propositions and Terms: This is the only point which Dahl directly attacks:

[Dahl 1980]'s reply disagrees in the following two points[1]

2.2.6.1.1 Propositions and Terms

Dahl [ibid.: 486] considers the example of *[ordinary [French house]]*, which needs some internal stacking in syntactic representation, if it should be able to prepare for semantic analysis. An ordinary *French house* is not just a *house* to which the concatenation of *ordinary* and *French* is applied, *ordinary* must be applied to *French house*.

In his reply to Dahl, Hudson suggests extending the use of the lexicon to cope with this example:

> For instance, there is a structure for *house* in the panlexicon, which is combined in the sentence structure with the structure for 'noun' (...), and which has various slots filled in to take account of the particular sentence and utterance in which it occurs – all without turning into anything which need to be recognized as a 'higher node'. (Hudson 1980b: 500)

As I understand Hudson, he wants to employ some kind of lexical rules to extend the panlexicon, much as can be done with LFG rules. (cf. [Schneider 1996] for an introduction to LFG lexical rules). In my opinion [Schneider 1997: 11] derivation is best dealt with by means of lexical rules, due its relatively high, but still quite restricted productivity. On the one hand, it is fair to conceive of *French house* as a lexical entry, because *French* is not only a modifier. On the other hand, *French house* is not an idiomatic expression. Although creating a lexical entry by a lexical rule pays

[1] There was a third point of disagreement concerning idioms and multi-words in the lexicon. However, this point turned out to be a misunderstanding only [Hudson 1980b: 490-5]

respect to the semantical problem (i.e. that *ordinary* does not only modify *house*, but *[French house]*), this creates the wrong impression that *French house* should be idiomatic.

Even if the case of *[ordinary [French house]]* may be solved by extending the lexicon, the general problem that internal stacking is impossible remains. In Radford's famous noun phrase [Radford 1988: 179 ff] *[[students of physics] with long hair]* it hardly makes sense to postulate a lexical entry for *students of physics*. We will come back to this problem in chapter 2.4.

2.2.6.1.2 Conjunctions

Dahl's argument about conjunctions [Dahl 1980: 487] convinces Hudson to "accept ... that, at least as far as coordinated conjunctions are concerned, there must be a higher node dominating the conjuncts" (Hudson 1980b: 497). Hudson therefore indirectly accepts Tesnière's *junctions*, and he admits that for this syntactic phenomenon, constituency is indeed necessary.

2.2.6.2 Conclusions

Debates like the one between Hudson and Dahl have shown that on the one hand any syntactical theory has to implement elements of constituency – at least for coordination –, but that, on the other hand, every constituent grammar also needs some kind of dependency relations – at least for verb valency, (cf. 2.2.3) – as Dahl himself states:

> I shall assume as 'common ground' the claims that just stating the constituency relations that hold in an utterance does not amount to an exhaustive grammatical analysis of it and that something like what has been called 'dependency relations' is needed in syntax. (Dahl 1980: 485)

Today, unlike in Mel'čuk [1988] or Hudson [1980a,b], the signs of the time are rather cooperation and integration.

> The debate between advocates of dependency and advocates of constituency [Hudson 1980, Dahl 1980] has lost much of its force as linguistic theories have increasingly come to incorporate aspects of both. (Fraser 1996: 75)

Chapter 2.3 explains in which ways many components of modern phrase structure grammars are basically dependency-based rather than constituency-based.

2.3 Basic Concepts of Dependency in GB and PSG terms

Covington [1992, 1994] offers a concise description of the dependency grammar (DG) philosophy:

> The key claim of dependency grammar (DG) is that all constituents are **endocentric**, i.e., every phrase has a most prominent element (the **head**) which determines its syntactic properties. Modifiers and complements of the head are called **dependents**.
>
> (Covington 1992: 1)

> The fundamental relation of DG is between head and dependent. One word (usually the main verb) is the head of the whole sentence; every other word depends on some head, and may itself be the head of any number of dependents. The rules of the grammar then specify what heads can take what dependents (for example, adjectives depend on nouns, not on verbs).
>
> (Covington 1994: 1)

If we want to use 'classical' GB or PSG terminology, then a number of concepts become central in dependency:

2.3.1 Subcategorisation and Valency

Subcategorisation means the division of different word classes into subclasses reflecting their different syntactic behaviour. The difference in syntactic behaviour consists in different sublasses having different *valencies*. Refer to chapter 4.2.2 for the argument of a possibly relevant distinction between valency and dependency.

In dependency, subcategorisation completely replaces all PS rules. In classical PS grammars, verbs subcategorise for subjects, objects, phrasal complements and the like. In dependency grammars, subcategorisation becomes all-encompassing. Nouns subcategorise for articles and optional adjectives, adjectives subcategorise for optional adverbs, participles subcategorise for auxiliaries, copulas subcategorise for predicative adjectives, subordinating conjunctions subcategorise for inflected verbs, and so on.

In a lexicalist or a strongly lexicon-based grammar, the term *valency* is preferable to *subcategorisation*, because where there are no categories there are no subcategories. What counts is that individual verbs open up positions for nouns to be filled, i.e. valencies. If one thinks of a sentence as a projection up from the lexicon rather than an instantiation of a PS rewrite rule, the term valency seems again more appropriate.

Valency is central to dependency grammar, as e.g. Schubert [1988] notes:

> Bei einer Präzisierung des Begriffs Valenz geht es um die Beschreibung von Kombinierbarkeitsregeln für Wörter (und eventuell für andere sprachliche Einheiten). Ein Grammatikmodell, das die direkten Beziehungen zwischen Wörtern zur unmittelbaren Grundlage hat und sie nicht erst auf dem Umweg über abstrakte Einheiten erschliesst, ist die **Dependenzgrammatik**. Es ist daher kein Zufall, dass der Begriff der Valenz in die Dependenzgrammatik gehört. Die

> Valenz ist eine der zentralen Grössen in den Arbeiten TESNIERES, der ja, trotz ähnlich gelagerter Denkrichtungen in den Jahrzehnten und Jahrhunderten vor ihm, als der Begründer der Dependenzgrammatik gilt.
>
> (Schubert 1988: 56)

Like Tesnière, Helbig focusses his dependency research on valency. An introduction to Helbig was given in 2.2.3.1.

2.3.2 Heads

> A further point of contact between dependency theory and modern syntactic theories is the central place of the notion 'head' in X-bar theory – which indeed could be defined as a version of phrase structure grammar in which every phrase is required to have a lexical head, where 'head' is used in almost the same way as in dependency theory. According to [Radford 1988: 545 ff] every construction is now assumed to have a head, an assumption expressed as 'The Endocentricity Constraint'.
>
> (Hudson 1990: 111-112)

Heads always subcategorise for dependents. A sentence element (a word, a compound, a nucleus or an idiomatic expression) may have several dependents, but usually only one head. Because the notion of head is so important, discussions on its status take a very prominent role in dependency theory. E.g. in [Jung 1995: 33 - 83], almost a third of this general book on dependency is devoted to the head question. I will follow his subchapter divisions for the following subchapters.

There are different uses and definitions of the notion of *head* in different types of grammars. They may be *semantic,* as in the case of Categorial Grammar , where the *functor* is the head, and the *argument* is the dependent. In X-Bar theory, on the other hand, heads are defined by syntactic means. On a *morphological* basis, we may also come to different conclusions about what is the head. E.g. in a simple Subject+Verb Construction, the subject may be seen as selecting Person and Number of the Verb.

2.3.2.1 Zwicky's head arguments

Zwicky [1985: 4-14] suggests the following arguments for determining the head of a combination of two constituents:

a. **The semantic argument**: in a combination X+Y, X is the "semantic head" if speaking very crudely, X+Z describes a kind of the thing being described by X. (Cf. 2.3.2.2)

b. **The subcategorisand**: the constituent which is subcategorized with respect to its sisters. (Cf. 2.3.2.4.1)
   E.g. in a PP, the preposition is subcategorised, unlike its sister NP. The preposition is therefore a head candidate.

c. **The morphosyntactic locus**: the (actual) inflectional locus within a construction is a candidate for the head of the construct. (Cf. 2.3.2.3).
   E.g. for a VP, the auxiliary verb carries the inflection and is a head candidate.

d. **The governor**: the constituent which determines the morphosyntactic form of some sister. (Cf. 2.3.2.3)

E.g. in a VP, the verb is head candidate, because it determines the case of an object

e. **The determinant of concord**: the constituent with which some or other constituent must agree. (Cf. 2.3.2.3)
   E.g. in a S, the Subject N is head candidate, because the main verb must agree with it.

f. **The distributional equivalent**: the constituent that belongs to a category with roughly the same distribution as the construct as a whole. (Cf. 2.3.2.4.5)
   E.g. in a NP, adjective + noun and noun only are both NPs, which makes the noun be a head candidate.

g. **The obligatory constituent**: the constituent which has to be present if the mother is to be categorized as it is. (Cf. 2.3.2.4.3 and 2.3.2.4.4)
   E.g. in an ADJP, an adjective is obligatory and therefore head candidate, while adverbs are not.

h. **The rulers in dependency grammar**: the ruler is the word on which other words depend.
   Since the head discussion here should support a head definition for dependency, using this argument here would lead to circularity. We will leave it away.

Zwicky discusses these arguments by use of the following English constructions:

1. DET + N, as in those penguins;
2. V + NP, as in control those penguins;
3. AUX + NP, as in must control thos penguins;
4. P + NP, as in toward those penguins;
5. NP + VP, as in we control those penguins;
6. COMP+S, as in that we control those penguins

This chart is based on Zwicky [1985]:

Zwicky[1985]	**V+NP**	**P+NP**	**NP+VP**	**DET+N**	**AUX+VP**	**COMP+S**
(a) Semantic Argument	NP	NP	NP	N	VP	S
(b) Subcategorisand	V	.	.	DET	.	COMP
(c) Morphosyntactic locus	V	P	VP	N	AUX	S
(d) Governor	V	P	VP	.	AUX	.
(e) Determinant of Concord	NP	.	NP	N	.	.
(f) Distribution. Equivalent	V	.	.	N	VP	S
(g) Obligatory Constituent	V	P	VP	N	VP	S
(Dependency Ruler)	V	.	.	N	VP	COMP

Table 1: Zwicky's heads

Hudson [1987] has re-analysed Zwicky [1985], using the above arguments except for (e.) *determinand of concord*, and summarises as follows:

Hudson[1987]	**V+NP**	**P+NP**	**NP+VP**	**DET+N**	**AUX+VP**	**COMP+S**
(a) Semantic Argument	V	P	VP	DET	AUX	COMP
(b) Subcategorisand	V	P	.	DET	.	COMP
(c) Morphosyntactic locus	V	P	VP	DET	AUX	COMP
(d) Governor	V	P	VP	.	AUX	.
(f) Distribution. Equivalent	V	.	.	DET	AUX	COMP
(g) Obligatory Constituent	V	P	VP	DET	AUX	COMP
(Dependency Ruler)	V	.	.	DET	AUX	COMP

Table 2: Hudson's different heads

The striking differences between these two analyses has led [Jung 1995] to take up the head discussion again, grouping the arguments into morphological, semantic and syntactical criteria. In the following discusion I will proceed from Jung' criteria.

2.3.2.2 Semantic Criterion: Function and Argument

This criterion corresponds to Zwicky's [1985] (a.) *semantic argument*.

Arguments modify predicates in function-argument structures. In *f(A), A* modifies *f*. *f(A,B)* is not the same as *f(A)*, but the same function with a different arity. Also *f(X)* ist still a kind of *f()*. *f()* can therefore be seen as the head, on which *A* depends.

Assuming Boolean true for the following functions, *f(A)* expresses that *A* is an element of the set *f*, while *A(f)* expresses that *f* is an element of the set *A*. *sleeps(Peter)* means that there is a set of sleepers, and that Peter is one of them. *Peter(sleeps)* means that sleeping is one of the activities of the set of activities of Peter. It is difficult to assess which of these functions is "closer" to the meaning of the sentence *Peter sleeps*. This reminds us of Engel's view [1994: 28] that the dependency direction should be mainly arbitrary (cf. 2.2.3.2). On the other hand, let us remind that Weber [1997: 22] stresses this root dependency is basically arbitrary, unlike others (cf. 2.1.1.2). We will see in 6.2 that *sleeps(Peter)* can be conceived of as the non-marked reading with *Peter* as *topic* and *sleeps* as *focus*, according to Sgall, Hajičová, Panevová [1986] and Peregrin [1996].

A function opens up *valencies* for arguments. Based on existence considerations (no adjective occurs without its noun) we can say that also attributive adjectives open up a valency for a noun, i.e. *tall(Peter)*. Different arguments lead to different answers about which element is functor and which is argument. In practice, such

considerations are often not important on the syntactic level. In our ExtrAns system, e.g., *Peter is tall* is represented by *Peter(x) AND tall(x)*[2], i.e. the noun and adjective are simply identified with each other, or *Peter drinks beer* is represented as *event(drink, Peter, beer).* Note, however, that on the semantic level, *topic-focus* considerations are not only important, they are essential. It is therefore questionable if it would not be advisable to use syntactic structures which are as close to semantics as possible. E.g. to represent adjectives as intersections, e.g. *Peter(x) AND tall(x),* completely blurs the distinction between presupposition and assertion.

The semantic criterion can be seen in line with the semantic question of the direction of the functional application in Montague grammar, which is a free variable and needs to be empirically tested for each grammar rule.

Jung [1995: 85] rejects the semantic criterion as a definition for syntactic head: in his view, dependents are allowed to have several heads in the semantic criterion, a fact which contradicts the fundamental dependency principle that each dependent has exactly one head. I therefore abandon this criterion.

2.3.2.3 Morphological Criterion: Head- vs. Dependent-Marking

Zwicky's [1985] arguments (c.) *morphosyntactic locus*, (d.) *governor* and (e.) *determinand of concord* are all morphological arguments and they are all dealt with together in Jung [ibid.].

Syntactic relations may be marked morphologically either on the head of a constituent or on its dependent, and this is a language-specific parameter, according to [Nichols 1986]. German and English can be classified as dependent-marking languages, i.e. they mark syntactic relations on the dependent. Therefore, the marked item is the dependent in German and English. This predicts that in most cases for dependent-marking languages, morphological and syntactical dependencies run in parallel. Note, however, that in very simple subject-verb constructions like

(37) The man walks.

this does not necessarily hold true, as the subject determines the verb morphologically, while (at least in the GB perspective, cf. 2.3.2.5.5 and 2.3.9) the subject depends on the verb syntactically. Zwicky's argument (d.) *determinand of concord* accordingly suggests NP as a head candidate of a NP+VP construction, in opposition to the standard VP head. Jung's answer [ibid: 72] is that INFL assigns Case to the subject NP. This is a standard X-bar assumption which will be explained in 2.3.2.4.2. Let us remind, however, that Mel'čuk [1988:109] has warned us of using

[2] This is simplified, but the comment still applies.

morphological dependency as a criterion for defining syntactical dependency in 2.2.2.1.1.

2.3.2.4 Syntactical Criteria

Since dependency grammars are intended to parse syntactical sentences, dependency is a syntactical concept, and the list of syntactical arguments for the definition of heads suggested by Jung [1995:44-59] is longest.

2.3.2.4.1 Subcategorisation

This criterion corresponds to Zwicky's [1985] (b.) the *subcategorisand* argument.

Subcategorisation, introduced in 2.3.1, is only partially suitable as a head definition argument because most authors consider only certain word classes (V, N, and Adj) to be subcategorisands. However, based on the observation that the concept of valency can be extended to all word classes, [Jung ibid.: 46] suggests redefining subcategorisation:

> Definition: In einer Konstruktion "A+B" ist A ein Subkategorizand, wenn A in bezug auf den Begriff der syntaktischen Relationalität eine Leerstelle für B eröffnet, wobei A (Subkategorizand) ein Kopf von B ist. (Jung 1995: 46)

Which would mean, in most simple terms, that a head predicts dependents. While this may be appropriate for verbs, it leads to incorrect results with adjectives. While verbs predict their arguments, adjectives are usually not predicted in any sense by nouns – they are usually optional. Rather the opposite seems to apply: the presence of an adjective predicts a noun. Jung's above definition can be seen as the opposite of Mel'čuk's[1988] *predictability criterion* (cf. 2.2.2.1.3.2):

> [T]he [syntactic] dependent is the element that (statistically) predicts the whole phrase ... This criterion is based on the hypothesis that any modifier (i.e. dependent) predicts its head (i.e. governor) ... This is true for some cases, e.g., for adjectives
>
> (Mel'čuk 1988:140).

But adjectives cannot be accepted as heads because nouns with articles and (perhaps several) adjectives receive several heads by this definition - a transgression of a basic dependency principle.

Jung therefore abandons the subcategorisation argument. Although Zwicky's argument (b.) *subcategorisand* was formulated differently, it runs into the same problems.

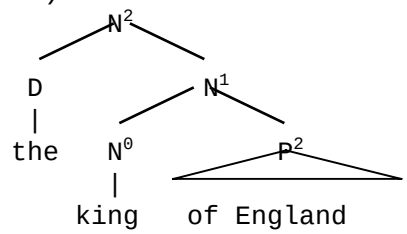
2.3.2.4.2 X-bar syntax

Unlike Jung [1995], Zwicky [1985] does not use X-bar arguments.

This chapter can not be an introduction to X-bar, for which the reader is referred to Cowper[1992] and Radford[1988,1997].

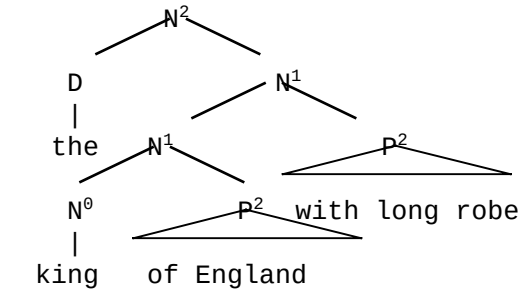
Radford[1988] shows the need for intermediate categories between phrases and terminal nodes, by means of *one-pronominalisation* [ibid.:174ff] and the differentiation between adjuncts and complements [ibid.:175ff]. Such arguments are the starting point for X-bar syntax, developed by Chomsky[1970] and Jackendoff[1977], based on Harris' [1951] system of 'raised numbers' [ibid.:266]. The basic idea is that the lexical head receives a numerical superscript, the lexical head together with adjuncts and complements the same numerical superscript raised by one, and this latter together with the determiner a superscript raised by one again - e.g. for the NP "the king of England":

(38)



It is also conventional to write X'' for X^2 and X' for X^1. The bar-level increases when moving up from the terminal lexical node, vice versa it decreases when stepping down the syntactic tree. But optional elements, *adjuncts*, may be inserted at the intermediate level under X'-recursion - e.g. for the NP "the king of England with long robe":

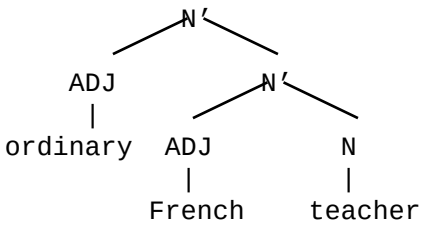
(39)



Usually, a maximal bar level of two is assumed. The node at the maximum bar level is called *maximal projection*. The term *projection* is used to illustrate that the syntactic structure is projected up from the lexical head. X-bar theory has paved the way for lexicalism in GB theory, finally leading to the complete abolition of PSG rewrite rules. The X-bar structure remains as the only structural constraint next to the lexicon (cf. Cowper 1992: Chapter 4, pp 57-70). A maximal bar level of two means that there is only one intermediate level (X'), which permits two distinctions: recursion to X' for adjuncts, and expansion to X^0 for complements, irrespective of whether they are subcategorised for or not.

Complements are semantically closer to the lexical head than adjuncts. But unfortunately, X-bar syntax cannot provide an answer to the dispute between Hudson[1980a, 1980b] and Dahl[1980] (cf. 2.2.6.1) about *[ordinary [French house]]*. *French* is not a complement to *house*, as it does not specify the house but only adds additional information. Let us take a more suitable example, e.g. *the ordinary French teacher*, in which *French* specifies the kind of *teacher* and is thus a complement, attached under expansion to X^0, while *ordinary* is adjunct, attached under recursion to X':

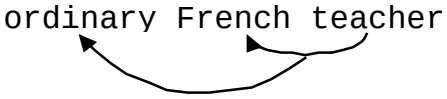
(40)



The *semantic proximity* determines the syntactic order: **French ordinary teacher* is unacceptable. In dependency, a distinction between complements and adjuncts is impossible to make, because higher nodes are unknown. Both *French* and *ordinary* depend on *house* directly:

(41)

But the desirable dependency structure, which would involve a concept of higher nodes, would be:



(42)

If the syntactic proximity always expresses semantic proximity, then it is nevertheless possible to predict the internal semantic hierarchy by starting to semantically attaching syntactically close elements first and progressively attach syntactically more remote elements later. In X-bar theory, complements are required to syntactically appear closer to their heads than adjuncts, otherwise the tree branches would cross. Accordingly, it must be possible to convert dependency structures into X-bar structures, as Covington[1992, 1994a] shows. We will discuss this in 2.4.5.

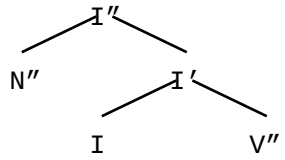
The lexical head and its projections bear the same grammatical features at all bar levels in X-bar syntax. As for the definition of heads, X-bar theory makes the same assumptions as Generalised Phrase Structure Grammar (GPSG) and Head-Driven Phrase Structure Grammar (HPSG) in their *Head Feature Principle* (HFP). Like Mel'čuk's [1998:135] *morphological contact point* (cf.2.2.2.1.3.2) and one of the following criteria of Jung [1995:54ff] in 2.3.2.4.5, the X-bar criterion also seems to boil down to a version of HFP (cf. 2.3.2.6).

In X-bar syntax, the top-node rewrite rule

(43) S -> NP VP

cannot be explained, because *S* is no maximal projection. It has therefore been suggested (cf. Radford[1988:508-15]) that *S* should be a maximal projection of *INFL* (also *I*), i.e. *S=I''*. Because I is marked on the verb and assigns nominative case to the subject (cf. [Cowper 1992:99]), the subject morphologically depends on the verb and not only vice versa. In X-bar syntax, *I* is the head of the subject:

(44)



2.3.2.4.3 Existence Relation ("Vorkommensbedingung")

This criterion corresponds to Zwicky's [1985] (g.) the *obligatory constituent* argument, as the next criterion, *obligatory constituent* (2.3.2.4.4), will explain.

This criterion is based on Engel [1971], who states "dass hier Dependenz als Vorkommensrelation verstanden wird, die es erlaubt, aus dem Vorkommen eines Elements mit zu spezifizierender Sicherheit auf das Vorkommen anderer Elemente zu schliessen" (Engel 1971:141). Nothing is said about the direction of the dependence in this quotation, however. Can we predict the head from the dependent, or vice versa? Either way, this argument looks reminiscent of the refuted predictability arguments in 2.3.2.4.1. But the difference is that Engel does not speak about the predictability of a word class, but the predictability of the existence. Jung's [ibid.] formulation:

> In einer Konstruktion "A+B" ist A ein Kopf (Regens) von B, wenn das Auftreten von A syntagmatische Vorkommensbedingung für B ist. (Jung 1995: 52)

This means that the dependent B can never occur alone, or A <- B. This criterion therefore turns out to be what Mel'čuk [1988:139-40] calls the *omissibility* criterion (cf. 2.2.2.1.3.2), or the *Weglass-Test* from Weber [1997:35]. Mel'čuk [ibid.] discusses two examples in which the dependent is obligatory, while the head can be omitted. In accordance with Weber, Mel'čuk [ibid.] agrees, however, that omissibility is an efficient heuristic means that works in most cases.

2.3.2.4.4 Obligatory Constituent ("Obligatorik")

That the head should be the obligatory word form of two connected word forms can be directly derived from the existence criterion above. Jung [1995:52] admits that these two criteria are basically the same. Therefore, not only this criterion, but also the existence criterion above correspond to Zwicky's [1985] (g.) the *obligatory constituent* argument.

2.3.2.4.5 Shared Syntactic Features ("Merkmalsbeständigkeit der Kategorie")

This criterion partly corresponds to Zwicky's [1985] (g.) the *distributional equivalent* argument. Zwicky starts from a language use related perspective (similar distribution in a text collection of head and whole construct), Jung from a morphological perspective, in which he mainly speaks about word classes. The common argument is that the head and the whole construct has to inherit its syntactic behaviour from the (word class of) the lexical head.

This criterion is a close relative or perhaps even identical to Mel'čuk's [1988:132] *syntactic role* criterion (cf. 2.2.2.1.3.2). Jung's [1995] definition is as follows:

> In einer Konstruktion "A+B" ist A der Kopf, wenn syntaktische Merkmale der gesamten Konstruktion "A+B" mit den syntaktischen Merkmalen von Kategorie A übereinstimmen. (Jung 1995:58)

Jung goes on to state that this criterion is also expressed by the GPSG and HPSG Head Feature Principle (HFP). This criterion thus boils down to yet another version of HFP. This entails that the essence of both of Mel'čuk's [1988:132-140] syntactical dependency direction criteria (cf. 2.2.2.1.3.2) are expressed by the HFP. We will take a closer look at the HFP in 2.3.2.6.

2.3.2.5 A Re-Analysis of Zwicky's Constructions

The above discussion allows us to analyze the constructions suggested by Zwicky [1985] again. We recall that Jung[1995] has abandoned (a.) *the semantic criterion* (cf. 2.3.2.2) and (b.) *the subcategorisation criterion* (cf. 2.3.2.4.1). Without these, Zwicky's chart looks as follows (where Hudson[1987] or Jung[1995] have come to different conclusions the constituent is ***in bold italics***):

Zwicky[1985]	**V+NP**	**P+NP**	**NP+VP**	**DET+N**	**AUX+VP**	**COMP+S**
(c) Morphosyntactic locus	V	P	VP	***N***	AUX	***S***
(d) Governor	V	P	VP	.	AUX	.
(e) Determinant of Concord	***NP***	.	***NP***	***N***	.	.
(f) Distribution. Equivalent	V	.	.	***N***	***VP***	***S***
(g) Obligatory Constituent	V	P	VP	***N***	***VP***	***S***

Table 3: Remaining Differences between Zwicky and Hudson

2.3.2.5.1 V+NP:

After dismissing (a.) *semantic argument* (cf. 2.3.2.1) only (e.) *Determinant of Concord* (which Hudson [1987] leaves away) has NP as head in this construction, according to

Zwicky [1985]. At least for English, this analysis seems unacceptable, however, as Jung [1995] points out, especially as it is based on subject agreement, which has nothing to do with the object here.

> Zwicky und Hudson [sehen] die Träger des Flexionsmorphems als Köpfe an: z.B. in einer Konstruktion "controls penguins" ist das Verb "controls" der Kopf, weil das Verb ein Flexionsmorphem "s" besitzt. Aber dies ist nicht akzeptabel, weil das Flexionsmorphem "s" mit NP gar nicht im Zusammenhang steht. Im Englischen weist das Verb stets einer Objekt-NP einen Kasus zu. (Jung 1995: 63).

Because the verb determines case concordance on the NP, also (e.) *Determinant of Concord* has V as head.

In languages like French, however, verb participles may have to agree with the NP:

```
( 45a) Il a acheté un livre.
(   b) Il a achetée une bière.
```

2.3.2.5.2 P+NP:

After dismissing (a.) *semantic argument* (cf. 2.3.2.1) P is always head.

But if we consider that *to Mary* in the sentence *I gave the book to Mary* and *Mary* in the *sentence I gave Mary the book* have the same grammatical function it remains questionable if they should be phrases with different heads. On the one hand such is the difference between syntax and a functional representation, on the other hand since prepositions can be thought of as grammatical markers much like Case why should one not try to use a more functionally oriented syntactic structure?

> Interessant ist auch die Frage, ob der Präposition eine Valenz zukommen kann. Engel [1977:93] sieht in einer Präpositionalphrase die Präposition als Regens an ... Nach dieser Auffassung würden die Präpositionen über eine Valenz verfügen. Dafür spricht auch die Tatsache, dass die Präpositionen immer ergänzungsbedürftig sind. für+Akk., vor+Akk./Dat. Wir möchten jedoch bei Präpositionen von einer anderen Art Valenz sprechen als z.B. bei Verben und Adjektiven. Die Präpositionen haben meist keine aussersprachlichen Referenten, sie sind grammatische Hilfswörter, denen in einigen Sprachen (wie im Finnischen) Endungen entsprechen. Vielleicht könnte man bei den Präpositionen von einer *grammatischen Valenz* sprechen.
>
> (Tarvainen 1981: 10)

In this perspective also the unanimity about P as the head is not really satisfying. We will come back to the discussion of marker versus functional head in 2.3.2.5.6.

2.3.2.5.3 Subject NP+VP:

After dismissing (a.) *semantic argument* (cf. 2.3.2.1) only (e.) *determinant of concord* (which Hudson [1987] leaves away) predicts NP to be head in Zwicky's [1985] chart. If we follow the X-bar syntax stipulation that INFL assigns nominative Case [Radford 1988:508-14], the morphological dependency is mutual and a determinant can no

longer be determined: NP determines person and number, VP determines Case. Like the X-bar theory argument presented by Jung [1995:46-51,72], all determinable arguments now suggest VP as head.

In more recent developments of GB, the subject NP becomes part of the VP in the D-structure to express the dependency more clearly, as 2.3.9 explains.

2.3.2.5.4 DET+N:

For this construction it seems to be hardest to determine a head and no clear answer seems to emerge yet. For Zwicky[1985], all of the arguments (except for the abandoned (b.) *the subcategorisation criterion*) suggest N as head, for Hudson [1987], the same arguments suggest DET as head. Engel [1994] describes the current indecidability "ob in Nominalgruppen wie ein Ast das Nomen den Artikel oder der Artikel das Nomen regieren soll. Er möge, vor allem, für seine Ansichten Argumente beibringen. Vermutlich wird er erkennen, dass sich für beide Ansichten gleich gute Argumente finden lassen." (Engel 1994: 28)

Let us carefully consider the remaining criteria:

- **Morphological Criterion** (c.), (d.), (e.) (cf. 2.3.2.3): The DET agrees with the lexical gender of the N. Adjectives within an NP or in languages like French or Danish even predicative adjectives outside the NP have to agree with N, which is morphologically clearly the head. In German, which knows strong and weak adjective declension, however, the article also influences the adjective declension, as in *ein grosser Mann* versus *der grosse Mann*.
- **X-bar syntax** (cf. 2.3.2.4.2): Traditionally in X-bar theory, DET takes a NP *specifier* (SPEC) position - this is the prototypical use of the specifier position. In this assumption, N is clearly the head. In more recent versions of GB, however, determiners have their own phrase (so-called DPs) in which the NP is only a complement (cf. Cook and Newson [1996: 183-5]). In this assumption, DET is the head, with N as a more local sub-head. 2.3.10 summarizes some of the proposed evidence in favour of the DP hypothesis.
- **Existence Relation** (g.) (cf. 2.3.2.4.3) and **Obligatory Constituent** (g.) (cf. 2.3.2.4.4): At first sight one may get the impression that the noun is obligatory and the determiner optional. However, in many cases the determiner is no less obligatory than the noun itself, e.g. *The book is boring* versus **Book is boring* or *My father is a doctor* versus **Father is doctor*. This criterion does not clarify anything.
- **Distributional Equivalent** (f.) (cf. 2.3.2.4.5): Intuitively, *DET+N* behaves like N. But whenever the determiner is obligatory this is not true. Since pro-forms

become pro-DPs under the DP hypothesis the two have to be closely related. Hudson [1987] even argues that they are often grammatically the same, which Jung [1995:75] finds unacceptable.

Like in *subject+verb* constructions, in *DET+N* constructions both elements are usually obligatory, both are usually semantically present even if one of them is not syntactically expressed. Perhaps Engel's term *Concomitance* [Engel 1994:23-28] suits better than dependency, because both elements require each other.

For the rest of this paper I will mainly use the traditional dependency and GB analysis with N as the head, but this decision is arbitrary. Note that whatever decision is taken, it does not affect the convertibility between dependency and constituency outlined in 2.4.5.

2.3.2.5.5 AUX+VP:

Following X-bar terminology, the title *INFL+VP* would be more correct. If INFL is the head of the sentence it governs VP and everything else, so that from the current perspective it is difficult to follow Zwicky's arguments for perceiving of VP as the head candidate for these arguments:

- **Existence Relation** (g.) (cf. 2.3.2.4.3) and **Obligatory Constituent** (g.) (cf. 2.3.2.4.4): In answers or tag questions the VP may well be lacking.
- **Distributional Equivalent** (f.) (cf. 2.3.2.4.5): At least finiteness and mood are present on INFL rather than on VP. An auxiliary verb in a tag question or an answer shows the same syntactic distribution as together with its VP.

Most authors will accept INFL as head. Note, however, that auxiliaries are function words [Finegan 1989: 175], that INFL is a functional head and can be thought of as a mere grammatical marker dependent on VP. Tesnière, e.g. keeps the auxiliaries and main verbs in the same nucleus, so do Järvinen & Tapanainen [1997:25] in their system: "Internally in the parser, the verb chain is considered as one syntactic element". We will come back to the functional marker analysis below.

HPSG [Pollard & Sag 1994], which is skeptical about functional heads, as we shall see below, does not use an INFL node. One of the suggestions brought forth [Borsley 1996: 88-90] is to treat the main verb (or VP except for AUX) as a complement of AUX, a suggestion that assumes that AUX is head, although they syntactically appear at the same level. Because the semantic argument is composed of the inflected AUX and the main verb this suggestion is called *argument composition*. I will show an implementation of this in 3.3.3.4.

2.3.2.5.6 COMP+S:

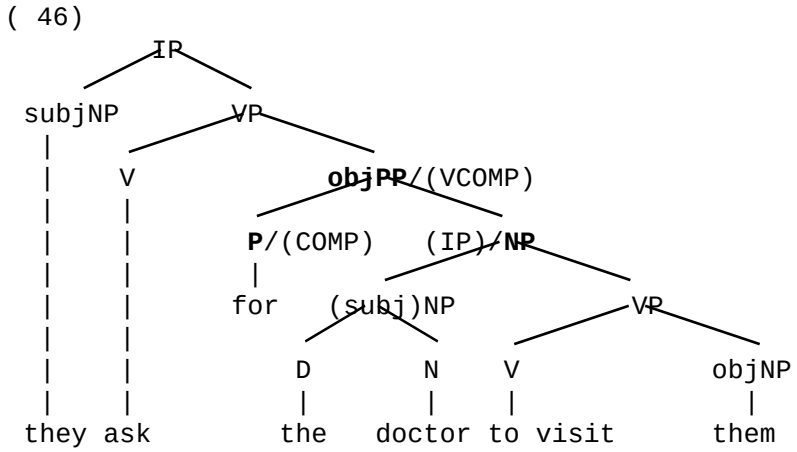
Also this argument has found a clear answer in X-bar syntax.

- **Morphological Criterion** (c.), (d.),(e.) (cf. 2.3.2.3): COMP requires specific S. COMP determines whether S shall be finite or not.
- **Existence Relation** (g.) (cf. 2.3.2.4.3) and **Obligatory Constituent** (g.) (cf. 2.3.2.4.4): In English, there are both examples with optional COMP as well as with optional S: *I think (that) the penguins are ready to eat* versus *I haven't seen him since (he left).* This criterion cannot help to decide.
- **Distributional Equivalent** (f.) (cf. 2.3.2.4.5): Following the examples above, this criterion also fails to decide.

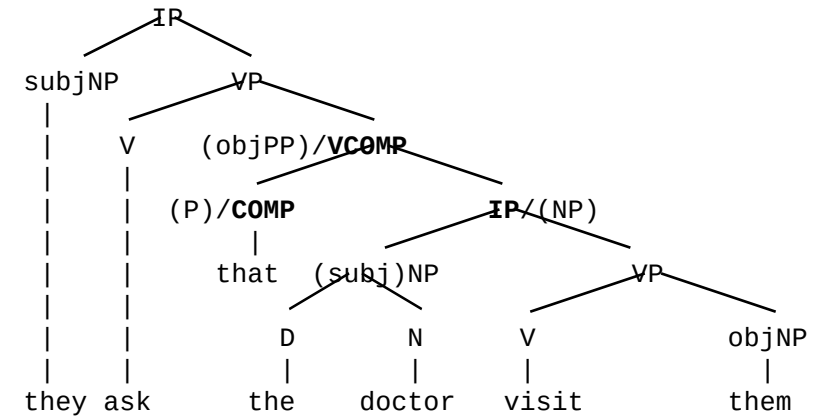
Pollard and Sag [1994:44-5] ask themselves if some of the decisions about which participant in a connection should be the head are rather based on X-bar internal evidence.

Jung [1995:71] discusses the close similarity of *P+NP* to *COMP+S*. Both are verb complements, i.e. selected by the verb valency, and they can often be transformed into each other. Although P assigns Case to NP or COMP assigns [+/-finite] to S, NP and S depend on the verb.

Sentences (46) *They ask for the doctor to visit them* and (47) *They ask that the doctor visit them* shows the gradience from *COMP+S* to *P+N*. The tree representations chosen here are deliberately non-X-bar compliant:

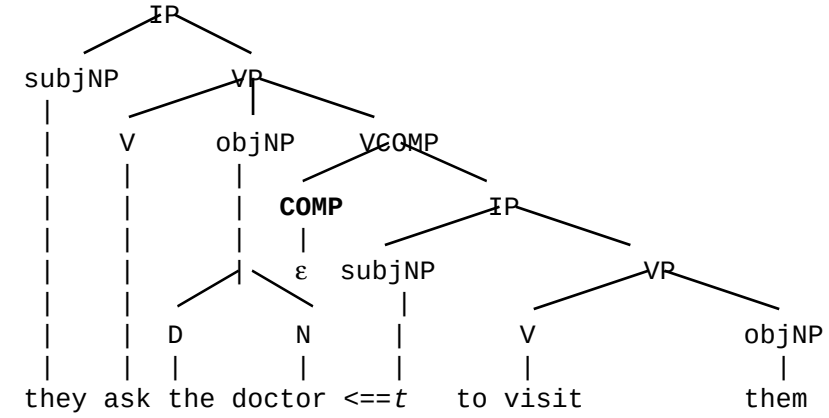


(47)



One version of this sentence also exists without any preposition: (48) *They asked the doctor to visit them*. Apart from the fact that it is now a control structure it is semantically very close to the two first sentences. Note that in an X-bar analysis, the COMP-head even remains empty:

(48)



It is questionable whether

(i) verb complements in *COMP+S* versus *P+NP* should be considered as fundamentally different,

(ii) verb complements *with* versus *without* preposition should be kept apart,

(iii) COMP and P should be head if they only play a grammatical role. Pollard and Sag [1994] suggest treating complementizers and prepositions as so-called *markers*.

> On our account, a marker is a word that is 'functional' or 'grammatical' as opposed to substantive, in the sense that its semantic content is purely logical in nature (perhaps even vacuous). A marker, so-called because it *marks* the constituent in which it occurs, combines with another element that heads that

> constituent. In addition to the complementizers *that* and *for*, other examples of markers include the comparative words than and as, the case-marking post-clitics of Japanese and Korean, and perhaps nonpredicative adpositions in (the vast majority of) languages where adpositions stranding does not occur.
>
> (Pollard & Sag 1994. 44-5)

Let us recall that Tesnière treats functional words as part of the nucleus (cf. 2.1.1.1). Functional words, according to [Finegan 1989: 175] are prepositions, conjunctions and auxiliaries. Tarvainen [1981:10] uses the term *grammatical valency* for dependencies from functional elements (cf. 2.3.2.5.2).

2.3.2.6 Head-Driven Phrase Structure Grammar (HPSG)

In the above subchapters, three of the presented criteria turned out to relate very closely to the HPSG *Head Feature Principle* (HFP) [Pollard & Sag 1994: 34]. They are:

- Mel'čuk's[1998:135] *morphological contact point* (cf.2.2.2.1.3.2)
- *X-bar syntax* (cf. 2.3.2.4.2)
- *Shared syntactic features* and *distributional equivalent* (cf. 2.3.2.4.5)

The essence of these criteria and HFP is that head and dependent share at least some syntactic features, and that only head and dependent "know" each others' features.[3] Across a chain of dependencies it is possible to share features with more remote nuclei (*morphological contact point*). Those features which are required to be shared (*shared syntactic features*) are the values of the HEAD attribute. Minimally, they are the word category, ensuring *distributional equivalent*. Pollard & Sag [1994] define HFP as follows:

> The HEAD value of any headed phrase is structure-shared with the HEAD value of the head daughter.
>
> (Pollard & Sag 1994: 34)

They explicate [ibid.] that "[t]he effect of the HFP is to guarantee that headed phrases really are projections of the head daughter" - a lexical *X-bar syntax* argument.

Borsley [1996:50-1] points out that, unlike for the GPSG Head Feature Convention which was a default principle that could be overridden, HFP is an absolute principle. This means that HPSG *requires* all of its accepted structures to conform to dependency theory. In this sense, HPSG is a dependency theory.

Two other fundamental HPSG principles are also dependency principles. They are the *Subcategorisation Principle* and the *Valence Principle*. The valence principle, "a generalized version of the Subcategorisation Principle" (Borsley 1996: 84):

> In a headed phrase, for any valence feature F, the F value of the head is the concatenation of the phrase's F value with the list of SYNSEM values of the F daughters.

[3] The head (say A) selects which features to share with which of its dependents (say B and C). But no dependent (e.g. B) has access to the features of the other dependent(s) (e.g. C)

(Borsley 1996: 84)

"The effect of these principle[s] is to 'check off' the subcategorization requirements of the lexical head" (Pollard & Sag 1994: 34), they work much the same way as cancellation in Categorial Grammar [ibid.].

2.3.2.7 Conclusions

Jung [1995] illustrates that some of the eight criteria discussed in the book lead to different heads than others, in order to finally conclude that especially the two criteria *subcategorisation* and *semantics* lead to controversial results. After excluding them, however, the definition of heads for the set of constructions employed is uncontroversial, except for the construction *DET+N*. The remaining criteria are:

- Morphological Criterion (c.), (d.),(e.) (cf. 2.3.2.3): Mel'čuk 1988: 109) warns us about basing syntactic dependency on morphological dependencies.
- X-bar syntax (cf. 2.3.2.4.2): Finds clear and systematic answers for most cases. Pollard and Sag [1994] warn that some decisions are rather based on X-bar internal than *functional* considerations.
- Existence Relation (g.) (cf. 2.3.2.4.3) and Obligatory Constituent (g.) (cf. 2.3.2.4.4): Works for most cases, at least as a pragmatic test, but cannot decide e.g. for COMP+S.
- Shared syntactic features (f.) (cf. 2.3.2.4.5): as expressed by the HPSG head feature principle. Together with a valency principle it is the cornerstone of HPSG theory.

For constructions like *subject+verb*, *AUX+VP* and *DET+N*, perhaps even *COMP+S* and *P+NP* it is questionable, however, if a clear dependent should be established, as both elements usually require each other. It is justifiable to think of them in terms of what Engel [1994: 25] calls *Concomitance* or to think of the first element in these constructions (except for *subject*, of course) as a functional marker or head. Pollard and Sag [1994] raise the question of how *surface-oriented* or how *functional* a syntactic theory should be, and if we should accept theory-internal arguments at all. Should we e.g. treat articles as heads (as in recent X-bar) if the arguments brought forth are to a big extent based on X-bar internal considerations, or as functional grammatical markers, as also Tesnière has suggested (cf. 2.1)? Theory-internal arguments are dangerous because the assumptions of linguistic theories cannot be scientifically proved, as I have explained in 1.3. It is therefore preferrable to base arguments on evidence.

The restrictions X-bar theory poses are very big. It is e.g. absolutely impossible to think of AUX+VP as a single constituent or nucleus, as Tesnière [1959] or Järvinen & Tapanainen [1997] think (and as is semantically appropriate), and only an ever more

complex system of transformations can guarantee that semantically related sentences (active-passive, topicalisation, there-insertion etc.) have the same D-structure and remain context-free.

Furthermore we have seen that HPSG is in some ways a dependency theory. In our discussion of *Bare Phrase Structure* in the Minimalist Program in 2.3.12 we will see that the notion of head becomes even more central in recent developments of GB.

2.3.3 Government

Government, a relatively complex constituent relation in GB which is e.g. needed to assign Case, is available in dependency in a much simpler and more basic way. Government in GB and government in dependency are equivalent. "It is interesting to see how hard it is to define the notion 'governor of [word]' when dependency relations as such are not available." (Hudson 1990: 113). The result of Chomsky's complex definitions and extensions [Chomsky 1986: 162] is, however, that "government is exactly equivalent to non-adjunct dependency in [Word Grammar]" (Hudson 1990: 113). [Covington 1992] provides the same answer, at least for lexical items as governors:

> ... it is clear that if only lexical items can govern, then the definition of government is:
>
> **A governs B iff B is an immediate dependent of A.**
>
> One can hardly ask for this to be simpler.
>
> (Covington 1992: 4)

In GB, on the other hand, government is not only complex and seemingly arbitrary, it cannot even be satisfactorily defined:

> A ... GB problem is the concept of government, which, despite having considerable empirical motivation, is none the less an 'arbitrary syntactic relation' (Lasnik, 1993, p.3). Moreover, there are many different notions of government, some working better than others for certain phenomena; unfortunately, no version is perfect for all purposes.
>
> (Cook & Newson 1996: 315)

Cook & Newson [ibid.] give the following example. If we assume one of the most "simple" GB government definitions like

> α governs β if and only if
>
> (1) α is a governor (e.g. N,V,P,A, etc.)
>
> (2) α and β mutually c-command each other
>
> (3) if α governs β, then α governs the specifier of β
>
> (Cook & Newson: ibid.)

too many relationships are allowed to govern. E.g. under this definition a verb is allowed to govern the specifier of its CP complement, therefore it can wrongly assign Case to a WH-element which has moved there. But since the original WH-position is also Case-marked, this means that this WH-element receives at least *two* Cases. Due to

the problems with the government definition, the minimalist program abandons government.

2.3.4 Grammatical Features

While PSG rewrite rules are the basics of constituency, agreement of grammatical features are the basics of dependency. If the grammar requires two word forms to share the same grammatical features, we have a dependency relation between them, although the direction of the dependency is not yet established. One of the characteristics of unification is that the direction of the variable binding is unspecified. **Dependency is unification plus specification of direction**. The fact that the direction needs to be specified can be seen as a disadvantage. It has the advantage, however, that we can expect proof trees to become much smaller. Note, however, that Prolog permits the direction of the unification to be pre-definded, so that similar speed gains can be achieved.

2.3.5 Agreement

Agreement, by means of shared grammatical features as seen above, is a classical case for dependency, so much so that Mel'čuk [1988] claims constituency can only have evolved from a language with an extremely simple morphology such as English:

> Even though this may sound a bit too Whorfian, I am fairly sure that PS-syntax could not have been invented and developed by a native speaker of Latin or Russian. Such languages feature an incredibly flexible (but far from arbitrary) word order and very rich systems of morphological markings; word arrangements and inflectional affixes are obviously contingent here upon relations between word-forms rather than upon constituency. To promote PS-representation in syntax, one has to be under the overall influence of English, with its rigid word order and almost total lack of syntactically driven morphology. (Mel'čuk 1988: 4)

2.3.6 Unbounded Dependencies

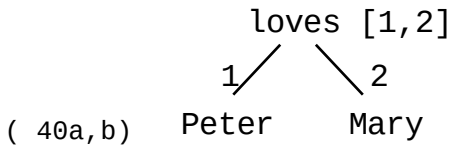
The treatment of unbounded dependencies is a home victory for dependency grammar. Word order is irrelevant for dependency, which means that it does not matter if a constituent is in its usual place or fronted.

Pollard & Sag [1994:157] group unbounded dependencies into two classes. In the first class, in GB terms, "there is an overt constituent in a nonargument position ... that can be thought of as strongly associated with (or filling) the gap or trace" (ibid.). In this class we find *topicalisations, WH-questions, WH-relative clauses* and *pseudo-cleft* constructions. For this class, the dependency structure expresses both the unmarked sentence and its version with 'movement'. As an example, the **topicalized** sentence

(49b) Peter$_i$, Mary loves _$_i$

receives the same dependency structure as the unmarked

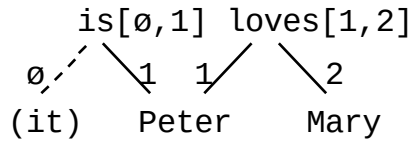
(49a) Mary loves Peter



(40a,b)

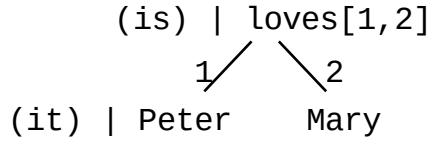
In the second class "there is no overt filler in a nonargument position, instead there is a constituent in an argument position that is – loosely speaking – interpreted as coreferential with the trace" (ibid.). In the second class we find *purpose infinitive*, *tough movement*, *relative clause* and *it-cleft* constructions. Here, the dependency structure for the sentence with 'movement' has to look different. For the **it-cleft** version of (49),

(49c) It is Peter$_i$ Mary loves $\__i$



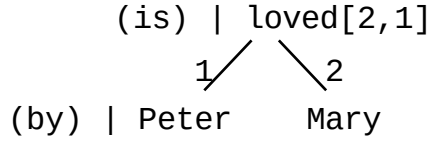
(49c)

ø marks the expletive *it*. The essential thing is that *Peter* has a double head in this representation. Usually, dependents are only allowed to have one head in dependency grammar. Hudson [1990:117] has suggested, however, that certain well-defined exceptions should be allowed. Alternatively, we may suggest that not only the pronoun *it* is a dummy, but also the verb *is* has dummy character in expressing only the topicalisation and should not get a nucleus of its own. If represented at all, it should a part of the main verb nucleus. If we assume so, then both classes can be dealt with in the same way.



(49c)

The desire to use the same uniform representation for active and **passive** sentences actually favours this last option, i.e. not to represent auxiliary or dummy verbs in a separate nucleus, but to represent them as a part of the main verb nucleus.



(49d)

2.3.7 Binding Theory

If we want to use a dependency grammar with binding for e.g. anaphors or pronouns we need to allow several heads for one dependent in such cases, as outlined above. As we will see in 5.2, Link Grammar uses links for the binding of relative pronouns. This introduces circular dependencies which can no longer be expressed in stemmas. For Tesnière, anaphora are in a strange position between being function words in the lexicon and syntactic nuclei in the text. Mel'čuk [1988: 107] recognizes the need for

binding links, but he doubts if they are directed. Anyway, binding is not a well-researched part of dependency theory and needs additions to the original framework.

2.3.8 Functional Roles and Theta Theory

GB theory is still caught in a configurational cage, even though to a lesser degree after the inclusion of theta theory.

2.3.8.1 Configurationality

In early GB days, a subject was simply a NP dominated by S, and an object was a NP dominated by VP. Nowadays, the definitions are more complex, but they are still configurationally based, i.e. still the position inside the D-structure, defines an element's θ-role:

> Every referring NP must bear exactly one θ-role. If we assume that once an NP receives a θ-role, it retains it, even when moved, then it is sufficient to say that the revised θ-criterion holds at all levels (that is, at D-structure and at S-structure). If the θ-criterion holds at D-structure, then every referring NP must originate in a θ-position.
>
> (Cowper 1992: 83)

2.3.8.2 Lexical-functional Grammar (LFG)

Such a configurational definition is not sufficient to accommodate the rich variety of languages, especially not in the so-called non-configurational languages (cf. [Horrocks 1987:231], [Sells 1987: 153-154]). One of the key ideas of LFG is to have a more powerful and flexible mapping between syntactic and functional structures at the linguist's disposal. LFG requires functional relations to be primitives. LFG continues to use constituent (c-) structure much alike GB theory (even if X-bar theory is not a stipulation in LFG it is usually adopted), but it additionally uses a functional (f-) structure.

> Bresnan's **lexical-functional grammar** [Bresnan 1982] retains the PS-tree for representing the surface-syntactic structure of a sentence but introduces additional machinery to explicitly express grammatical relations: the so-called "functional structure", which is essentially a specification of dependency relations over the set of lexemes of the sentence under description.
>
> (Mel'čuk 1988: 7)

In this sense, LFG is the natural answer to the question if constituency and dependency are alternatives or complementary, as we shall see in 2.4.6. For an introduction to LFG, refer to [Bresnan 1982] or [Dalrymple et al. 1995].

In LFG, mapping between syntactical and functional descriptions follows from *annotated* PSG rules, e.g. for the assignation of functional subject to the NP immediately dominated by S:

$$\underset{(50)}{\quad} \quad S \rightarrow \underset{(\uparrow SUBJ)=\downarrow}{NP} \quad \underset{\uparrow=\downarrow}{VP}$$

The functional structures built up from these mappings are similar to dependency structures. Let us consider the example

(51)The elephant was worshipped by the child.

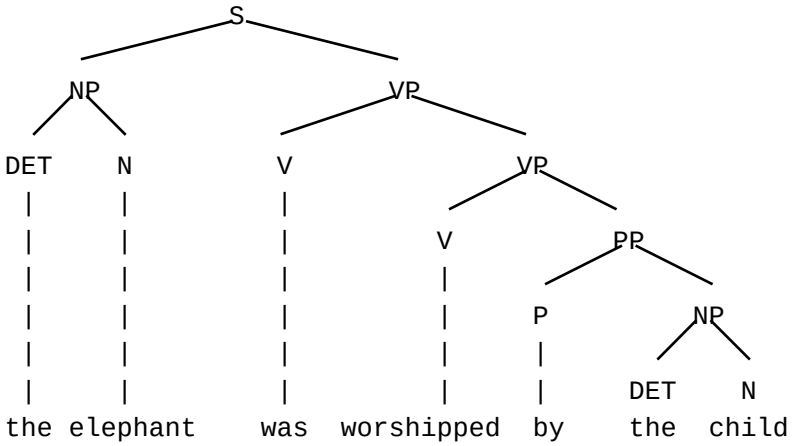
When parsing with the following PSG rules

(50 *relevant PSG-rules*):

```
(A)  S     →   NP            VP
               (↑SUBJ)=↓      ↑=↓

(C)  VP    →   V             VP
               ↑=↓          (↑VCOMP)=↓
                           (↑SUBJ)=(↑VCOMP SUBJ)

(B)  VP    →   V             (NP)        (PP)
               ↑=↓          (↑OBJ)=↓  (↓PCASE)=BY
                                        (↑BY OBJ)=↓
(D)  PP    →   P             NP
               ↑=↓          ↑=↓
```

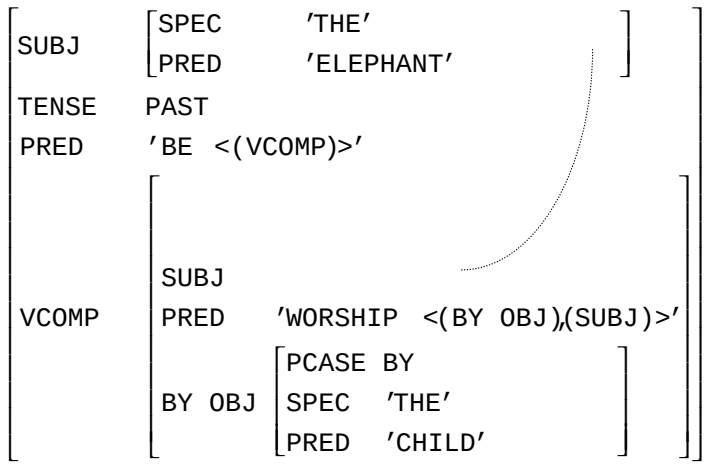
we get this constituent structure:

(50 *c-structure*):



The mapping coded in the functional annotations builds up the following f-structure (for more detailed examples, refer to [Schneider 1996]):

(50 *f-structure*):



F-structures, obtained in LFG or from a functional dependency parser, are not semantic structures, but they provide a suitable intermediate step:

> One of the principal advantages inherent in the architecture of LFG, noted in its earliest formulations, is the clear suitability of the f-structure as a syntactic basis for semantic analysis. F-structures provide a uniform format for stating the syntactic information that is most important for semantic composition.
>
> (Dalrymple et al. 1995: 275)

2.3.8.3 Functionalism

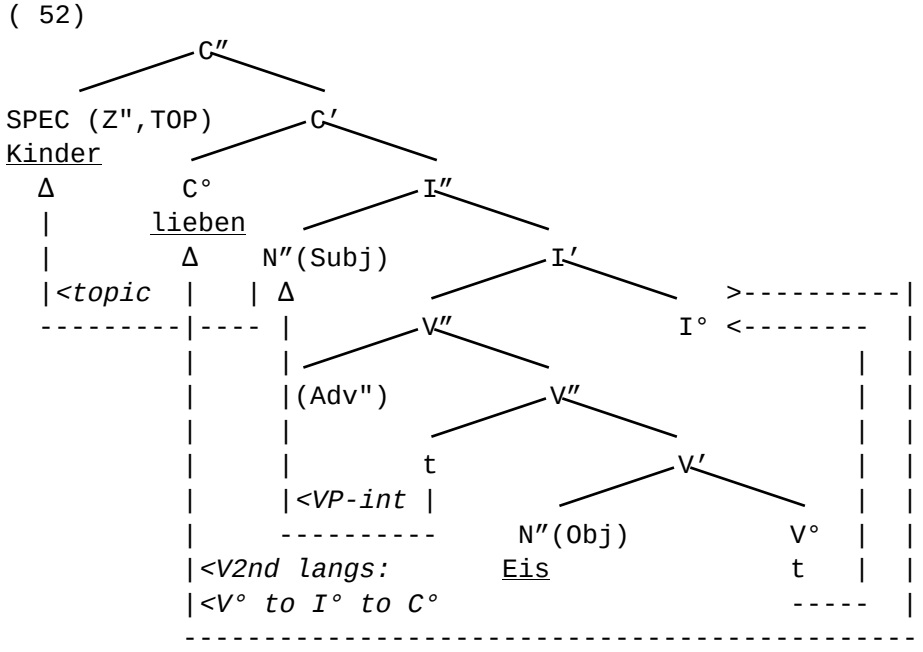
Except in non-configurational languages, it is possible (even though linguistically less elegant) to derive a functional structure from the configurational GB D-structure. Perhaps at a first sight it is not even obvious why dependency is inherently more functional than constituency. Let us consider some of the reasons:

- **Free word order**: Free word order facilitates functionalism. Otherwise it is very difficult to assign the same syntactic structure to active and passive sentences, dative movement, there-insertion etc. GB has to employ a complex system of transformations and postulate D-structure for the same effect. But free word-order allows us to parse directly for functional relations, this is more efficient and constitutes a main reason why dependency parsers are faster than constituency parsers.
- **Tesnière's translations**, or a similar system for opening up valencies to other structures (cf. 2.1.4.2) with similar grammatical functions is a prerequisite for functionalism.
- **Non-projectivity or context-sensitivity**: Some structures, e.g. the stranded quantifiers in (55) (cf. 2.3.9.3), many unbounded dependencies (cf.

Tapanainen & Järvinen [1997] for an example) or sentence-final verbal particles in phrasal verbs (cf. 3.4.1) are only context-free because of transformations or because of functional nodes.

- **Scattered constituents**: Even though most sentences would remain context-free, what makes up a functional constituent or nucleus can be scattered over long distances if we do not postulate movements and empty categories.[4] From a functional perspective, e.g. AUX+V form the verb constituent, but (e.g. in verb second languages like English or German) there may be many intervening constituents which have to be bridged by the transformation that moves the inflected part of the verb nucleus up to the INFL node in the S-structure when generating or back from INFL to V^0 in the D-structure when parsing. HPSG uses the term *argument composition* (cf. Borsley [1996: 89]) for this process of collecting semantic units from different constituency nodes. I will implement argument composition in 3.3.3.4
- **Transformational complexity**: What follows is a fairly standard way, assuming VP-internal subject hypothesis (cf. 2.3.9), to analyze the German three-word sentence (52) *Kinder lieben Eis*. If we want to respect subordinate clauses or possibly intervening constituents appropriately (cf. Vikner [1995] and Schneider [1996]), no simpler analysis is possible – we need four transformations for three words. Then, at D-level, we can finally abstract a functional structure.

[4] Transformations are a machinery to deal with the fact that constituents *are* scattered at S-structure. An advantage of non-projective parsing is that such an counter-intuitive machinery is not necessary. In GB terminology, the D-structure constituents are picked up across all constituents intervening at S-structure. Since non-projectivity contradicts a maxim of constituency, dependency is naturally more suitable for non-projective parsing.



- The complex mapping problems between c- and f-structure which are a focus of LFG research have shown that configuriationality is *sui generis* inappropriate to derive functional relations.

Refer to 2.3.6 for an example on how a functional dependency parser can assign the same structure to topicalized, cleft or passive sentences as to the corresponding non-marked sentence.

2.3.8.4 Critique of Thematic Roles

Already for Tesnière, the functional orientation of his theory was essential. He distinguishes between (subcategorized) *actants* and (free) *circonstants*. He distinguishes three 'theta' roles for actants (and uses a fourth one for control), but his system is criticized for being as simplistic as simple:

> En ne distinguant que fonctions actantielles et fonctions circonstantielles dans l'unité verbale, Tesnière offre un schéma simple, mais également simpliste. Les trois actants qu'il reconnaît (le quatrième n'apparaissant que dans les structures causatives) sont définis de deux manières: formelle et sémantique. Cette dernière est reprise sans distance critique de la grammaire traditionelle: le sujet fait l'action, l'objet la subit et le tiers actants est l'entité au bénéfice ou au détriment de laquelle l'action se fait. Il y a tellement d'objections qui viennent à l'esprit quand on réflechit sur cette manière de procéder ...
>
> (Feuillet 1996: 129-30)

While LFG is not entirely Chomskyan, the Chomskyan development of theta-roles points to a similar direction, by including typical dependency relations into constituency. The introduction of theta theory was a clear step towards functionalism. Cowper [1992: 61-4] explains that the use of both the lexical strict subcategorisation

and the theta grid is highly redundant, and one of them can be eliminated, just as PS rules could be eliminated by X-bar grammar and direct subcategorisation. Based on functional arguments similar to those in 2.1.4.1, Cowper shows that eliminating direct subcategorisation in favour of the theta grid allows us to make elegant generalizations.

> Since it does not seem plausible to eliminate the θ-grid in favor of strict subcategorisation statements, and since the converse is apparently quite feasible, we will assume that strict subcategorisation statements are to be eliminated from the grammar and the θ-grid retained.
>
> (Cowper 1992: 64)

Now the universal constraints of X-bar, language specific X-bar parameters and a strong projecting lexicon which is only constrained by every entry's theta grid starts to generate and will seriously overgenerate:

First, we have seen in 2.1.4.1 that if we use functional categories in a generating system it will seriously overgenerate.

Secondly, Cowper [ibid.] herself gives a word of warning directly following the above quotation: "Recall, however, that we saw ... that strict subcategorisation statements did not make phrase structure rules completely unnecessary, and that much work in this area remains to be done." (ibid.)

Thirdly we have seen in 2.2.3.1.3 that Case theory is not uniform and controversial. Bouchard [1995:41-2] discusses that theta theory is inadequate. It only manages to cover stereotypical situations and fails in many cases, perhaps the majority of cases.

It is to be hoped that - language specific - grammatical terms are more robust than theta roles. Language-specific means that for Germanic and Romance languages grammatical terms like subject or object can be relied on as primitives, while in other languages we have to employ different primitives like e.g. topic and focus or yet other terms.

2.3.8.5 Layers, Mappings, Complexities

Feuillet [1996] (cf. 2.1.4) stresses that Tesnière confounds categories (in the word-class sense), grammatical relational functions and grammatical case, terms applying to different levels or layers of analysis. On the other hand he also frankly admits that we often either find a simple 1:1 correspondence between these levels – which renders any distinction redundant – or that we sometimes face very intricate and complex mapping difficulties:

> Se pose ici la question de l'intrusion de la sémantique dans la définition des fonctions syntaxiques. Le mélange de plusieurs critères [de Tesnière] fait penser à la manière dont on définit les parties du discours: un ensemble hétérogène de traits qui appartiennent à différents niveaux d'analyse. On se sert de la morphologie (le sujet est au nominatif dans les langues à déclinaison), de la position (en français, le sujet précède l'objet), et du sens (l'ergatif est le sujet, car

> il fait l'action). Dans les langues ergatives, il est vrai qu'il y a coïncidence entre marquage casuel et rôle sémantique: l'absolutif est le patient et l'ergatif l'agent. Mais ce schéma n'est pas transposable aux langues accusatives où le sujet peut être agent à l'actif, mais patient au passif. Par conséquant, les fonctions traditionelles valables pour un certain type de langues sont inadaptées à d'autres. Quelle que soit l'ingéniosité dont font preuve nombre de linguistes, il n'y pas de définition universelle du sujet, et l'on peut même se demander s'il est utile de parler de sujet dans certaines langues.
>
> (Feuillet 1996: 131)

2.3.9 VP-internal Subject Hypothesis

In 2.3.2.5.3, we saw that VP is now generally assumed to be the head in a subject NP+VP construction. Still, the fact that it is actually only INFL which dominates the subject rather than VP directly remains unsatisfactory. Here we see a more recent development in X-bar theory, closer to the dependency spirit, and as such support of the dependency grammar maxim of *verb supremacy*.

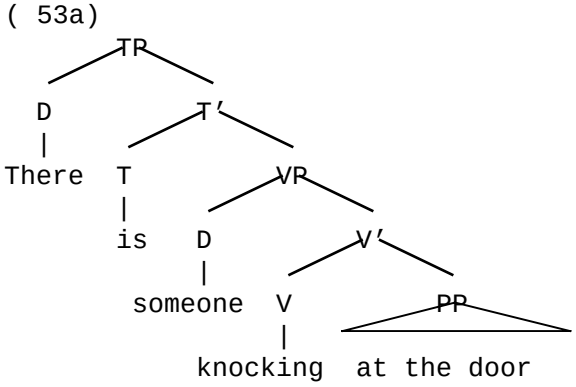
The VP-internal subject hypothesis is the X-bar answer in the search for evidence of verb supremacy, i.e. that the subject syntactically depends on the verb rather than vice versa.

> The subject of the sentence can now be brought into this framework. The VP as presented so far has a specifier position that has been unfilled. According to current theory, this is where the subject belongs in the D-structure of the sentence: the subject is the specifier of the VP. This is a different location from that used in earlier versions of principles and parameter theory, or indeed in earlier Chomskyan models of syntax ... (Cook and Newson 1996: 146)

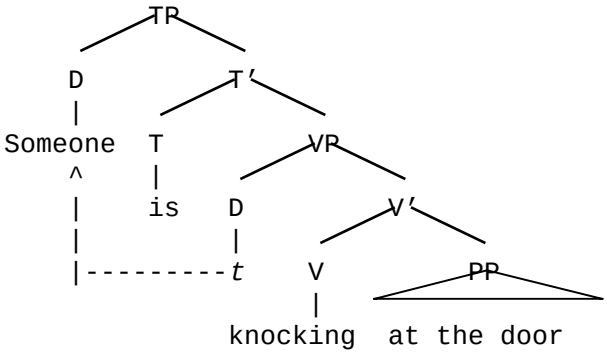
Radford [1997: 315-24] gives several arguments in support of VP-internal subject:

2.3.9.1 Expletive *there*:

The two following structures receive the same D-structure under the VP-internal subject analysis, expressing the semantic near-synonymity. In (53b), the subject *someone* also originates in the same VP-specifier position, but moves up ('<--') to the unoccupied TP-specifier position.



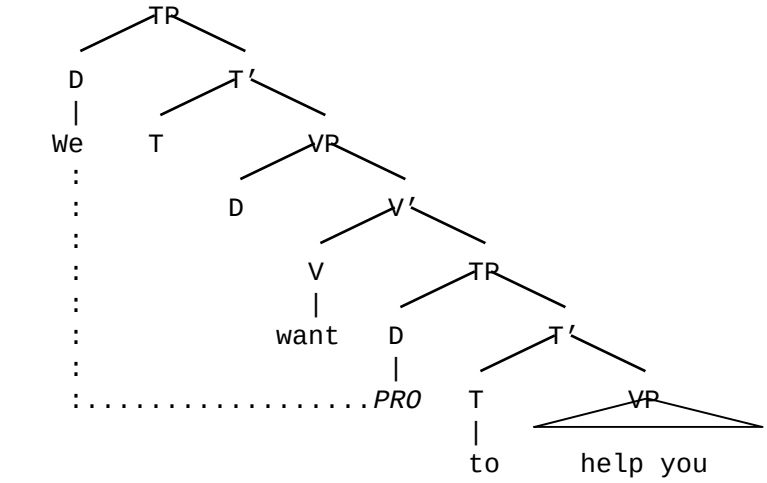
(53b)



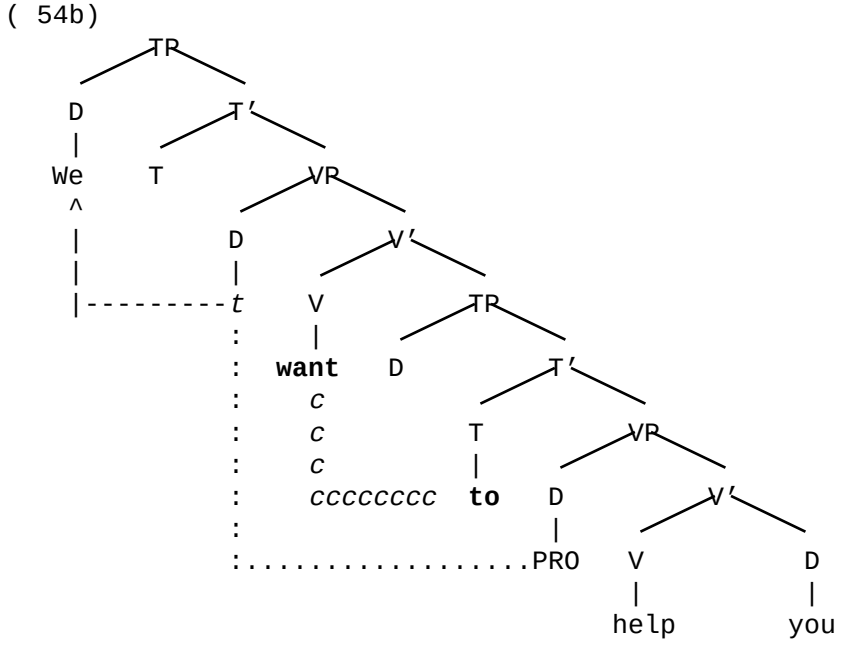
2.3.9.2 Cliticization (*wanna*-contraction):

Perhaps the only piece of evidence for the GB assumption that there are empty categories is *cliticization* of e.g. *they have* to *they've* and *want to wanna* which is blocked by intervening empty categories [Radford 1997:269]. In the control structure (54a) below, however, the intervening *PRO* (the subject of the subordinate clause, which has an anaphoric link (':') to the main clause subject *we*) should block such a cliticization, although it is acceptable. The traditional GB analysis makes wrong predictions:

(54a)



But under the VP-internal subject hypothesis cliticization ('*c*') complies to the analysis (54b):

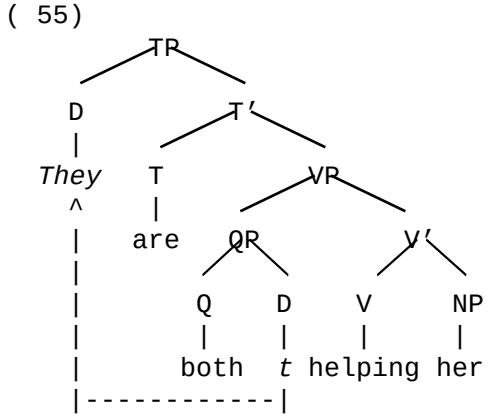


2.3.9.3 Stranded quantifiers:

In sentences like

(55) **They** are *both* helping her

the quantifier *both* is separated from the subject *they*. The VP-internal subject hypothesis allows the following analysis:



2.3.9.4 Semantic argument-predicate structure:

Perhaps even more convincing, “there is also strong semantic evidence in favour of the hypothesis” (Radford 1997: 324), because a VP-internal subject is much closer to semantic argument-predicate structures. GB moves closer to semantics - and also to dependency.

2.3.9.5 Conclusion

The VP-internal hypothesis still faces some problems with Case assignment [Radford 1997:329-333], but it elegantly assigns the same D-structures to sentences with or without expletive *it* or *there*, to active and passive sentences [Radford 1997:341-4], and permits an explanation for stranded quantifiers. Especially due to its semantic appeal it is rapidly becoming the standard analysis. As often in recent developments in GB, a clear movement towards the spirit of dependency theory is evident. The verb does not only have a functional head over the noun, it directly becomes head of the noun.

The debate whether the subject or the verb is more fundamental will go on despite the provisional decision for the verb. Being the most fundamental ontological linguistic question, it will continue to attract many linguists, even if this question may be undecidable.

2.3.10 DP hypothesis

We saw in 2.3.2.5.4 that it is still undecided if a determiner should be a functional head to its noun or a grammatical marker dependent on the noun. In GB, determiners are traditionally located in the SPEC position of the NP. In more recent versions of GB, however, determiners have their own phrase (so-called DPs) in which the NP is only a complement (cf. [Cook and Newson 1996: 183-5]). In this assumption, DET is the head, with N as a more local sub-head.

While the DP hypothesis constitutes a step away from Tesnière's original conception of determiners as function words which are part of the noun nucleus, it is a step closer to semantics. It may also serve to solve the problem of the double role of anaphora as simultaneously function word and nucleus (cf. 2.1.1.1).

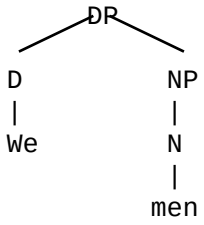
The DP hypothesis has been suggested for the following reason:

- Non-heads in X-bar syntax are usually required to be maximal projections. Determiners were an exception in the traditional view. Now they get their maximal projection. Note, however, that this is a theory-internal argument, a hypothesis that is solely based on other hypotheses. In 1.3, Yngve [1996] warns us of theory-internal arguments.
- NPs and Sentences have much in common. E.g. Engel [1994] states that they are "im besonderen Mass in der Lage, Sachverhaltsvorstellungen in kompakte Form zu bringen, das heisst grammatisch gesehen: Sätze in übersichtlich abgewandelter Form wiederzugeben ... Es kann nicht wundernehmen, dass sich eine enge Verwandtschaft zwischen der Strukur von Sätzen und der von Nominalphrasen erkennen lässt: Sätze sind nach

bekannten und greifbaren Regeln in Nominalphrasen überführbar." Like IP for S, NPs receive a functional head with DP.

- Structures like (56) *We men* or *I Claudius* now find a plausible analysis:

(56)



- The fact that most pro-forms appear without articles can easily be explained by postulating that they are pro-DPs [Radford 1997: 96]
- Like in the VP-internal subject hypothesis, the semantic argument (although abandoned by Jung [1995] plays an essential role. Semantically, nouns depend on their determiners.

2.3.11 Functional Heads

2.3.11.1 Covert Functional Preposition Hypothesis

I have not found any mention of what should be called *the covert functional preposition hypothesis* in the literature, but it may exist in some form.

Just above we saw that NPs and sentences have much in common. According to current theory, they both have functional heads, IP for S and DP for NP. The category *S,* however, no longer exists. Following the *VP-internal subject hypothesis* (cf. 2.3.9) all the content words originate from within V" at D-structure. V" comes closest to the former S category. V" has two functional heads: I(NFL) on the first level and C(OMP) on the farther second level (farther because it is head of I(NFL) rather than the verb itself). According to the *DP hypothesis* (cf. 2.3.10), NPs have a DP functional head. In 2.3.2.5.2 we saw that P can be seen as a functional head of NP, or rather of DP.

Therefore, both NP and VP can be seen as having very similar and symmetrical functional heads on two levels: On the first level, quantification for nouns and inflection for verbs. This expresses a simple ontological truth. Real-world objects (nouns) are instantiated by quantification, events (verbs) are instantiated by setting them into a time frame. On the second level, P for DPs and C for VPs.

As is typical for functional heads, all of them can remain empty. On the frist level, noun determiners can be syntactically absent (in generic reference they are even semantically absent) and verb inflections are absent in non-finite clauses. On the second level, e.g. subject or nouns have empty prepositions and main clause verbs have empty functional complementizers.

This suggestion implies that every NP is a PP, in case there is no overt preposition the P slot is simply empty - such is the state of functional heads. But functional heads ideally mark some grammatical relation, contain grammatical features. If P is overt, Deep Case is marked on it. Why should Deep Case be marked on the noun for covert Ps? After all, tense is also marked on I even if it is empty. In such a case, V features *percolate* to I, (cf. Radford 1997: 138-44). Why should Case not percolate to P in exactly the same manner? This would finally allow us to suggest the same D-structure for

(57a) She gave Peter the book.
(b) She gave the book *to* Peter.

Especially, it permits the formulation of a uniform topic-focus theory. In analogy to (57b) in which P-overt *to Peter* is more topical than in (57a), I-overt *does read* in (58b) is also more topical.

(58a) She reads.
(b) She *does* read.

2.3.11.2 Functional Heads In Dependency

The above suggestion is borrowed from dependency grammar, where (57a) and (57b) have the same representation (cf. 4.2.4). The fact that P and Case are the same functional head is evident to most speakers of Finnish or Estonian. The other of the above functional heads rather originate in GB theory and have found their way to dependency:

- I(NFL)-AUX: In 2.2.3.1.4 we have seen that Tesnière treats I-AUX as a part of the verb nucleus. Partly due to the problems mentioned there and in 2.2.2.2.1, 2.3.2.5.5 has shown that an INFL head is now generally accepted in dependency theory.
- COMP: In 2.3.2.5.6 we have seen that this functional head is also generally accepted.
- DET: After the X-bar DP hypothesis (cf. 2.3.10) dependency theory will probably follow X-bar. Currently, most authors still have the noun as head. But also in GB, many authors still do not use DPs.

I conclude that current GB and dependency grammars use functional heads in almost the same way, except for covert PPs. I would not be surprised, however, that they will be used sooner or later in GB.

2.3.12 The Minimalist Programme: Bare Phrase Structure

There is a counterforce in current GB theory, however, which tries to get rid of many of the baroque inventions of the theory. It is called the minimalist programme, and it tries to use principles of economy for describing grammar. No redundancy will be accepted, everything that is not really necessary should be abandoned. This paper cannot be an introduction to the minimalist programme – refer rather to [Cook & Newson 1996: 311-44] or [Radford 1997]. I want to focus on only one aspect of the minimalist programme; on *bare phrase structure*. In it, even X-bar grammar, which had become more and more central to GB is questioned.

I mentioned in 1.3 that X-bar theory cannot be accounted for if we follow standard scientific assumptions. Questioning and challenging it is therefore a long-expected purification anyway.

2.3.12.1 The *Merge* Operation

The minimalist programme is focused on the process of building up syntactical structures.

> In GB, D-structure was presented as a complete structure and not much was said about the internal process of how it was formed ... However, in getting rid of D- and S-structure, minimalism places more emphasis on the internal workings of the structure formation process ... part of this process involves building individual trees for lexical items (...) and then combining these at some point to form a larger tree. How and when they [=individual trees] are combined will have repercussions on the eventual SD [=syntactic description] formed
>
> (Cook & Newson 1996: 323)

The operation that combines lexical items or partial trees with others is called *Merge* by Chomsky, "an operation that forms larger units out of those already constructed" (Chomsky 1995: 396). Aiming at minimalism, *Merge* is always a binary relation, i.e. it combines two partial trees to form something new. "Applied to two objects α and β, Merge forms the new object γ. What is γ? γ must be constituted somehow from the two items α and β ..." (ibid.) Cook & Newson [1996] put Chomsky's subsequent answer as follows:

> The minimal assumption is that the combination of two elements forms a set consisting of these two elements ... However, this is obviously too minimal as it misses the important observation that combined sets of lexical items have special features which they primarily take from one of their members: in other words, each 'phrase' has a 'head' that determines the properties of that 'phrase'. Thus, what is formed by the combination of two elements has to also include information about which of the two provides the properties for the combined set.
>
> (Cook & Newson 1996: 338-9)

2.3.12.2 Head Label

Chomsky suggests to give a label with the name of the head the tree which is formed by a combination under Merge. Using set theory notation, the combined tree or set thus consists of the label, and a set of the trees combined: The combination of α and β, if α is the head, is thus {α,{α, β}}.

> γ must ... at least (and we assume at most) be of the form {δ,{α, β}}, where δ identifies the relevant properties of γ, call δ the *label* of γ. The label must be constructed from the two constituents α and β. ... [t]he label δ is either α or β; one or the other *projects* and is the *head* of γ. If α projects, then γ = {α,{α, β}}.
>
> (Chomsky 1995: 396-7)

For the 'NP' the dog we therefore get (59a) under the DP hypothesis, or (59b) for the traditional analysis with N as head:

Except for the "repetition" of the head in Bare Phrase Structure, the corresponding Dependency stemmata look exactly the same:

The decision which of the two arguments of the Merge operation is head is based on lexical properties, just like lexical properties decide on the direction of a dependency. The label of the Merge node is therefore called *projection*.

2.3.12.3 X-bar levels

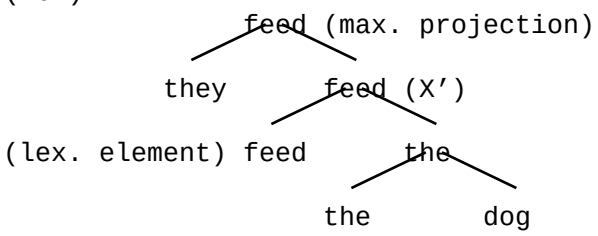
So far we have only seen lexical entries and *projections,* which correspond to X-bar maximal projections, i.e. phrases. We may ask ourselves the justified question where the intermediate X' category can fit in Bare Phrase Structure:

> But how are the basic observations of X-bar theory captured by this method? Since the early days of GB theory it has been observed that the only active elements in a derivation are heads and maximal projections: these are the only elements that are involved with syntactic processes such as movement and these are the elements which enter into semantic relations, denoting predicates and arguments. If the X' category exists at all, and there is empirical evidence for this, it should be a derived category rather than being inserted as an element in its own right. (Cook & Newson 1996: 340)

This is a radical departure from traditional X-bar theory. The intermediate X' level is now only a derived category. How can X' be derived ? Chomsky's answer [Cook & Newson 1996: 340-1] is naturally that the derived X' is a projection which is

not maximal because it projects further. An example [ibid.] is (61) *They feed the dog* analyzed under the VP-internal subject hypothesis (cf. 2.3.9):

(61)



At some stage in the derivation, when the lexical element *feed* and the local tree *the dog* are Merge-d, the label *feed* of this Merge operation is a maximal projection. But because feed is also the head and thus label of the next Merge, it projects further, the new *feed* shown at the top of SD (61) is now the maximal projection, the *feed* one level below can be derived as X'.

Dependency recognizes higher nodes as a derived concept (cf. 2.3.13). A phrase, called nucleus by Tesnière, consists of a head and its dependents and recursively all their dependents.

Intermediate categories are one of the last, and most crucial differences between dependency and constituency. For Covington [1994], as described in 2.4.5, it is even the only difference between them. Although there are additional differences, at least if we consider the original Tesnière [1959] conception, there are many similarities, as we will see in 2.4.6, X' is now a derived category both in Bare Phrase Structure and in dependency grammar.

2.3.12.4 The Minimalist Programme as a Dependency Grammar

We have seen above that X' is a derived category, as in dependency. But in dependency, also maximal projections, nuclei, are a derived concept. So far we may get the impression that maximal projections (X") and lexical heads (X^0)[5] are basic concepts. This is in fact no longer true either:

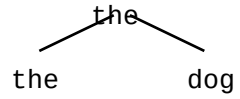
> First, the X' schema stipulates that XP must dominate X' and X' must dominate X. Second, it stipulates that the complement is adjoined to the head and that the specifier is adjoined to the result. Suppose that we simply abandon the notion that phrasal categories are distinguished by their level; on such a view heads and maximal projections are all of the same category, and the are distinguished by where they appear in the structure. An X that does not branch is a head; an X that does not have an X above it and that has a path of nodes only of category X down to a head of category X is a maximal projection.

[5] Note that the GB *head* notion is slightly different from the one in dependency. Terminal nodes which project to an XP are called *head* . It is unusual in dependency terms to think of an X^0, which does not govern anything, as a head. But if the distinction between X^0 and X" is given up, as in the following quotation, the term *head* has the same meaning in GB and dependency.

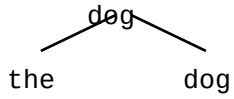
(Culicover 1997: 355-6)

Now even X" becomes a derived concept, and maximal projections and lexical entries are basically the same. But if they are the same, why should they be repeated again below the last projection level ? E.g., assuming the DP hypothesis in (a) but not in (b), why write

(59a)

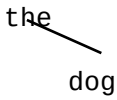


(b)

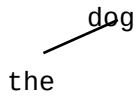


instead of the simpler

(60a)



(b)



In search of minimalism, Chomsky's economy principle should in fact get rid of such redundancies:

> **Economy principle**: both derivations and representations are subject to a certain form of "least effort" condition and are required to be minimal in a fairly well-defined sense, with no superfluous steps in derivations and no superfluous symbols in representations.

(Chomsky 1991: 69)

Now dependency stemmata and Bare Phrase Structure SDs really look identical, perhaps they even are. One difference that remains is that Merge is a binary operator, i.e. in dependency terms, each head can only have two dependents, but under head repetition one nucleus or bar-level lower additional nuclei can attach. One might want to argue that such a head repetition is again not in the spirit of the economy principle, but two arguments speak against this. (1) Merge is defined as a binary operation because of the economy principle , and (2) this head-repetition allows to configurationally derive intermediate (X') nodes.

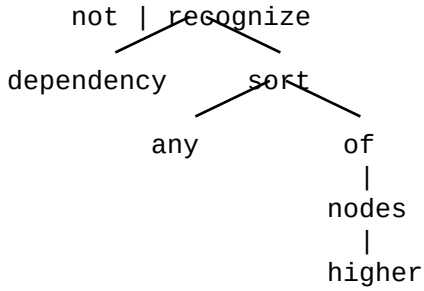
Note that my above suggestion not to repeat heads at X^0 does not violate the rule that Merge is binary, although it looks like a unary operation in these cases. But how could (assuming DP hypothesis) {dog} merge with itself in order to form {the{the,dog}} or rather, under my suggestion {the{Default-HEAD-SELF,dog} ? The head is available at LF, it is simply cancelled at X^0 level for the SD, but may always be uniformly derived again when needed.

But the minimalist programme is only in its infancy, let us step back to more traditional GB, although I will make a few remarks about minimalism in the following.

2.3.13 Constituents and Nuclei

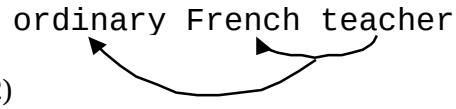
Because dependency does not recognize any sort of higher nodes, as we saw in 2.2.6.1.1, we may have the impression that dependency can only operate on a word-level and cannot use constituents of any kind. This is not true, however, for two reasons. **First**, as Tesnière [1959] stresses, the fundamental element of his theory is not the word, but the *nucleus*. A nucleus may consist of single words, it may also consist of an idiomatic multi-word expression like *a priori* or *Head Driven Phrase Structure Grammar*. But nuclei may themselves consist of other nuclei. E.g. Covington (1994: 2) reminds us that "[d]ependency grammar still recognizes constituents, but they are a defined rather than a basic concept. The usual definition is that a constituent consists of any word plus all its dependents, their dependents, and so on recursively. (Tesnière calls such a constituent a NŒUD.)"

(62) Dependency does not recognize any sort of higher nodes.



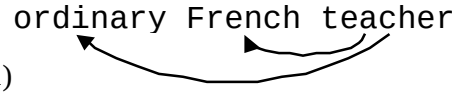
The Dependency representation (62) is one of the possible ways to analyze this sentence. While the single word *higher* is the nucleus dependent on *nodes*, *sort* is head of the complex nucleus *of higher nodes* etc.

This works just like in constituency. The only place where dependency is possibly less powerful than constituency is when intermediate categories are required, i.e. X', as we have seen in 2.3.2.4.2. It is not possible to make a difference between adjuncts (attached under X'-recursion) and complements (attached under expansion from X' to X), because



(42)

is impossible and in



(41)

both dependents directly depend on the noun.

What we would need in order to analyze (42) as in constituency would be some sort of "re-entrancy"[6]. I have suggested in 2.2.6.1.1 that a lexical "re-entrancy" by means of an LFG-style lexical rule which creates a lexical entry *French house* online could solve the problems. But if we consider sentences like

(63) I see Mary run.

there are three valid alternatives to conceive of its structure (semantic heads are in *italics*): (1) I see *Mary*, who happens to be running, (2) I see the act of *running* exerted by Mary (3) I see the indivisible act of *Mary-running.* If we opt for (3) we can hardly postulate a lexical re-entrancy for an obviously syntactic operation. Analysis (3) is a problem for any headed structure, because it is unclear which is the head and if Zwicky's [1985] criterion *(f.) Distributional equivalent* (cf. 2.3.2.4.5) should apply or if we want to abide to X-bar syntax then analysis (3) is ruled out in principle anyway. (3) is an impossible analysis for any headed structure, i.e. in most modern PSG grammars. Perhaps it is a question of a scope ambiguity, the scope of the verb being unclear[7]. If so, as suggested in 2.4.3.2.4, it is perhaps best to leave the scope underspecified in order not to treat a semantic problem at the syntactic, i.e. the wrong level.

If we prefer (3), then I would like to tentatively suggest two possible solutions:

One is, of course, that the minimalist programme also treats intermediate nodes as derivable, the derivation occurring along the same lines as those outlined for derivation of intermediate nodes in Covington [1994], as described in 2.4.5.

The other is that Tesnière's translations can close this gap. They are exactly syntactic re-entrancy rules, as we have seen in 2.1.4. The attentive read may wonder where the second of the two reasons announced at the beginning of this subchapter, which predict an impression that dependency could not use constituents, may have got lost. Here it is. **Secondly**, Tesnière's translations are a syntactic operation converting a string of several words to a function which might be expressed by one word or a nucleus. Engel [1996] stresses:

> *Die Dependenzgrammatik behandelt nurr die Relationen zwischen Wörtern, vermag also keine Zwischenkategorien zu efassen.*
>
> Eine solche Behauptung kann im Grunde nur aufstellen, wer bei der Lektüre der Eléments nicht bis zur Translation vorgedrungen ist. Denn es ist offensichtlich, dass jedes über dem Translationszeichen stehende Symbol die

[6] What I mean by "re-entrancy" is a device that joins a goup of words, i.e. a chain of dependents to a single undivisable unit, as if it was e.g. a single word from the lexicon (re-entered to the lexicon). Tesnière's translations or HPSG argument composition (cf. 2.3.2.5.5 and 3.3.3.4) are such devices.

[7] It is of course questionable if a verb can be said to have 'scope', as adjectives or articles do. As an intuitive concept, it makes sense, however. In the sentence *I saw the man with the telescope* the question whether the PP attaches to the object NP or theverb can be seen as a question of the scope of the verb: Does the verb have scope over a complex object NP or over both a simple object NP and a PP?

> syntaktische Funktion einer **Wortgruppe** bezeichnet, und es ist ebenso offensichtlich, dass der unter dem Translationszeichen stehende Translativ die interne Struktur einer Wortgruppe (Phrase) bestimmt.
>
> (Engel 1996: 55)

2.3.14 Transformations

All dependency grammars or dependency-related grammars (except for the Prague school approach) are monostratal, they do not know any transformations. If we consider a set of suggested transformations (I will take the transformations suggested in the Standard Theory, whose names still continue as metaphors even after the introduction of move α), then transformations in dependency theory are

- partly unnecessary because its free word order – like in Immediate Dominance (ID) rules, covers both S- and D-structure. Here we find all kinds of *topicalisations* belonging to the first class the structures showed in 2.3.6, as presented in [Pollard & Sag 1994: 157]. In addition, we also find *dative movement*, *extraposition*, *particle movement*, etc.
- partly unnecessary because in addition to free word order, (at least the original conception of) dependency allows crossing, so-called *non-projective* connections (cf. 2.4.7). An example for this are the QPs in 2.3.10, complex verb-chains in Dutch or Swiss-German, clause-final verbal particles in phrasal verbs and poetical, marked aberrations in many other languages.
- partly replaced by Tesnière's translation rules, sometimes in addition to the above. Translation rules are involved, if we want to use a dependency theory which employs them, in e.g. *genitive movement*, *nominalisation*, but also in *subordinated clauses* (which may occur at many positions in the surface sentence and should therefore be subject to movement from a single D-structure position) or *control* structures (cf. [Hudson 1997]).
- partly replaced by anaphoric links, which should be a part of every dependency theory. Here we find *relativisation* and *reflexivisation*. .If we want to express dummy-*it* or -*there* in the syntactic structure (cf. 2.4.7), then we also find *cleft* and *pseudo-cleft* constructions here, or *tag formation*.

At this stage we can safely claim that any syntactic theory will have to use either *transformations* or *restricted non-projectivity* if it should recognize the functional relations between sentences with the same D-structure, e.g. *active/passive or non-marked/fronted*. I consider the recognition of these functional relations to be an important part of the linguistic adequacy of a formal grammar. I will tentatively revise this claim in 6.3.1, but this will only apply if we accept non-standard dependency structures, as they are used by topic-focus articulation (TFA) – see 6.2.

2.3.15 Lexicalism

Link Grammar and Word Grammar [Hudson 1984, 1990] (cf. 4.2.1) are admittedly extremely lexical. Although Tesnière defines not the word but the nucleus to be the most basic element, nuclei on the lexical or 'terminal node' level usually consist of only one word. We have seen in 2.3.2 that the lexical properties of the head determine the behaviour of a more complex construction, as e.g. the criterion "distributional equivalence" (cf. 2.3.2.4.5) clearly expresses.

Modern PSGs have developed fully towards adapting these dependency assumptions. Structural descriptions are projected up from the lexicon, with X-bar theory as the last structural constraint, which is even abandoned in Bare Phrase Structure. But because lexical items are the elementary symbols of dependency grammars, whereas for PSGs non-terminal nodes are as elementary, Rambow & Joshi [1994] are convinced that there are fundamental problems with a lexical approach for PSGs: "We will ... address the question how a phrase structure-based syntactic theory can be adapted to a lexical approach. It turns out that there are intrinsic, formal problems." (Rambow & Joshi 1994: 2)

As for lexicalism, constituency and dependency strive to take the same viewpoint now. It is indeed time to rise the question if there are still any real differences between dependency and constituency:

2.4 Equivalence of Dependency and Constituency

Sleator & Temperley [1993] state that "[i]n most sentences parsed with our dictionaries, constituents can be seen to emerge as contiguous[8] connected collections of words attached to the rest of the sentence by a particular type of link" (Sleator & Temperley 1993: 3). Although "this is not the way we think about Link Grammars, and we see no advantage in taking that perspective" (ibid.), the question of equivalence and translatableness is certainly relevant to assessing the theoretical linguistic performance of dependency and ultimately Link Grammars.

Covington (1994: 2) states that "[d]ependency grammar still recognizes constituents, but they are a defined rather than a basic concept. The usual definition is that a constituent consists of any word plus all its dependents, their dependents, and so on recursively. (Tesnière calls such a constituent a NŒUD.)." In e.g. GB,

[8] The unobtrusive word *contiguous* anticipates a major link grammar problem: All moved (dislocated, raised, non-projective) elements cannot belong to the constituent they should and have to be linked by auxiliary links, which are often linguistically inappropriate.

constituents are basic, and government is derived, but cannot be derived unambiguously (see 2.3.3); in dependency, government is basic, and constituents are derived. Can they be unambiguously derived from constituency? Let us clarify if constituency and dependency can be mapped onto each other, or if they are radically different.

2.4.1 Mutually Excluding Alternatives

> Die Alternative Dependenz-Konstituenz ist von grundlegender Natur, und ein schlüssiges grammatisches System lässt sich erst dann aufstellen, wenn das eine oder das andere Prinzip als Ausgangspunkt gewählt wird, oder möglicherweise eine genau beschrieben Mischung aus beiden. (Schubert 1988: 56)

> Seitdem es eine Beschäftigung mit der Sprachstruktur gibt, konkurrieren in ihr zwei Erklärungsweisen, die sich heute – unter unserem methodologisch geschärften Blick – als die beiden grundlegenden Prinzipien der Grammatik herausstellen. (Baumgärtner 1970: 52)

For Jung [1995: 24-27], only dependency is able to express the syntactic 'inner' relations of a sentence, but only constituency is able to express the linear order, the 'outer' relations of a sentence:

> Mit Hilfe der Dependenzgrammatik kann zwar die innere Orginsation eines Satzes explizit dargestellt werden, aber eine adäquate Erklärung der linearen Abfolge ist nicht möglich. Die Phrasenstruktur ihrerseits liefert zwar die lineare Abfolge, kann aber die anderen syntaktischen Relationen nicht hinreichend explizieren.
>
> (Jung 1995: 26)

Jung therefore sees the two principles as complementary, following Baumgärtner [1970]. They suggest that "das Konstituenzprinzip syntaktische Positionen bestimmt, und das Dependenzprinzip für die Satzbedeutung grundlegend ist." (Baumgärtner 1970: 74). While this is intuitively convincing it is all the more disappointing that all semantic formal theories in the Montague style, except for perhaps the Prague School are based on constituency (cf. chapter 6.2).

2.4.2 Weak Equivalence

Gaifman [1965] and Hays [1964] gave dependency a rule formalism and proved that this rule formalism is context-free and weakly equivalent to constituency. Weakly equivalent means that they are able to describe the same set of sentences, but by means of different descriptions. If they were strongly equivalent, a dependency structure could be derived from constituency, and vice versa. In what ways are they different ? Baumgärtner [1970] states:

> Erstens: Zwischen Phrasenstruktur- und Dependenzgrammatik herrscht die Relation der sogenannten schwachen *f*quivalenz. Das bedeutet, dass beide Grammatiken die gleichen Mengen von Sätzen erzeugen. [...] Zweitens: Die beiden Grammatiken sind – in gewisser Weise – auch stark äquivalent, erzeugen also die gleichen Sätze mit den gleichen Kategorialstrukturen. Das gilt jedoch nicht voll wechselseitig. Nur zu jeder beliebigen Dependenzgrammatik lässt sich eine kategorial wirklich äquivalente Phrasenstrukturgrammatik effektiv

> konstruieren. Die umgekehrte Konstruktion setzt eine ausserordentlich beschränkte Phrasenstruktur–grammatik voraus. (Baumgärtner 1970: 59)

If this holds true, constituency is more powerful. As we will see in 2.4.5, the "ausserordentlich beschränkte Phrasenstrukturgrammatik" (= extremely reduced PS grammar) Baumgärtner mentions in the above quotation turns out to be a an X-bar grammar without intermediate levels, i.e. with only maximal projections (X") and lexical items (X).

Let us consider closely how far conversions are possible, i.e. how far they are strongly equivalent, and if Baumgärtner's quotation holds true:

2.4.3 Derivation of Constituency from Dependency

As for the derivation of constituency from dependency, Jung [1995: 25] shows that for many dependency structures, the translation is unambiguous. His example structure of

(64) wohnt in einem Studentenwohnheim

has only the constituent structure which is represented by the labeled bracket [wohnt [in [einem Studentenwohnheim]]] as a counterpart. But when a head has two or more dependents, the translation becomes ambiguous. His example structure of

(65) das Brot in der Küche

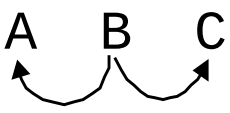
has the two possible constituent translations

(65a) [[das Brot] [in [der Küche]]] and

(65b) [das [Brot [in [der Küche]]]]

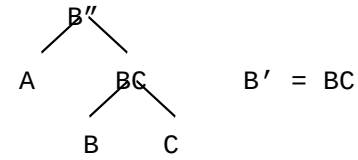
The question is if the article *das* or the PP *in der Küche* should be attached more closely to the head. In other words, for a dependency

(66)



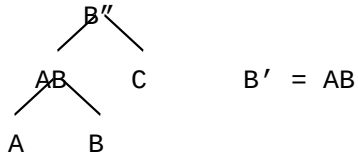
should the corresponding constituent structure be

(66 a)



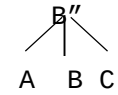
or

(66 b)



If we use a structure with three instead of two branches, we can unambiguously convert to

(66 c)



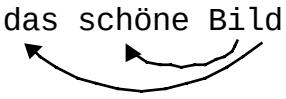
As we shall see in 2.4.5, however, this may blur important distinctions which one can make in X-bar theory; namely if [BC] as in (66 a) should be B', or if rather [AB] should be B'.

While (65a.) is the usual labeled bracketing, (65b.) is not what we wanted to express. If a head has two or more dependents, then either all of them point into the same direction, or they do not:

2.4.3.1 Several Dependents into Same Direction

If they do point into the same direction, as in the example of

(67)



we may unambiguously translate to the appropriate constituency labeled bracketing [das [schöne Bild]] by postulating a rule of proximity, which is, loosely

`Rule: Take closer dependents first.`

This idea was formulated by Hudson [1990: 149-50], it builds on the assumption that semantic proximity can be derived from syntactic proximity:

> If ... both adjuncts precede the head, as in *typical French houses*, then neither of them can be combined with the head until this is processed, but if we assume a push-down stack for holding unattached words then they will be attached in reverse order, again producing a subsense for the nearer adjunct first. This has the advantage of producing an entity, 'French house', to which the meaning of

> *typical* can be applied, thus solving a problem for dependency theory which was raised by Dahl (1980) (Hudson 1990: 150)

Covington [1992, 1994a] takes up this suggestion in order to build up intermediate (X') nodes correctly when converting from dependency to X-bar. cf. 2.4.5.

2.4.3.2 Several Dependents into Different Directions

If the dependents point into different directions, the problem persists, as shown in the above example of *Das Brot in der Küche*. We either have to reject Baumgärtner's statement that "zu jeder beliebigen Dependenzgrammatik lässt sich eine kategorial wirklich äquivalente Phrasenstrukturgrammatik effektiv konstruieren" (Baumgärtner 1970: 59), as Jung [1995:26] does; or we have to find some principled system which would allow us to predict which of the possible constituent structures is the appropriate one.

If we bear in mind that in the Tesnière framework ordering plays no role, the problems listed here also appear if both dependents are in the same direction.

I will present four suggestions here, which may be helpful, but maybe they cannot solve the problem satisfactorily. They all start with the same question: If we have two dependents in opposite directions, how can we decide which one we should attach first? For example in (65), if we attach the article on the left first and then the PP on the right, we get

```
( 65a) [[das Brot] [in [der Küche]]]
```

but if we attach the PP right first and then the article left, we get

```
( 65b) [das [Brot [in [der Küche]]]]
```

2.4.3.2.1 Priority Based on Word Class or Link Type

We may suggest that some word classes or link types (which requires us to use labeled dependencies, of course) get priority over others as to when they are attached. Specifiers like articles will indeed attach first, leading to the correct structure (65a). A more elaborate version of this idea can be found in Covington [1994a], cf. 2.4.5.

2.4.3.2.2 Head-Initial and Head-Final Languages

Head-initial languages like English usually have heads to the left and complements to the right. Therefore, elements on the left are attached first. Note that Chomsky [1995: 413] uses this argument as a key factor for the process of ordering in the minimalist programme, because order is no longer a syntactic, but a phonological component: in another crucial way, Chomsky adopts dependency tradition, although without explicit reference to the sources of his ideas.

2.4.3.2.3 Minimalist Approach

Since Bare Phrase Structure is dependency-inspired, as we have seen in 2.3.12, we may use its tools for finding an answer. As we have seen in 2.3.12.3, X' and indeed all bar levels are derived in Bare Phrase Structure. In such a view, all X-bar levels are configurationally derived. This means that the Merge operation faces exactly the same questions as a conversion from dependency to constituency. Merge, exactly like dependency, links two structures together and finds out which one is the head, i.e. in which direction a dependency goes. Chomsky [1995: 397] e.g. states that "[t]he head-complement relation is the "most local" relation of an XP to a terminal head Y", but this information is not available when parsing a sentence, i.e. calculating LF from the numeration, because e.g. in the numeration in (65) *in* appears just as local as *das*. I am not sure how the minimal programme solves this question, but it will probably be by means of labeled dependencies.

2.4.3.2.4 Underspecifying Scope-Ambiguities

If a problem persists, declare the bug as a feature! While (65a.) looks more usual, we cannot rule out that a specifier modifies the more complex N', as in (65b.).

(68) Would you like this [bread with cheese] here or that [roll with cheese] over there ?

When e.g. an noun is modified by several adjectives the scope of them is often very unclear, it is indeed an open question if syntax should deal with them.

(69) A big red interesting book

Should this be [big[red[interesting book]]] or rather [big, red,interesting [book]] ? Is *a big interesting red book* different from *an interesting big red book* ? Or, to take an example with modifiers in both directions

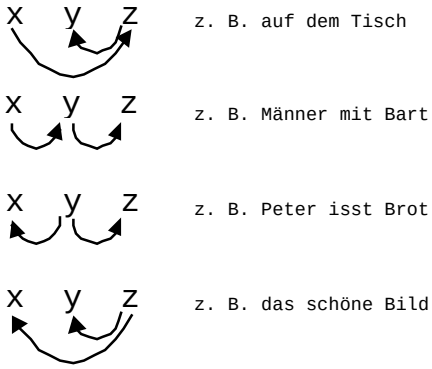
(70) An interesting proposal suggested by the chairman

Is this rather *an [[interesting proposal] suggested by the chairman] or an {interesting (proposal} suggested by the chairman)* ? Unfortunately, due to the crossing constituents, constituency cannot even express the second alternative! It seems that the serious flaw is not that dependency cannot always be converted to constituency, the serious flaw is that constituency forces one to make distinctions between ambiguities which are usually unnoticed and irrelevant, but – even worse – constituency is unable to express all of them. These mostly useless distinctions we are forced to make are irrelevant at the syntactic level anyway, they are – if at all – a semantic question. On a syntactic level, the unambiguous conversion to (66 c) seems to be sufficient.

This does not mean that for other phenomena intermediate nodes would not be helpful. There the dependency restrictions are of course a drawback.

2.4.4 Derivation of Dependency from Constituency

For the translation from constituency to dependency, Jung [1995: 25] gives the example of a constituent structure represented by the labeled bracketing [x [y z]] and shows that in natural language, at least the four following dependency structures can be counterparts:



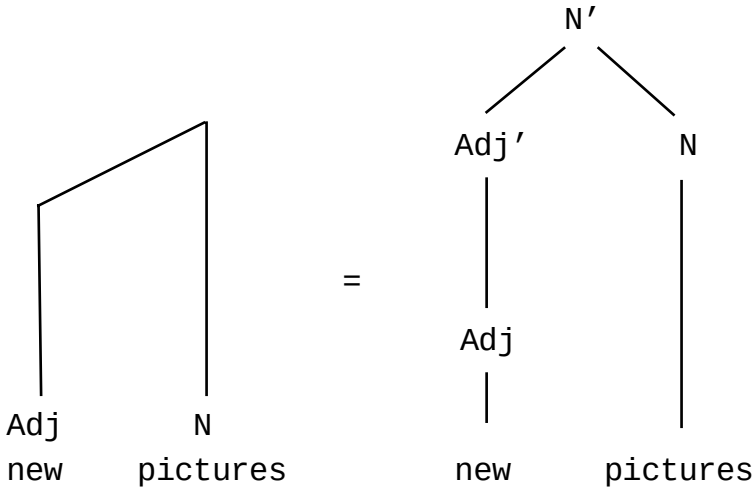
Jung concludes: “Hier zeigt sich offensichtlich, dass das Konstituenzprinzip die innere Organisation des Satzes nicht hinreichend erklären kann.”(ibid.) But as soon as the head information is available, as in all modern PSG grammars, the translation from constituency to dependency becomes unambiguous in all cases. Please refer to chapter 2.3.2 for a detailed discussion of heads and possible differences of its definition between constituency and dependency.

But as Baumgärtner [1970] stresses, this conversion only works for a restricted constituent PS grammar. The "ausserordentlich beschränkte Phrasenstruktur–grammatik" [Baumgärtner 1970: 59] mentioned turns out to be a restricted version of X-bar theory, as we shall see in the following subchapter:

2.4.5 X-bar Theory and Dependency Grammar

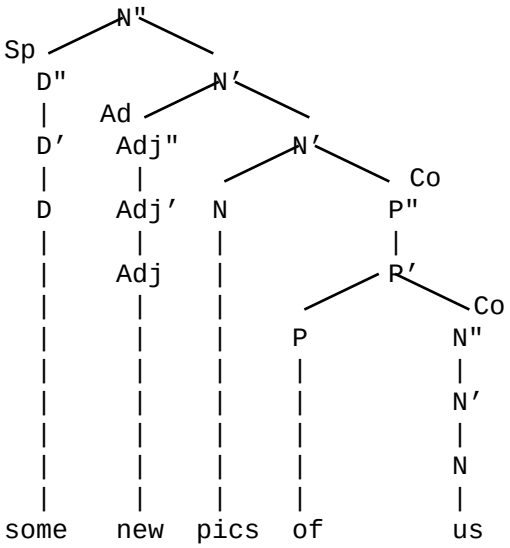
It is surprising to see Dependency grammar and (a version of) X-bar syntax, a very conservative stronghold of PSG, emerge as equivalent. According to Covington [1992, 1994a], dependency grammar (DG) is equivalent to X-bar theory with only one non-terminal bar-level (e.g. X-bar theory with X’’ and X, but no X’, or with X’, but no X”).

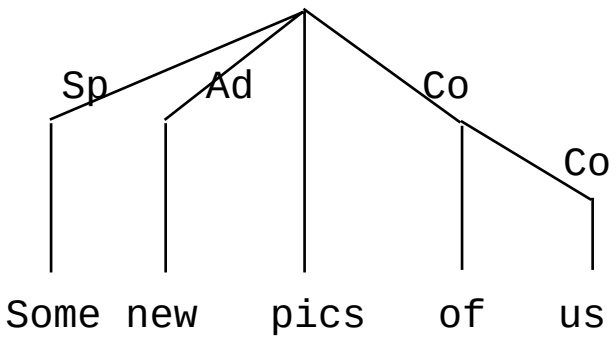
> GB theory shares with DG the crucial claim that every phrase has a head. Dependency trees are equivalent to X-bar phrase-structure trees with only one non-terminal bar level, like this:



> Since the D-tree contains no non-branching nodes, the Adj' node here has no direct counterpart in it; rather, Adj' is supplied implicitly by X-bar theory, which stipulates that all sisters of the head are maximal projections. ... One non-terminal bar level is not enough for current GB theory. (Covington 1992: 2)

The three kind of dependencies X-bar theory knows, i.e. *Specifier* (Sp), *Adjunct* (Ad), and *Complement* (Co) can be emulated by explicitly labeled dependencies.





Covington [1994a] states that the real problem is not that there is only one non-terminal bar-level, but that no stacking of X' nodes are allowed.

> DG formalism is equivalent to a particular strict form of X-bar theory in which:
>
> - There is only one non-terminal bar level (...);
> - Apart from bar level, X and the X' immediately dominating it cannot differ in any way, because they are "really" the same node;
> - There is no "stacking" of X' nodes (an X' node cannot dominate another X' with the same head).
>
> The third of these observations is the critical one.
>
> (Covington 1994a: 3)

Covington uses the standard examples from Radford [1988: 179-195], i.e. *one* pronominalization and Adjunct vs. Complement N' constituents (*[N"[N'[N'[N[students]PP[of physics]]][PP[with long hair]]]]*) to give evidence for the necessity of N' recursion. He also mentions Dahl's example *([typical [French houses]]*) which we have discussed in chapters 2.2.6.1.1 and 2.3.2.4.2.

Covington's solution is simply to use consistently labeled dependency types, as lined out in 2.4.3.2.1. Because it would be difficult to establish an explicit hierarchy of link types, he consequently suggests to label the links as *specifier*, *adjunct* and *complement*. Covington urges us to "[r]ecall that in a dependency grammar, constituency is a defined concept. The solution is therefore to change the definition." Covington suggests the following principled rules for mapping the dependency structures onto X-bar structures:

> ...instead of being considered equivalent to flat X-bar trees, dependency structures can be mapped onto X-bar trees that introduce stacking in a principled way. Here is a sketch of such a reinterpretation, consistent with current X-bar theory. Given a head (X) and its dependents, attach the dependents to the head by forming stacked X' nodes as follows:
>
> 1. Attach subcategorized complements first, all under the same X' node. If there are none, create the X' node anyway.
> 2. Then attach modifiers, one at a time, by working outward from the one nearest to the head noun, and adding a stacked X' node for each.
> 3. Finally, create an X" node at the top of the stack, and attach the specifier (determiner), if any.
>
> (Covington 1994: 7)

Covington stresses that "DG always WAS a notational variant of X-bar theory" (Covington 1994: 8) and that "DG still imposes stricter requirements than transformational grammar, because in DG, violations of X-bar theory are flatly impossible, not just undesirable" (Covington 1994: 9). Baumgärtner [1970] mentions that already Gaifman [1965: 325-334] has proved this: "Gaifman beweist die wechselseitige "Korrespondenz" von Dependenz- und Phrasenstrukturgrammatik über die sog. Unique Phrase-Structure Grammar, in der nur grammatische Klassen

(Kategorien) erster Stufe zugelassen sind, mit Ausnahme der Selbstdominanz von Kategorien" (Baumgärtner 1970: 59, Footnote 12).

Hudson [1990] has anticipated and laid the ground for Covington's suggestion. He is also acutely aware of the restrictions X-bar theory imposes:

> One of the reasons for the relative neglect of dependency theory by modern linguists is surely the fact that dependency grammarians have tended to favour a very conservative version of dependency theory which makes it equivalent to X-bar theory (except for the explicit recognition given, in the structures generated, to the head). We all know that natural languages contain phenomena that cannot be explained by means of an ordinary context-free phrase-structure grammar, of which X-bar theory is an example (...), so this simple kind of dependency grammar adds little to our understanding of syntax and has little appeal to linguists who are accustomed to more powerful systems like GB, LFG, GPSG and CG. (Hudson 1990: 113-4)

Hudson therefore argues for a more powerful version of dependency grammar (namely his Word Grammar, [Hudson 1990: 113-122]), in which exceptions to the principle that every word has exactly one head are introduced, especially for raising structures. He also claims to thereby solve a general problem in functional control [Hudson 1997].

2.4.6 What Tesnière said

It is time to raise some doubts. If Dependency is really a variant of X-bar theory, then, because Lexical-Functional Grammar (LFG) (cf. [Dalrymple et al.: 1995]) usually adheres to X-bar theory, so does LFG c-structure. At least Tesnière's dependency is based on grammatical functional relations, like LFG f-structure, and because they have similar aims, it should ideally be possible to map them onto each other along principled rules. But then LFG c- and f-structures should also be very similar and perhaps mappable onto each other? Why would they be distinguished at all ?

I had unfortunately no time to follow this line of thought, but doubts about the correctness of the above subchapter come from other sources. Baumgärtner [1970: 60] and Järvinen & Tapanainen [1997: 3] point out that there has to be an error in the proof of context-freeness by Gaifman [1965] and Hays [1964]. The formalization which they suggested for dependency is indeed context-free, but the original framework by Tesnière [1959] is context-sensitive.

Gaifman [1965] and Hays [1964] start from the conception of dependency as an existence relation (cf. 2.3.2.4.3). If we can conclude from the existence of a head χ to a dependent φ then

(71) $(\varphi)\chi$

Because there can be several dependents, because they can be situated on both sides of the head, they use the general form:

(72) $(\varphi_m) \ldots (\varphi_1)\ \chi\ (\kappa_1)\ (\kappa_n)$, where 0<m & 0<=n or
0<=m & =<n respectively

Baumgärtner [1970: 57] explains that this formula allows easy comparison between constituency and dependency, because it can be transformed into the following rewrite rule (where [#] is the position of *i+1* of the governing symbol χ:

(73) $\chi \rightarrow \varphi_1 \ldots \varphi_i$ [#] $\varphi_{i+2} \ldots \varphi_n$ where 0 < I <= n.

"Eine Dependenzgrammatik besteht dann, formal gesichert, aus einer endlichen Menge von Regeln der Form (73). Sie ist dann aber zum Verwechseln ähnlich der Phrasenstrukturgrammatik" (ibid.: 58) and it is easy to make Gaifman's [1965] and Hays' [1964] proof. But (72) and (73) fail to express a dependency assumption which was central for Tesnière:

> Nach Tesnière besteht der grammatische Satz aus einem Ensemble von strukturellen Konnexionen, die zwischen seinen einzelnen Einheiten in bestimmter Weise Abhängigkeit etablieren, so dass eine Konnexion $\chi\varphi$ bedeutet, dass χ regierendes und φ regiertes oder untergeordnetes Symbol ist. Diese Abhängigkeits-Konnexion ist – wie die Relation der Konstituenz – nicht kommutativ. Damit wird aber – im Gegensatz zur Konstituenz – nicht eine Eigenschaft der linearen Ordnung angegeben. Die Position der Einheiten spielt höchstens in zweiter Linie eine Rolle. Tesnière geht es ausschlieesslich um die inneren Beziehungen der Einheiten, die dem linearen Satz zugrunde liegen ... In dieser Auffassung müssen die oben eingeführten Dependenz-Ausdrücke [(72) and (73)] als überkonstruiert erscheinen. Sie stellen sich als zweckmässige Anpassung an den mathematischen Vergleich von Konstituenz und Dependenz heraus. Sie gewähren diesen Vergleich überhaupt nur durch ihre linerare Ordnung. Nach Tesnières ursprünglicher Theorie sind sie jedoch von jeder Kennzeichnung der Position zu entlasten.
>
> (Baumgärtner 1970: 60-1)

His newly introduced dependency formulation [ibid.] is context-sensitive and does not take word-order into consideration.

(74) $\chi(\varphi_1 \ldots \varphi_n)$ where 0 < n.

Note that dependency is now also functional in the mathematical sense. Because (74) cannot be translated to a PSG rule Baumgärtner concludes that constituency and dependency are complementary.

> Nach all dem ist das Prinzip der Dependenz unentbehrlich für die Erklärung der inneren Organisation des Satzes. Damit erhält die Phrase von der inneren Organisation des Sates eine tiefere Bedeutung: Sie bezieht sich nicht mehr direkt auf seine jeweilige Konstituentenstruktur. Das Prinzip der Dependenz liegt vielmehr der Theorie der '*Funktionalität*', in älterer Redeweise: der '*Beziehungsbedeutungen*' des Satzes zugrunde, womit nun klar wird, dass dies Prinzip nicht bloss von Position und Morphologie absehen kann, sondern nicht einmal die lexematischen Verhältnisse des Satzes im ganzen zu umfassen braucht. Eine Dependenzgrammatik kann diese verschiedenen grammatischen Eigenschaften überhaupt nicht adäquat erklären.
>
> Umgekehrt vermag das Prinzip der Konstituenz solche funktionalen Beziehungen nicht zu erklären, weder in einzelnen Zügen noch gar in ihrer Gesamtheit. Dies ist auch unabhängig davon, wie abstrakt eine

> Konstituentenstruktur angelegt ist. Damit lässt sich ... deutlich machen, dass sich die beiden Prinzipien komplementär verhalten.
>
> (Baumgärtner 1970: 66)

Although Baumgärtner's discovery is fundamental and central, many linguists have failed to recognize it.

> *Die Dependenzgrammatik (DG) verhält sich komplementär zur Konstituenzstrukturgrammatik (KSG), beide Theorien ergänzen sich also.*
>
> Diese von Baumgärtner [1970] und anderen vertretene These wurde bemerkenswerterweise von den meisten Dependenzgrammatikern nie akzeptiert. Diese fassten DG und KSG als alternative, aber weitgehend äquivalente Theorien auf und fanden sich damit in Übereinstimmung mit Gaifman und Hays, die prinzipielle Vergleiche angestellt hatten.
>
> (Engel 1996: 54)

Engel [ibid.] also states that Tesnière himself has never addressed the first precursors of transformational grammar of his time.

It has to be added that Baumgärtner [1970] had to base his judgement on the PS theories of his time. He could not know about the advent of lexical-functional grammar (LFG, cf. ([Bresnan 1982], [Dalrymple et al.: 1995] and 2.3.8), which can be seen as a direct answer to his statements by its use of a distinct constituent (c-) structure and functional (f-) structure. What he could even know less is that one of the godfathers of constituency, Noam Chomsky would write anno 1995 about his minimalist programme:

> Nothing has yet been said about ordering of elements. There is no clear evidence that order plays a role at LF [logical function ≈ semantic component] or the computation from N [numeration = chaîne parlée] to LF. Let us assume not. It must be, then, that ordering is part of the phonological component, a proposal that has been put forth over the years in various forms. It seemed natural to suppose that ordering applies to the output of Morphology, assigning a linear (temporal, left-to-right) order to the elements it forms, all of them X^0s though not necessarily lexical elements.
>
> (Chomsky 1995: 413)

It is difficult to understand, however, why Chomsky does not pay appropriate reference to the people who have defended these ideas in times when Chomskyan theories looked so different – i.e. Tesnière or Baumgärtner or other dependency linguists.

2.4.7 Non-Projectivity

Covington [1990] describes an experimental non-projective, i.e. context-sensitive parser. The Finnish *Dependency Parser for English* [Järvinen & Tapanainen 1997, 1998] is also context-sensitive.

The problem is that in most cases unrestricted context-sensitivity cannot be allowed, otherwise the parser will present massively ambiguous structures, most of them completely absurd. Every article will e.g. try to modify each noun anywhere in a

given sentence, no matter how far away, as we will see in 3.3. If we really need non-projectivity, then the real challenge for all non-projective parsing is to write correct linearisation rules and to determine where context-sensitivity should be allowed, while it should generally be forbidden. As a first approximation one can follow Tesnière [1959], who suggest that transgressions of projectivity are usually marked, e.g. by morphological agreement.

> Usually dependents either immediately follow or precede their heads (projectivity) and when they do not, (2) there can be additional devices such as morphological agreement indicating the connection.
>
> (Järvinen & Tapanainen 1998: 4)

The discussion of stranded quantifiers in 2.3.10 has shown that monostratal theories have to employ non-projective links if they want to treat quantifiers as part of the DP. Indeed, context-free systems like Link Grammar have to use a set of ad hoc links which are linguistically hard to account for. Link Grammar takes an extreme position here. Any Link Grammar grammar has to be fully context-free. In addition every link allowed or required by the lexical entry of a given word needs specification if the word to be linked to occurs before or after it. This extreme position means that Link Grammar is unable to express a big number of elegant generalizations, as chapter 5.1 discusses. E.g. verb-subject links in question or in passive sentences are of a different type than the corresponding assertive and active sentences, because they are in opposite directions.

2.5 Parsing Efficiency

Parsing efficiency of non-projective dependency-based parsers is still a debated topic. On the one hand, due to the smaller number of nodes we expect to be able to parse faster:

> An advantage of dependency trees consists in the fact that the number of their nodes can be kept relatively low: function words can be differentiated from 'autosemantic' lexical units and labeled as parts of complex nodes ... Also, a dependency tree contains no non-terminal nodes.
>
> (Sgall, Hajičová, Panevová 1986:5)

On the other hand, no well-behaved dependency parsing algorithms could be found yet:

> ... we know that we can parse a string in a CFG [context-free grammar] in at most $0(n^3)$ time, i.e. in an amount of time proportional to the cube of the length of the input string. Though the parsing of non-projective DGs [dependency grammars] has been discussed (see (Covington, 1990) and the references therein), to our knowledge no formal result has been published. There is reason to believe that in the worst case they can be parsed in a time proportional to an exponential function of the length of the input string ($0(2^n)$). If this worst case actually occurred in natural language parsing, than a DG would not be a very appealing candidate for a model of human language processing.

(Rambow & Joshi 1994: 9)

The question is how often this worst case scenario occurs. According to the authors of the Finnish Functional Dependency Grammar (cf. 4.2.4) [personal communication] it very rarely occurs; their parser is extremely fast: "it runs using a Pentium 166 MHz machine at the speed of 350 words per second" (Tapanainen & Järvinen 1997: 1).

Because the efficiency question is not sufficiently answered yet, Michael Covington has plans to investigate it: "What I want to do right now is make a general study of dependency parsing algorithms" [personal email message, May 19, 1998].

2.6 Conclusions: Remaining Differences

This is the longest chapter of this paper. It is not easy to draw simple conclusions. Yet I hope to have shown that only seven differences between dependency and constituency remain. Because both approaches show a certain variety of schools, not all of these differences apply to all schools, however. E.g. *(I.) No intermediate categories in dependency* does not apply if we use Tesnière's transformations, *(III.) Fixed Word-Order* does not apply to *Immediate Dominance* Rules of ID/LP grammars or Chomsky's minimalist programme, *(IV.) functionalism* can be partly simulated by a *theta theory*, or for *(VI)* the Prague School FGD (cf. 2.2.5) aim for a generative version of dependency.

(I.) **No intermediate categories in dependency**: As we have seen in 2.2.6.1.1, dependency is indeed unable to express intermediate categories (X'-level). It only recognizes X" and X (cf. 2.4.5). Note however that translation rules, as Tesnière suggested them, may deliver the required "re-entrancy" at X'-level. In many cases, lexical rules as used in LFG or HPSG can also repair this deficiency of dependency.

(II.) **No context-sensitivity in constituency**: For constituency, non-projective structures cause principal problems. For dependency, projectivity is just a parameter that can be set or relaxed as appropriate, as seen in 2.4.6.

(III.) **Free word order in dependency**: In many languages, configurationally fixed word-order misses the possibility for many generalizations or requires an artificially complex transformational system (cf. 2.3.8.3)

(IV.) **Serious problems for functionalism in constituency**: as described in 2.3.8.3, because of (II.) and (III.) a truly functional approach to syntax cannot be based on constituency only. We have to enrich constituency with a functional (or theta) layer and map structures in a principled way, as e.g. LFG does.

Using only one layer, a functional one in the case of dependency, will speed up parsing.[9]

(V.)**Parsing efficiency: Fewer nodes in dependency**: It is debatable, however, if dependency parsers are faster. Most dependency parsers are NP-complete in the worst case, but in practice dependency parsers can be extremely fast (cf. 2.5).

(VI.)**No generation in dependency**: Dependency structures are not suitable for language generation, because they totally underspecify linearisation (some formalisms also do not represent function words in the structure, instead the functional relation is reported). This allows for economical and functional parsing, but a parsed sentence cannot be reconstructed using the same method.

(VII.)**No coordination in Dependency:** in pure dependency, coordination cannot be expressed. A dependency system will have to employ a constituency element like Tesnière's *jonctions*.

[9] It is possible, however, to pre-compile a two-level grammar down to one level.

3. Toy Dependency Parsers

Unfortunately I cannot prove (or disprove) point (V.) above. I have developed some small dependency parsers in order to show that dependency parsing is not more complicated or inefficient than constituency parsing.

3.1 A Toy Parser for German in Perl

If we are interested in efficient fast parses, it may also be worth considering to implement a system in the faster imperative languages instead of a descriptive language like Prolog.

I have therefore started to write a toy parser in Perl. It bases its parses on the output of the morphological analyzer *Gertwol* [Lingsoft 1994]. The parser is very incomplete, it does not even allow ambiguity yet, but it is equally short and simple. It only took a day to write it, and I hope to show that dependency grammars can be easily implemented in imperative languages. Explanatory comments follow after the listing:

(75) The present incomplete simple Perl Toy Parser for German:

```
#!/opt/gnu/bin/perl
# experimental dependency parser for German
# (C) 1998 Gerold Schneider, University of Zurich
# Version 0.3 : THE BASICS
# no ambiguity
# no word-order
# fully context-sensitive
# V 0.31: markread
##############################################################################

# preliminaries

$in=@ARGV[0];
$gwpath='/opt/src/lang/Gertwol/ger-980126/bin';
$path=`pwd`; chop $path;
$source="$path/tmp";
$gwout="$source.gwout";
print "$source\n";

open (TMP,">$source");                          ## write sent to tmp file
print TMP "@ARGV[0]\n";
print "@ARGV[0]\n";
close (TMP);

# This is the call for the Gertwol morphological analyser
 system("rsh claude $gwpath/tw-ger-pp $source \\| $gwpath/tw-ger -u -
     \\\\#\\\\# > $gwout");

# Load the sentence into an array, word by word, each with
# Gertwol morphological info
$/="\n\n";
open (MORPH,"<$gwout");
$n=0;
while (<MORPH>) {
        tr/~|#//d;
```

```
      print "##$n $_\n";
      $sent[$n]="$n\n$_";                    ## load sent to array
      $n++;

}
close(MORPH);

print
"#########################################################################\n";

# simplest root dependency: take indicative verb
@list=&take(V,IND);  # 'take' an indic. verb from sentence, return
                     # dependency structure to @list
print "\n====> STRUCTURE: $list[0]  - ";
foreach $word(@sent) {
  ($wordpos,$token,@rest) = split("\n",$word);
  unless ($markread[$wordpos]) { print "$token " }
}
print "\n\n";

#########################################################################

sub take {

     local($no,$cat,@cats,$feats,$feat,@featlist,$cand,$pos,$word,$lemma,
     $class,@ambi,@catdeps,$line,@linefeat,$ffl);
  $cat=$_[0];  # first argument          ## sought category
  @cats = grep(/ $cat /, @sent);         ## find cat.s anywhere in sent
                                         ## without word-order!
  $feats=$_[1]; # second argument        ## sought features
  @featlist = split(/ /,$feats);         ## make array
  print "\ntake / $cat / $feats\n";
  ### WANT TO PRINT OUT CANDIDATES ?: UNCOMMENT BELOW:
  #&printarray(@cats);
  foreach $cand (@cats) {                ## candidates from @cat
    ($pos,$word,@ambi) = split(/\n/,$cand); ## @ambi=all Gertwol lines
    print "candidate $word :\n";
    foreach $line(@ambi) {               ## check each gertwol-line
      ($lemma,$class,@linefeat) = split(/ +/,$line); ## features in array
      if ($cat eq $class) {              ## if line has correct category
        $fflag="TRUE";
        foreach $feat(@featlist) {       ## every feature has to fit
          unless("$line " =~ / $feat /) { $fflag="FALSE" }
        }
        if ($fflag eq "TRUE") {
          ## if all features fit find dependents
          ## 'dependents are new local heads' -> recursion
          @catdeps=&fetchdeps($lemma,$cat,$line,$pos,@markread);
                     ## $line=HPSG HFP
          # print "STRUC: $catdeps[0]\n";
        }
      }
    }
  }return $catdeps[0];
}

#########################################################################

sub fetchdeps {
  local(@depcat,@deps,$dep,@depfeatlist,$valcount,$arglist);

####### DEPENDENCY RULES = Grammar

## VERB
  if ($cat eq "V") {
    if ($line =~ / (IND |KONJ )/) {
      $depcat[0].="S";             # nomen
      $depfeatlist[0].="NOM";      # subj
    }

    if ($lemma =~ /(\"haben\"|\"sein\")/) {
      $depcat[1].="V";                   # past participle
```

```
      $depfeatlist[1].="PART PERF";
    }

    if ($lemma =~ /(\"werden\")/) {
      $depcat[1].="V";                    # future
      $depfeatlist[1].="INF";
    }

    if ($lemma =~ /(\"bleiben\")/) {
      $depcat[1].="A";                    # copula
      $depfeatlist[1].="UNDEKL";
    }


# verb valencies HERE: extend later [2]
  }

## NOUN
  if ($cat eq "S") {
    $depcat[0].="ART";             # det
    if ("$line " =~ / (MASK |FEM |NEUTR )/) {;
      $depfeatlist[0].=$1 }
    if ("$line " =~ / (NOM |GEN |DAT |AKK )/) {;
      $depfeatlist[0].=$1 }
    if ("$line " =~ / (SG |PL )/) {;
      $depfeatlist[0].=$1 }

  }

####### END OF DEPENDENCY RULES

  $valcount=0;
  ## preview of valencies
  if (scalar(@depcat)) {     ## if array exists
    print " is head of\n";
    &printarray(@depcat);
  }
  else {
    print "## $line +> END\n"; ## no array -> no dependencies found
    $lemma =~ s/[ \t"]//g;
    $markread[$pos]++;   ## mark position read for linkage completeness
    return "$lemma:$cat"; ## return local dependency structure
  }
  foreach $dep(@depcat) {
  ## 'take' new dependencies for each dependent
  ## 'local dependents are new subheads'
    print "## $line [$valcount] ==> $dep:$depfeatlist[$valcount]\n";
    @list=&take($dep,$depfeatlist[$valcount]);
    $lemma =~ s/[ \t"]//g;
    # print "LIST: $lemma($list[0])\n";
    $arglist .=" $list[0]";
    $valcount++;
  }
  $arglist =~ s/^ //;
  $arglist =~ s/ /,/g;
  $markread[$pos]++;
  return "$lemma($arglist):$cat";
}

################################################################################

sub printarray {
  local($ele,$eleno);
  print"v====================\n";
  $eleno=0;
  foreach $ele(@_) {
    print "\@\@ $eleno: $ele\n";
    $eleno++;
  }
  print"^====================\n";
}
```

The sentence to be parsed is first morphologically analyzed by Gertwol. Then the real parser starts. The core of the program consists of **two subprograms** which recursively call each other:

The **first subprogram**, `&take`, finds the dependents required by category and features in the Gertwol analysis of the sentence. The Perl function `grep` works similar to the UNIX `grep` command, but it searches an array (here the array `@sent`, which contains the Gertwol output for the sentence) instead of a file. It will find the candidate words of the required category (non-projectively across the whole sentence) and check which candidate(s) have a morphological analysis that satisfies all the features. At the beginning of the parsing, `&take` is simply called for any indicative verb – a coarse approximation to finding the head of the sentence: `@list=&take(V,IND);`

At the end of the first subprogram, before a result is reported to the variable `@list`, the **second subprogram** is called, `&fetchdeps`. This subprogram contains the grammar, which specifies the categories and features of dependents for the head found by the `&take` from which it was called. When these categories and features are established, each of them has to find the dependents in the Gertwol analysis of the sentence – the task for which `&take` was written. Therefore, this first subprogram is called again.

Whenever the end of a dependency tree is reached, i.e. no dependencies can be found, the subprogram returns the current lemma, i.e. the end of the list and stops recursing (perl command `return`). Resolving the recursive chain of subprograms returns the growing dependency list (much like in Prolog), until the next unsaturated valency is found (`foreach $dep(@depcat)`, counted by `$valcount`), eventually up to the first call, where it is displayed.

A few sample sentences follow below. The unlinked words are displayed after the dependency structure.

```
( 76) Sample Sessions:
<310 kasos /gschneid/dependenz> dparse-v0.31.perl "eine Frau hat hier
     gearbeitet"
/home/ludwig7/gschneid/dependenz/tmp
eine Frau hat hier gearbeitet

GERTWOL 980126
Copyright (C) Lingsoft, Inc. 1994-1995

TWOL 1998/01/16
Copyright (C) K. Koskenniemi and Lingsoft, Inc. 1983-1998
Two-Level Compiler
Copyright (C) 1994, Xerox Corporation. All rights reserved.
.. save file loaded
##0 "<eine>"
        "ein"  ART INDEF SG NOM FEM
        "ein"  ART INDEF SG AKK FEM
        "einer"  PRON INDEF SG NOM FEM
        "einer"  PRON INDEF SG AKK FEM
```

```
        "einen"  V IND PRfS SG1
        "einen"  V KONJ PRfS SG1
        "einen"  V KONJ PRfS SG3
        "einen"  V IMP PRfS SG2


##1 "<*frau>"
        "*frau"  S FEM SG NOM
        "*frau"  S FEM SG AKK
        "*frau"  S FEM SG DAT
        "*frau"  S FEM SG GEN


##2 "<hat>"
        "haben"  V IND PRfS SG3


##3 "<hier>"
        "hier"  ADV


##4 "<gearbeitet>"
        "arbeiten"  V PART PERF
        "gearbeitet"  A(PART) POS UNDEKL


#####################################################################
     #

take / V / IND
candidate "<eine>" :
 is head of
v====================
@@ 0: S
^====================
##      "einen"  V IND PRfS SG1 [0] ==> S:NOM

take / S / NOM
candidate "<*frau>" :
 is head of
v====================
@@ 0: ART
^====================
##      "*frau"  S FEM SG NOM [0] ==> ART:FEM NOM SG

take / ART / FEM NOM SG
candidate "<eine>" :
##      "ein"  ART INDEF SG NOM FEM +> END
candidate "<hat>" :
 is head of
v====================
@@ 0: S
@@ 1: V
^====================
##      "haben"  V IND PRfS SG3 [0] ==> S:NOM

take / S / NOM
candidate "<*frau>" :
 is head of
v====================
@@ 0: ART
^====================
##      "*frau"  S FEM SG NOM [0] ==> ART:FEM NOM SG

take / ART / FEM NOM SG
candidate "<eine>" :
##      "ein"  ART INDEF SG NOM FEM +> END
##      "haben"  V IND PRfS SG3 [1] ==> V:PART PERF

take / V / PART PERF
candidate "<eine>" :
candidate "<hat>" :
```

```
candidate "<gearbeitet>" :
##      "arbeiten"  V PART PERF +> END
candidate "<gearbeitet>" :

====> STRUCTURE: haben(*frau(ein:ART):S,arbeiten:V):V  - "<hier>"

************************
<305 kasos /gschneid/dependenz> dparse-v0.31.perl "der Mann ist im Walde
     gegangen"
/home/ludwig7/gschneid/dependenz/tmp
der Mann ist im Walde gegangen

GERTWOL 980126
Copyright (C) Lingsoft, Inc. 1994-1995

TWOL 1998/01/16
Copyright (C) K. Koskenniemi and Lingsoft, Inc. 1983-1998
Two-Level Compiler
Copyright (C) 1994, Xerox Corporation. All rights reserved.
.. save file loaded
##0 "<der>"
        "der"  ART DEF SG NOM MASK
        "die"  ART DEF SG DAT FEM
        "die"  ART DEF SG GEN FEM
        "die"  ART DEF PL GEN
        "der"  PRON DEM SG NOM MASK
        "die"  PRON DEM SG DAT FEM
        "die"  PRON DEM VERALTET SG GEN FEM
        "die"  PRON DEM VERALTET PL GEN
        "der"  PRON RELAT SG NOM MASK
        "die"  PRON RELAT SG DAT FEM
        "die"  PRON RELAT GESPROCHEN PL GEN


##1 "<*mann>"
        "*mann"  S EIGEN Famname SG NOM
        "*mann"  S EIGEN Famname SG AKK
        "*mann"  S EIGEN Famname SG DAT
        "*mann"  S EIGEN Famname SG GEN
        "*mann"  S MASK SG NOM
        "*mann"  S MASK SG AKK
        "*mann"  S MASK SG DAT


##2 "<ist>"
        "sein"  V IND PRƒS SG3


##3 "<im>"
        "in-der"  PRƒP ART DEF SG DAT MASK
        "in-das"  PRƒP ART DEF SG DAT NEUTR


##4 "<*walde>"
        "*wald"  S MASK SELTEN SG DAT


##5 "<gegangen>"
        "gehen"  V PART PERF
        "gegangen"  A(PART) POS UNDEKL


#############################################################################
     #

take / V / IND
candidate "<ist>" :
 is head of
v====================
@@ 0: S
@@ 1: V
^====================
```

```
##      "sein"  V IND PRƒS SG3 [0] ==> S:NOM

take / S / NOM
candidate "<*mann>" :
 is head of
v====================
@@ 0: ART
^====================
##      "*mann"  S EIGEN Famname SG NOM [0] ==> ART:NOM SG

take / ART / NOM SG
candidate "<der>" :
##      "der"  ART DEF SG NOM MASK +> END
candidate "<im>" :
 is head of
v====================
@@ 0: ART
^====================
##      "*mann"  S MASK SG NOM [0] ==> ART:MASK NOM SG

take / ART / MASK NOM SG
candidate "<der>" :
##      "der"  ART DEF SG NOM MASK +> END
candidate "<im>" :
candidate "<*walde>" :
##      "sein"  V IND PRƒS SG3 [1] ==> V:PART PERF

take / V / PART PERF
candidate "<ist>" :
candidate "<gegangen>" :
##      "gehen"  V PART PERF +> END
candidate "<gegangen>" :

====> STRUCTURE: sein(*mann(der:ART):S,gehen:V):V  - "<im>" "<*walde>"
```

Both the grammar and the parser are very simple and need to be extended. But the basics of non-projective dependency parsing are easy to implement, also in a fast imperative language. Since the CL research community still prefers to use Prolog, I will devote the rest of the chapter to Prolog implementations.

3.2 Dependency Unification Grammar (DUG)

Steimann and Brzoska [1995] present their formalization of Hellwig's Dependency Unification Grammar (DUG) [Hellwig 1986]. But they base their implementation on the context-free dependency grammar definition [Steimann and Brzoska 1995:95-6] that we have seen in 2.4.6:

(77) $\chi \rightarrow \varphi_1 \ldots \varphi_i$ [#] $\varphi_{i+2} \ldots \varphi_n$ where 0 < I <= n.

In a Prolog DCG, e.g. for the verb *to give*, this is:

```
n(give, verb(N)) -->
  n(_, noun(N)),
  [n(give, verb(N))],
  n(_, noun(N)),
  n(_, noun(N)).
```

(Steimann and Brzoska 1995: 96)

For efficiency reasons, their implementation uses a formalism which is slightly different from DCG, but largely equivalent and also context-free. The translation of

their syntax to Horn clauses uses the same difference lists as the Horn clause translation of a DCG does [Steimann and Brzoska 1995:95-6].

Unlike DCG, their formalism does not express word order, which is therefore free. But I would like to go one step further and present a toy parser which features both free word order and is context-sensitive, i.e. which allows non-projectivity.

3.3 Dependency Existence Prolog Parser (DEPP)

3.3.1 The Basic Idea

If A is a head and B a dependent, how should we express this dependency in predicate logic and Prolog? It is tempting to suggest to write

(78) A → B

in analogy to the syntactic representation

(79) B A

.

In Horn Clause Logic (HCL), which is used by Prolog, the same relation is expressed by

```
( 80) B :- A.
```

While this looks intuitively convincing, two serious problems crop up:

- Heads often have several dependents, e.g.

  (81) A → B, C

  or in clause logic

  ```
  ( 82) B,C :- A.
  ```

  This cannot be expressed in HCL, however.

- A → B suggests that head A necessarily predicts the existence of dependent B. This is not true, however. It is a characteristic of the head that it can occur alone. Here we have a confusion between *necessary* and *sufficient* conditions. As we have seen in 2.3.2.4.3 and 2.3.2.4.4, rather the opposite holds: We can usually conclude from a dependent to the existence of the head.

The *existence* or *omissibility* criterion described there is a widely used criterion for dependency. It says that the head is compulsory and the dependent optional, and that from the existence of the dependent we can conclude to the existence of a head, i.e. (again for head A and dependents B,C)

(83) B,C → A

or in HCL

(84) A :- B,C.

Because of the widely accepted syntactic dependency maxim that every dependent only has one head the limitations of Prolog pose no problems to this suggestion.

A Prolog grammar rule template will therefore simply look as follows:

```
( 85) head(HeadWord, HeadClass, AuxiliaryArguments ... ) :-
         dep(DepWord1, DepClass1, AuxiliaryArguments1 ... ),
         dep(DepWord2, DepClass2, AuxiliaryArguments2 ... ).
```

If we base the dependencies on grammatical word classes such as 'VV' for main verb or 'N' for noun, then for a transitive full verb with its dependents subject and object the rule derived from template (85) will be something like:

```
( 86) head(HeadWord, 'VV', AuxiliayArguments ... ) :-
         dep(DepWord1, 'N', ... subject ... ),
         dep(DepWord2, 'N', ... object ... ).
```

I will explore this idea in the following subchapters to write and extend a small toy parser:

3.3.2 The Fundamental Prolog Program

Let us keep things as simple as possible for the start. I would like to transform the Prolog template (85) into a program along several steps:

- **Complex rule template instead of complex grammar rules in order to keep the grammar compact**: Because the `AuxiliaryArguments` in the template can easily grow complex and because additional conditions needed later are likely to add to the complexity of the template we can expect every grammar rule to become much more complex than the schematic example (86). But because grammars typically consist of a large number of rules, which would contain a lot of redundant information due to the expected complexity just mentioned, it is advisable to split the whole rule into few complex grammar templates (`head`) and many very compact non-redundant grammar rules (`depgrammar`), as shown in the following:

```
( 87) head(HeadWord, HeadClass, AuxiliaryArguments ... ) :-
         depgrammar(HeadClass, DepClass1),
         dep(DepWord1, DepClass1, AuxiliaryArguments1 ... ).

      depgrammar('VV','N').
```

- **Transitivity**: Template (85) shows a head with two dependents, e.g. a transitive verb, while the `head` template of (87) shows a head with only one

dependent. We could write several templates for mono-, di-, tri-, etc.-transitive heads. In order to keep things simple at the start we will only use the monotransitive template of (87), and tackle transitivity in 3.3.3.1 by recursion.

- **Dependents are new subheads**: This fundamental linguistic fact is most naturally expressed by recursion, which we include into the head template clause:

```
( 88) head(HeadWord, HeadClass, AuxiliaryArguments ... ) :-
          depgrammar(HeadClass, DepClass1),
          dep(DepWord1, DepClass1, AuxiliaryArguments1 ... ),
          head(DepWord1,DepClass1, NewAuxiliaryAguments1 ...).
            % Is dependent new local head ?
```

- **Building up the parsing structure backwards from the end**: One of the auxiliary arguments is a variable which collects the parsed structure and which occupies the third argument position of head. As this structure will be built up from the innermost argument towards the outside it will have to be built up backwards by iteratively binding the variables on exiting the clause - a common Prolog practice. The variable will be bound for the first time when a dependency branch reaches an end, i.e. when the head clause (88) fails. To this end we add the following head clause (89) under (88):

```
( 89) head(HeadWord,_,HeadWord, ... ).
          % No dependents found -> copy head to 3rd arg (Res)
```

  This clause also has the desirable effect that any dependent can be optional. Although e.g. the verb *to eat* is transitive, constructions where *eat* has no object are correct and have to parsable.

  In order to ensure the correct build-up of the dependency structure we have to extend the first head clause (88) as follows:

```
( 90) head(HeadWord, HeadClass, Res, ... ) :-
          depgrammar(HeadClass, DepClass1),
          dep(DepWord1, DepClass1, Res, ... ),
          head(DepWord1,DepClass1, NewRes, ...),
            % Is dependent new local head ?
          Res =.. [HeadWord,NewRes].
```

- **Searching for dependents**: The remaining argument positions will be used for the original sentence and position markers in it. The dep clause will have to find a dependent of class DepClass1 (3rd arg. pos.). Since dependency is inherently context-sensitive, we will allow any fitting word to become a dependent, irrespective of its position in the sentence. Each article in the sentence should e.g. become a dependent of each noun in the whole sentence. This results in an extremely high number of ambiguities, which is

intended (notice that this is not a bug, but a desired feature of this dependency parser at the current stage). To this end, we simply search the whole sentence from beginning to end, which is very simple:

```
( 91)
dep(DWord,DTag,Res,Sent,HPos):-  /*Start searching whole sent*/
       finddep(DWord,DTag,Res,Sent,HPos,Sent,1).

finddep(Word,Tag,Res,_,_,[Word,Tag|Rest],SearchPos). /*FOUND!*/
finddep(Word,Tag,Res,_,_,[_,_|Rest],SearchPos):- /*Search on */
       NewSearchPos is SearchPos+1,
       finddep(Word,Tag,Res,_,_,Rest,NewSearchPos).
```

The fundamental program, as printed below, is now finished. Note that recognition of ambiguity has to be explicitly forced by `fail` in this design.

```
( 92) Listing of the fundamental DEPP parser:
/* DEPENDENCY GRAMMAR V 0.21 */

test :-
     dparse(['the','ART','man','N','loves','VV','a','ART','woman','N']).
   /* This is test sentence */

dparse(Sent):-
     head(root,'ROOT',Res,Sent,0),
     print(Res), nl,
     fail.

dep(DWord,DTag,Res,Sent,HPos):-         /*Start searching whole sent*/
     finddep(DWord,DTag,Res,Sent,HPos,Sent,1).

finddep(Word,Tag,Res,_,_,[Word,Tag|Rest],SearchPos).  /* FOUND! */
finddep(Word,Tag,Res,_,_,[_,_|Rest],SearchPos) :-    /* Search on */
     NewSearchPos is SearchPos+1,
     finddep(Word,Tag,Res,_,_,Rest,NewSearchPos).

/* GRAMMAR TEMPLATES */

head(HWord,HClass,Res,Sent,HPos) :-
     depgrammar(HClass,DClass),
     dep(DWord,DClass,Res,Sent,DPos),
     head(DWord,DClass,NewRes,Sent,DPos), % Is Dependent new local Head?
     Res =.. [HWord,NewRes].
     % print(NewRes), nl.

head(HWord,_,HWord,_,_). % No dependents found -> copy Head to Res

/* GRAMMAR RULES */

depgrammar('ROOT','VV').
depgrammar('VV','N').
depgrammar('N','ART').
```

For the test sentence, which is input annotated with grammatical tags as shown in the `test` clause,

```
( 93) The man loves a woman
```

it will correctly report the solutions:

```
( 94)
| ?- test.
root(loves(man(the)))
root(loves(man(a)))
root(loves(man))
root(loves(woman(the)))
```

```
root(loves(woman(a)))
root(loves(woman))
root(loves)
root

no
```

Note that

a) transitivity is not implemented here, as explained above

b) the parser is fully context-sensitive, i.e. does not respect word-order, and

c) since all dependents can be optional due to the second head clause as shown in (89), also solutions without subject or main verb are reported.

3.3.3 Extensions to the Program

This dependency parser obviously falls short in many ways. It will be extended in the following subchapters.

3.3.3.1 Transitivity: Several Arguments

We have seen that so far the parser only accepts one dependent per head. In order to be able to treat transitive structures, one could add more head grammar templates with higher arity. A more elegant solution, however, is to keep one universal head grammar template, to which we add the ability to recursively traverse a list of dependents. This requires two extensions to the program, which depend on the original head/5 grammar template:

```
( 95) Original head/5:
head(HWord,HClass,Res,Sent,HPos) :-
     depgrammar(HClass,DClass),
     dep(DWord,DClass,Res,Sent,DPos),
     head(DWord,DClass,NewRes,Sent,DPos), % Is Dependent new local Head?
     Res =.. [HWord,NewRes].
     % print(NewRes), nl.
```

- depgrammar/2 has to report a list of dependents as its second argument (DClass in (92) and (95). Accordingly, dep/5 needs a recursive version, which traverses this list of dependents:

```
 ( 96) dep/5:
dep([],[],_,_,_). % End of Recursion
dep([DWordF|DWordR],[DTagF|DTagR],Res,Sent,HPos):-
    /*Start searching whole sent*/
    finddep(DWordF,DTagF,Res,Sent,HPos,Sent,1),
    dep(DWordR,DTagR,Res,Sent,HPos).
```

- dep/5 now reports a list of dependents, dependent words at the first position (DWord), dependent classes at the second position (DTag). Accordingly, the head/5 call in which the dependents become new head candidates (%Is Dependent new local Head ?) has to be replaced by a version which traverses the list of these head candidates, which I call headarity/5. If the list of head candidates only contains one element, the old head/5 clause is

used, otherwise the list is traversed recursively and the structure collected by *append*:

```
 ( 97) headarity/5:
headarity([DWord],[DClass],Res,Sent,Dpos) :-
    head(DWord,[DClass], Res,Sent,DPos).
headarity([DWordF|DWordR],[DClassF|DClassR],CoRes,Sent,DPos)
:-
    head(DWordF,[DClassF],Res,Sent,DPos),
    headarity(DWordR,DClassR,NewRes,Sent,DPos),
    append([NewRes],[Res],CoRes).
```

After these exensions, the program looks as follows:

```
( 98) DEPP with transitivity:
/* DEPENDENCY GRAMMAR */
/* V 0.3 with several arguments */

:- use_module(library(lists)).

test :-
     dparse(['the','ART','man','N','loves','VV','a','ART','woman','N']).
  /* This is test sentence */

dparse(Sent):-
     head(root,['ROOT'],Res,Sent,0),
     print(Res), print('.'), nl,
     fail.

dep([],[],_,_,_). % End of Recursion
dep([DWordF|DWordR],[DTagF|DTagR],Res,Sent,HPos):-
     /*Start searching whole sent*/
     finddep(DWordF,DTagF,Res,Sent,HPos,Sent,1),
     dep(DWordR,DTagR,Res,Sent,HPos).

finddep(Word,Tag,Res,_,_,[Word,Tag|Rest],SearchPos).  /* FOUND! */
finddep(Word,Tag,Res,_,_,[_,_|Rest],SearchPos) :-    /* Search on */
     NewSearchPos is SearchPos+1,
     finddep(Word,Tag,Res,_,_,Rest,NewSearchPos).

/* GRAMMAR TEMPLATES */

head(HWord,[HClass],Res,Sent,HPos) :-
     depgrammar(HClass,DClass),
     dep(DWord,DClass,Res,Sent,DPos),
     headarity(DWord,DClass,NewRes,Sent,DPos), % Is Dep. new local Head?
     Res =.. [HWord,NewRes].
     %print(Res), nl.

head(HWord,_,HWord,_,_). % No dependents found -> copy Head to Res

headarity([DWord],[DClass],Res,Sent,Dpos) :-
     head(DWord,[DClass], Res,Sent,DPos).
headarity([DWordF|DWordR],[DClassF|DClassR],CoRes,Sent,DPos) :-
     head(DWordF,[DClassF],Res,Sent,DPos),
     headarity(DWordR,DClassR,NewRes,Sent,DPos),
     append([NewRes],[Res],CoRes).

/* GRAMMAR RULES */

depgrammar('ROOT',['VV']).
depgrammar('VV',['N','N']).
depgrammar('N',['ART']).
```

For the test sentence, it correctly reports:

```
( 99) “The man loves a woman”:
| ?- test.
root(loves([man(the),man(the)])).
root(loves([man(a),man(the)])).
root(loves([man,man(the)])).
root(loves([man(the),man(a)])).
root(loves([man(a),man(a)])).
root(loves([man,man(a)])).
root(loves([man(the),man])).
root(loves([man(a),man])).
root(loves([man,man])).
root(loves([woman(the),man(the)])).
root(loves([woman(a),man(the)])).
root(loves([woman,man(the)])).
root(loves([woman(the),man(a)])).
root(loves([woman(a),man(a)])).
root(loves([woman,man(a)])).
root(loves([woman(the),man])).
root(loves([woman(a),man])).
root(loves([woman,man])).
root(loves([man(the),woman(the)])).
root(loves([man(a),woman(the)])).
root(loves([man,woman(the)])).
root(loves([man(the),woman(a)])).
root(loves([man(a),woman(a)])).
root(loves([man,woman(a)])).
root(loves([man(the),woman])).
root(loves([man(a),woman])).
root(loves([man,woman])).
root(loves([woman(the),woman(the)])).
root(loves([woman(a),woman(the)])).
root(loves([woman,woman(the)])).
root(loves([woman(the),woman(a)])).
root(loves([woman(a),woman(a)])).
root(loves([woman,woman(a)])).
root(loves([woman(the),woman])).
root(loves([woman(a),woman])).
root(loves([woman,woman])).
root(loves).
root.

no
```

3.3.3.2 Check For Complete Linkage

Two serious shortcomings of this parser are that it may allow several dependencies between the same words(e.g. allowing `root(loves([woman(the),woman(the)])).`) , and that it does not check if all the words of a sentence are linked, i.e. if the linkage is complete. It is possible to remedy this shortcoming by keeping a list of linked words, `SearchedList` at the 8th argument position in `finddep/9`. In order not to link words which are linked already, the list library predicate `member/2` only allows non-members to be linked and consecutively appends them to the list of linked words:

```
    ( 100) finddep/9:
finddep(Word,Tag,Res,_,_,[Word,Tag|Rest],SearchPos,SearchedList,NewSearchedLi
        st) :-
        /* FOUND! */
        \+(member(SearchPos,SearchedList)),
        append([SearchPos],SearchedList,NewSearchedList).
```

Some more changes to the program are needed, mainly additional arguments to manage the list of linked words, and the clause `complete_check/3` which marks

incompletely linked sentences or suppresses their printout, as the complete listing reveals:

```
     ( 101) DEPP with linkage check:
/* DEPENDENCY GRAMMAR */
/* V 0.4 with several arguments and linkage check */

:- use_module(library(lists)).

test1 :- dparse(['the','ART','man','N','loves','VV2','a','ART','woman','N']).
test2 :- dparse(['Peter','N','sleeps','VV1']).
test3 :-
     dparse(['the','ART','man','N','loves','VV2','a','ART','beautiful','ADJ','
     woman','N']).
test4 :-
     dparse(['the','ART','man','N','loves','VV2','a','ART','woman','N','with',
     'P','long','ADJ','hair','N']).
  /* These are test sentences */

dparse(Sent):-
     head(root,['ROOT'],Res,Sent,0,[],SList,FinSList),
     complete_check(Sent,FinSList,Marker),
     print(Marker),print(Res), print('. PARSED: '),
     print(FinSList),
     nl,
     fail.

complete_check(Sent,PList,'+'):- % Complete
     length(Sent,SentL),
     length(PList,PListL),
     SentL is (PListL *2),!.
% complete_check(_,_,'    -'). % Incomplete UN/COMMENT for NON/VERBOSE

dep([],[],_,_,_,SList,SList). % End of Recursion
dep([DWordF|DWordR],[DTagF|DTagR],Res,Sent,HPos,SList,NewNewSList):-
     /*Start searching whole sent*/
     finddep(DWordF,DTagF,Res,Sent,HPos,Sent,1,SList,NewSList),
     dep(DWordR,DTagR,Res,Sent,HPos,NewSList,NewNewSList).

finddep(Word,Tag,Res,_,_,[Word,Tag|Rest],SearchPos,SearchedList,NewSearchedLis
     t) :-
     /* FOUND! */
     \+(member(SearchPos,SearchedList)),
     append([SearchPos],SearchedList,NewSearchedList).
finddep(Word,Tag,Res,_,_,[_,_|Rest],SearchPos,SList,NewSList) :-    /* Search
     on */
     NewSearchPos is SearchPos+1,
     finddep(Word,Tag,Res,_,_,Rest,NewSearchPos,SList,NewSList).

/* GRAMMAR TEMPLATES */

head(HWord,[HClass],Res,Sent,HPos,SList,NewSList,FinSList) :-
     depgrammar(HClass,DClass),
     dep(DWord,DClass,Res,Sent,DPos,SList,NewSList),
      headarity(DWord,DClass,NewRes,Sent,DPos,
     NewSList,NewNewSList,AriSList,FinSList),
      %Is Dep. new local Head?
      Res =.. [HWord,NewRes].
      % print('      ? '), print(Res), print(' - '), print(NewSList), nl.

head(HWord,_,HWord,_,_,X,X,X). % No dependents found -> copy Head to Res

headarity([],[],_,_,_,_,_,AriSList,AriSList).
headarity([DWordF|DWordR],[DClassF|DClassR],
     CoRes,Sent,DPos,SList,NewSList,NewNewSList,FinSList) :-
     /* Several Args */
     head(DWordF,[DClassF],Res,Sent,DPos,SList,NewSList,AriSList),
     headarity(DWordR,DClassR,NewRes,Sent,DPos,
     NewSList,NewNewSList,AriSList,FinSList),
     append_if_nonvar(Res,NewRes,CoRes).
```

```
append_if_nonvar(X,Y,XY) :-
     nonvar(Y),
     append([X],[Y],XY).
append_if_nonvar(X,Y,X) :-
     var(Y).


/* GRAMMAR RULES */

depgrammar('ROOT',['VV2']).
depgrammar('ROOT',['VV1']).
depgrammar('VV2',['N','N']).
depgrammar('VV1',['N']).
depgrammar('N',['ART']).
depgrammar('N',['ADJ']).
depgrammar('N',['ART','ADJ']).
depgrammar('N',['P']).
depgrammar('N',['ADJ','P']).
depgrammar('N',['ART','P']).
depgrammar('N',['ART','ADJ','P']).
depgrammar('P',['N']).
```

The grammar and the collection of test sentences have also been extended. Test sentence 1 yields the expected permutations as complete linkages:

```
( 102) "The man loves a woman":
| ?- test1.
+root(loves([man(the),woman(a)])). PARSED: [4,1,5,2,3]
+root(loves([man(a),woman(the)])). PARSED: [1,4,5,2,3]
+root(loves([woman(the),man(a)])). PARSED: [4,1,2,5,3]
+root(loves([woman(a),man(the)])). PARSED: [1,4,2,5,3]

no
```

The parser also copes with free word order in more complex sentences. Test sentence 4 ("The man loves a woman with long hair.") reports 84 complete linkages:

```
| ?- test4.
+root(loves([man(the),woman(with(hair([a,long])))])). PARSED:
     [7,4,8,6,1,5,2,3]
+root(loves([man(the),woman([long,with(hair(a))])])). PARSED:
     [4,8,6,7,1,5,2,3]
+root(loves([man(the),woman([a,with(hair(long))])])). PARSED:
     [7,8,6,4,1,5,2,3]
+root(loves([man(a),woman(with(hair([the,long])))])). PARSED:
     [7,1,8,6,4,5,2,3]
+root(loves([man(a),woman([long,with(hair(the))])])). PARSED:
     [1,8,6,7,4,5,2,3]
+root(loves([man(a),woman([the,with(hair(long))])])). PARSED:
     [7,8,6,1,4,5,2,3]
+root(loves([man(long),woman([the,with(hair(a))])])). PARSED:
     [4,8,6,1,7,5,2,3]
+root(loves([man(long),woman([a,with(hair(the))])])). PARSED:
     [1,8,6,4,7,5,2,3]
+root(loves([man([the,long]),woman(with(hair(a)))])). PARSED:
     [4,8,6,7,1,5,2,3]
+root(loves([man([the,long]),woman([a,with(hair)])])). PARSED:
     [8,6,4,7,1,5,2,3]
+root(loves([man([a,long]),woman(with(hair(the)))])). PARSED:
     [1,8,6,7,4,5,2,3]
+root(loves([man([a,long]),woman([the,with(hair)])])). PARSED:
     [8,6,1,7,4,5,2,3]
+root(loves([man,woman([the,with(hair([a,long]))])])). PARSED:
     [7,4,8,6,1,5,2,3]
+root(loves([man,woman([a,with(hair([the,long]))])])). PARSED:
     [7,1,8,6,4,5,2,3]
+root(loves([man(the),hair(with(woman([a,long])))])). PARSED:
     [7,4,5,6,1,8,2,3]
+root(loves([man(the),hair([long,with(woman(a))])])). PARSED:
     [4,5,6,7,1,8,2,3]
```

```
+root(loves([man(the),hair([a,with(woman(long))])])). PARSED:
     [7,5,6,4,1,8,2,3]
+root(loves([man(a),hair(with(woman([the,long])))])). PARSED:
     [7,1,5,6,4,8,2,3]
+root(loves([man(a),hair([long,with(woman(the))])])). PARSED:
     [1,5,6,7,4,8,2,3]
+root(loves([man(a),hair([the,with(woman(long))])])). PARSED:
     [7,5,6,1,4,8,2,3]
+root(loves([man(long),hair([the,with(woman(a))])])). PARSED:
     [4,5,6,1,7,8,2,3]
+root(loves([man(long),hair([a,with(woman(the))])])). PARSED:
     [1,5,6,4,7,8,2,3]
+root(loves([man([the,long]),hair(with(woman(a)))])). PARSED:
     [4,5,6,7,1,8,2,3]
+root(loves([man([the,long]),hair([a,with(woman)])])). PARSED:
     [5,6,4,7,1,8,2,3]
+root(loves([man([a,long]),hair(with(woman(the)))])). PARSED:
     [1,5,6,7,4,8,2,3]
+root(loves([man([a,long]),hair([the,with(woman)])])). PARSED:
     [5,6,1,7,4,8,2,3]
+root(loves([man,hair([the,with(woman([a,long]))])])). PARSED:
     [7,4,5,6,1,8,2,3]
+root(loves([man,hair([a,with(woman([the,long]))])])). PARSED:
     [7,1,5,6,4,8,2,3]
+root(loves([woman(the),man(with(hair([a,long])))])). PARSED:
     [7,4,8,6,1,2,5,3]
+root(loves([woman(the),man([long,with(hair(a))])])). PARSED:
     [4,8,6,7,1,2,5,3]
+root(loves([woman(the),man([a,with(hair(long))])])). PARSED:
     [7,8,6,4,1,2,5,3]
+root(loves([woman(a),man(with(hair([the,long])))])). PARSED:
     [7,1,8,6,4,2,5,3]
+root(loves([woman(a),man([long,with(hair(the))])])). PARSED:
     [1,8,6,7,4,2,5,3]
+root(loves([woman(a),man([the,with(hair(long))])])). PARSED:
     [7,8,6,1,4,2,5,3]
+root(loves([woman(long),man([the,with(hair(a))])])). PARSED:
     [4,8,6,1,7,2,5,3]
+root(loves([woman(long),man([a,with(hair(the))])])). PARSED:
     [1,8,6,4,7,2,5,3]
+root(loves([woman([the,long]),man(with(hair(a)))])). PARSED:
     [4,8,6,7,1,2,5,3]
+root(loves([woman([the,long]),man([a,with(hair)])])). PARSED:
     [8,6,4,7,1,2,5,3]
+root(loves([woman([a,long]),man(with(hair(the)))])). PARSED:
     [1,8,6,7,4,2,5,3]
+root(loves([woman([a,long]),man([the,with(hair)])])). PARSED:
     [8,6,1,7,4,2,5,3]
+root(loves([woman,man([the,with(hair([a,long]))])])). PARSED:
     [7,4,8,6,1,2,5,3]
+root(loves([woman,man([a,with(hair([the,long]))])])). PARSED:
     [7,1,8,6,4,2,5,3]
+root(loves([woman(the),hair(with(man([a,long])))])). PARSED:
     [7,4,2,6,1,8,5,3]
+root(loves([woman(the),hair([long,with(man(a))])])). PARSED:
     [4,2,6,7,1,8,5,3]
+root(loves([woman(the),hair([a,with(man(long))])])). PARSED:
     [7,2,6,4,1,8,5,3]
+root(loves([woman(a),hair(with(man([the,long])))])). PARSED:
     [7,1,2,6,4,8,5,3]
+root(loves([woman(a),hair([long,with(man(the))])])). PARSED:
     [1,2,6,7,4,8,5,3]
+root(loves([woman(a),hair([the,with(man(long))])])). PARSED:
     [7,2,6,1,4,8,5,3]
+root(loves([woman(long),hair([the,with(man(a))])])). PARSED:
     [4,2,6,1,7,8,5,3]
+root(loves([woman(long),hair([a,with(man(the))])])). PARSED:
     [1,2,6,4,7,8,5,3]
+root(loves([woman([the,long]),hair(with(man(a)))])). PARSED:
     [4,2,6,7,1,8,5,3]
+root(loves([woman([the,long]),hair([a,with(man)])])). PARSED:
     [2,6,4,7,1,8,5,3]
```

```
+root(loves([woman([a,long]),hair(with(man(the)))])). PARSED:
      [1,2,6,7,4,8,5,3]
+root(loves([woman([a,long]),hair([the,with(man)])])). PARSED:
      [2,6,1,7,4,8,5,3]
+root(loves([woman,hair([the,with(man([a,long]))])])). PARSED:
      [7,4,2,6,1,8,5,3]
+root(loves([woman,hair([a,with(man([the,long]))])])). PARSED:
      [7,1,2,6,4,8,5,3]
+root(loves([hair(the),man(with(woman([a,long])))])). PARSED:
      [7,4,5,6,1,2,8,3]
+root(loves([hair(the),man([long,with(woman(a))])])). PARSED:
      [4,5,6,7,1,2,8,3]
+root(loves([hair(the),man([a,with(woman(long))])])). PARSED:
      [7,5,6,4,1,2,8,3]
+root(loves([hair(a),man(with(woman([the,long])))])). PARSED:
      [7,1,5,6,4,2,8,3]
+root(loves([hair(a),man([long,with(woman(the))])])). PARSED:
      [1,5,6,7,4,2,8,3]
+root(loves([hair(a),man([the,with(woman(long))])])). PARSED:
      [7,5,6,1,4,2,8,3]
+root(loves([hair(long),man([the,with(woman(a))])])). PARSED:
      [4,5,6,1,7,2,8,3]
+root(loves([hair(long),man([a,with(woman(the))])])). PARSED:
      [1,5,6,4,7,2,8,3]
+root(loves([hair([the,long]),man(with(woman(a)))])). PARSED:
      [4,5,6,7,1,2,8,3]
+root(loves([hair([the,long]),man([a,with(woman)])])). PARSED:
      [5,6,4,7,1,2,8,3]
+root(loves([hair([a,long]),man(with(woman(the)))])). PARSED:
      [1,5,6,7,4,2,8,3]
+root(loves([hair([a,long]),man([the,with(woman)])])). PARSED:
      [5,6,1,7,4,2,8,3]
+root(loves([hair,man([the,with(woman([a,long]))])])). PARSED:
      [7,4,5,6,1,2,8,3]
+root(loves([hair,man([a,with(woman([the,long]))])])). PARSED:
      [7,1,5,6,4,2,8,3]
+root(loves([hair(the),woman(with(man([a,long])))])). PARSED:
      [7,4,2,6,1,5,8,3]
+root(loves([hair(the),woman([long,with(man(a))])])). PARSED:
      [4,2,6,7,1,5,8,3]
+root(loves([hair(the),woman([a,with(man(long))])])). PARSED:
      [7,2,6,4,1,5,8,3]
+root(loves([hair(a),woman(with(man([the,long])))])). PARSED:
      [7,1,2,6,4,5,8,3]
+root(loves([hair(a),woman([long,with(man(the))])])). PARSED:
      [1,2,6,7,4,5,8,3]
+root(loves([hair(a),woman([the,with(man(long))])])). PARSED:
      [7,2,6,1,4,5,8,3]
+root(loves([hair(long),woman([the,with(man(a))])])). PARSED:
      [4,2,6,1,7,5,8,3]
+root(loves([hair(long),woman([a,with(man(the))])])). PARSED:
      [1,2,6,4,7,5,8,3]
+root(loves([hair([the,long]),woman(with(man(a)))])). PARSED:
      [4,2,6,7,1,5,8,3]
+root(loves([hair([the,long]),woman([a,with(man)])])). PARSED:
      [2,6,4,7,1,5,8,3]
+root(loves([hair([a,long]),woman(with(man(the)))])). PARSED:
      [1,2,6,7,4,5,8,3]
+root(loves([hair([a,long]),woman([the,with(man)])])). PARSED:
      [2,6,1,7,4,5,8,3]
+root(loves([hair,woman([the,with(man([a,long]))])])). PARSED:
      [7,4,2,6,1,5,8,3]
+root(loves([hair,woman([a,with(man([the,long]))])])). PARSED:
      [7,1,2,6,4,5,8,3]

no
```

But the correct reading only appears at the third position.

3.3.3.3 Restrict Context-Sensitivity

A parser with completely free word order and unrestricted context-sensitivity is rather of academic than any practical significance. The third and longest extension to this parser spots sentences which are non-projective, i.e. in which dependencies cross.

3.3.3.3.1 New Search Strategy for Dependents

Until now searching for dependents meant traversing the sentence from the beginning until a suitable dependent was found by `finddep/9`. The search strategy used here starts at the position of the head and looks 1 word to the left and then to the right, then 2 words, and so on. Most local, i.e. most adjacent dependents are found first. This search strategy also reflects that linear proximity and semantic proximity usually correspond. These are the new predicates `finddep_adjac/11` and `finddep_adjac_lr/11`:

```
     ( 103) finddep_adjac/11 and finddep_adjac_lr/11:
finddep_adjac(DWordF,DTagF,Res,Sent,HPos,DPos,DList,
     Sent,DeltaSPos,SList,NewSList):- %FindLeft
     LeftSPos is (HPos-DeltaSPos),
     LeftSPos > 0,
     finddep_adjac_lr(DWordF,DTagF,Res,Sent,HPos,DPos,DList,
     Sent,LeftSPos,SList,NewSList).
finddep_adjac(DWordF,DTagF,Res,Sent,HPos,DPos,DList,
     Sent,DeltaSPos,SList,NewSList):- %FindRight
     RightSPos is (HPos+DeltaSPos),
     finddep_adjac_lr(DWordF,DTagF,Res,Sent,HPos,DPos,DList,
     Sent,RightSPos,SList,NewSList).
finddep_adjac(DWordF,DTagF,Res,Sent,HPos,DPos,DList,
     Sent,DeltaSPos,SList,NewSList):- %Search on
     length(Sent,Border),
     (HPos+DeltaSPos) < Border,
     NewDeltaSPos is (DeltaSPos+1),
     finddep_adjac(DWordF,DTagF,Res,Sent,HPos,DPos,DList,
     Sent,NewDeltaSPos,SList,NewSList).

finddep_adjac_lr(Word,Tag,Res,Sent,HPos,SPos,DPosList,Sent,SPos,SList,NewSList
     ):-
     SSPosM is ((SPos*2)-1),
     SSPos is (SPos*2),
     nth(SSPos,Sent,Tag),
     nth(SSPosM,Sent,Word),
     \+(member(SPos,SList)),
     append([SPos],SList,NewSList).
```

3.3.3.3.2 Check for Crossing Dependencies

In order to be able to check if dependencies cross it is necessary to keep a list of dependencies. This is done by the variables `DList`, `NewDList`, etc. which work very similar to `SList`, `NewSList`, etc. Due to the complex interactions they both use up to four argument positions, e.g. in the predicate `headarity/15`:

```
     ( 104) Example of complex predicate:
headarity([],[],_,_,_,_,_,_,DL,DL,_,_,SL,SL,_).
headarity([HWordF|HWordR],[HClassF|HClassR],CoRes,Sent,[HPosF|HPosR],
     DPos,DList,NewDList,NewNewDList,FinDList,SList,NewSList,NewNewSList,FinSL
     ist, PFLAG) :-
```

```
     /* Several Args */
     head(HWordF,[HClassF],Res,Sent,HPosF,DPos,DList,NewDList,AriDList,
     SList,NewSList,AriSList,PFLAG),
     headarity(HWordR,HClassR,NewRes,Sent,HPosR,NewDPos,
     NewDList,NewNewDList,AriDList,FinDList,
     NewSList,NewNewSList,AriSList,FinSList,PFLAG),
     append_if_nonvar(Res,NewRes,CoRes).
```

After new dependents are found in dep/12, they can now be checked for non-projectivity by cross_check/3.

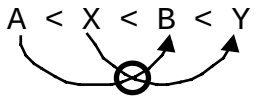
```
     ( 105) Calling cross_check/3 from dep/12:
dep(_,[],[],_,_,_,_,DList,DList,SList,SList,_). % End of Recursion
dep(HWord,[DWordF|DWordR],[DTagF|DTagR],Res,Sent,HPos,[DPosF|DPosR],DList,AriD
     List, SList,AriSList,PFLAG):-
     /*Start searching whole sent*/
     finddep_adjac(DWordF,DTagF,Res,Sent,HPos,DPosF,DList,Sent,1,SList,NewSLis
     t),
     append([[HPos,DPosF]],DList,NewDList),
     cross_check(NewDList,[HWord,DWordF],PFLAG),
     dep(HWord,DWordR,DTagR,Res,Sent,HPos,DPosR,NewDList,AriDList,
     NewSList,AriSList,PFLAG).
```

cross_check/3 chops the first (i.e. most recent) head-dependent position pair from the list of dependencies and checks if it crosses any of the existing dependency links. If so, the variable PFLAG is instantiated to the value 'YES'.

```
     ( 106) cross_check/3, cross_check_f/4, cross_checking/4:
     cross_check([LF|LR],Res,PFLAG) :-
          cross_check_f(LF,LR,Res,PFLAG),!.

     cross_check_f([_,_],[],_,_). % End of Recursion

     cross_check_f([A,B],[[X,Y]|R],Res,' YES'):-
          nonvar(A),nonvar(B),nonvar(X),nonvar(Y),
          cross_checking(A,B,X,Y),
          print('!!NON_PROJECTIVE!! '), print(Res), nl.
     cross_check_f([A,B],[[X,Y]|R],Res,PFLAG):-
          cross_check_f([A,B],R,Res,PFLAG).

     cross_checking(A,B,X,Y):-
          A<X, B<Y, X<B.
     cross_checking(A,B,X,Y):-
          B<X, A<Y, X<A.
     cross_checking(A,B,X,Y):-
          A<Y, B<X, Y<B.
     cross_checking(A,B,X,Y):-
          B<Y, A<X, Y<A.
     cross_checking(A,B,X,Y):-
          X<A, Y<B, A<Y.
     cross_checking(A,B,X,Y):-
          X<B, Y<A, B<Y.
     cross_checking(A,B,X,Y):-
          X<B, Y<A, B<Y.
     cross_checking(A,B,X,Y):-
          Y<B, X<A, B<X.
```

If A and B are linked and X and Y are linked we get a structure like in (107):

(107) Detection of crossing links by cross_checking/4:



There are 8 possible permutations of such crossing structures, as captured in cross_checking/4. Now the parser correctly detects non-projective parses. Every

attempt of a non-projective parse results in a online warning ("!!NON_PROJECTIVE!!") and the variable PFLAG which is reported to `dparse/1` for printout is instantiated to 'YES'. A verbose output for test sentence 1 "The man loves a woman" is printed in (108). Note that the third complete reading (`+root(loves([woman(a),man(the)]))`) does not contain crossing links. It is incorrect because subject and object positions are mistaken. Since the parser shown here is based on part-of-speech word class rather than functional classes, both the subject and the object simply have a noun tag 'N'. The search algorithm will bring up the correct subject first in assertive sentences and yes/no-questions, since it is the noun closest to the verb.

```
( 108) Verbose output of "The man loves a woman":
| ?- test1.
[the,ART,man,N,loves,VV2,a,ART,woman,N]
+root(loves([man(the),woman(a)])). PARSED: [4,1,5,2,3]. DEP.list:
     [[5,4],[2,1],[3,5],[3,2],[0,3]]. NON_PROJ: _183
    -root(loves([man(the),woman(a)])). PARSED: [4,1,5,2,3]. DEP.list:
     [[5,4],[2,1],[3,5],[3,2],[0,3]]. NON_PROJ: _183
    -root(loves([man(the),woman])). PARSED: [1,5,2,3]. DEP.list:
     [[2,1],[3,5],[3,2],[0,3]]. NON_PROJ: _183
!!NON_PROJECTIVE!! [man,a]
!!NON_PROJECTIVE!! [woman,the]
+root(loves([man(a),woman(the)])). PARSED: [1,4,5,2,3]. DEP.list:
     [[5,1],[2,4],[3,5],[3,2],[0,3]]. NON_PROJ:  YES
    -root(loves([man(a),woman(the)])). PARSED: [1,4,5,2,3]. DEP.list:
     [[5,1],[2,4],[3,5],[3,2],[0,3]]. NON_PROJ:  YES
!!NON_PROJECTIVE!! [woman,the]
!!NON_PROJECTIVE!! [woman,the]
!!NON_PROJECTIVE!! [woman,the]
    -root(loves([man(a),woman])). PARSED: [4,5,2,3]. DEP.list:
     [[2,4],[3,5],[3,2],[0,3]]. NON_PROJ:  YES
!!NON_PROJECTIVE!! [man,a]
!!NON_PROJECTIVE!! [man,a]
!!NON_PROJECTIVE!! [man,a]
    -root(loves([man,woman(a)])). PARSED: [4,5,2,3]. DEP.list:
     [[5,4],[3,5],[3,2],[0,3]]. NON_PROJ: _183
!!NON_PROJECTIVE!! [woman,the]
    -root(loves([man,woman(the)])). PARSED: [1,5,2,3]. DEP.list:
     [[5,1],[3,5],[3,2],[0,3]]. NON_PROJ:  YES
!!NON_PROJECTIVE!! [woman,the]
!!NON_PROJECTIVE!! [woman,the]
!!NON_PROJECTIVE!! [woman,the]
    -root(loves([man,woman])). PARSED: [5,2,3]. DEP.list:
     [[3,5],[3,2],[0,3]]. NON_PROJ: _183
+root(loves([woman(a),man(the)])). PARSED: [1,4,2,5,3]. DEP.list:
     [[2,1],[5,4],[3,2],[3,5],[0,3]]. NON_PROJ: _183
    -root(loves([woman(a),man(the)])). PARSED: [1,4,2,5,3]. DEP.list:
     [[2,1],[5,4],[3,2],[3,5],[0,3]]. NON_PROJ: _183
    -root(loves([woman(a),man])). PARSED: [4,2,5,3]. DEP.list:
     [[5,4],[3,2],[3,5],[0,3]]. NON_PROJ: _183
!!NON_PROJECTIVE!! [woman,the]
!!NON_PROJECTIVE!! [man,a]
+root(loves([woman(the),man(a)])). PARSED: [4,1,2,5,3]. DEP.list:
     [[2,4],[5,1],[3,2],[3,5],[0,3]]. NON_PROJ:  YES
    -root(loves([woman(the),man(a)])). PARSED: [4,1,2,5,3]. DEP.list:
     [[2,4],[5,1],[3,2],[3,5],[0,3]]. NON_PROJ:  YES
!!NON_PROJECTIVE!! [man,a]
!!NON_PROJECTIVE!! [man,a]
!!NON_PROJECTIVE!! [man,a]
    -root(loves([woman(the),man])). PARSED: [1,2,5,3]. DEP.list:
     [[5,1],[3,2],[3,5],[0,3]]. NON_PROJ:  YES
!!NON_PROJECTIVE!! [woman,the]
!!NON_PROJECTIVE!! [woman,the]
```

```
!!NON_PROJECTIVE!! [woman,the]
    -root(loves([woman,man(the)])). PARSED: [1,2,5,3]. DEP.list:
     [[2,1],[3,2],[3,5],[0,3]]. NON_PROJ: _183
!!NON_PROJECTIVE!! [man,a]
    -root(loves([woman,man(a)])). PARSED: [4,2,5,3]. DEP.list:
     [[2,4],[3,2],[3,5],[0,3]]. NON_PROJ:  YES
!!NON_PROJECTIVE!! [man,a]
!!NON_PROJECTIVE!! [man,a]
!!NON_PROJECTIVE!! [man,a]
    -root(loves([woman,man])). PARSED: [2,5,3]. DEP.list:
     [[3,2],[3,5],[0,3]]. NON_PROJ: _183
    -root(loves). PARSED: [3]. DEP.list: [[0,3]]. NON_PROJ: _183
    -root. PARSED: []. DEP.list: []. NON_PROJ: _183

no
```

3.3.3.3.3 Fast Projective Parsing

In 3.3.3.3.2 we have seen that the correct reading of test sentence 4 is only the third one. Parsing for all 84 complete linkages takes about a minute. Most dependency parsing algorithms are NP-complete in the worst case. In a practically usable parser it is therefore important to make sure that this worst case scenario will occur as infrequently as possible. The search algorithm developed in 3.3.3.3.1 helps to come up quickly with the most local and therefore often most probable parses. It does not help to yield the correct reading of test sentence 4 first, because of the order of the grammar rules.

As a first very crude approximation one can suggest that if projective parses can be found the non-projective parses are usually incorrect and that the first projective parse will be the most likely one. The new predicate `fastparse/1` stops searching when the first complete and projective reading has been found. A verbose output of parsing test sentence 4 “The man loves a woman with long hair” with `fastparse/1`, which only takes one or two seconds, is as follows:

`( 109) Verbose fastparse of “The man loves a woman with long hair”:`

```
| ?- ftest4.
[the,ART,man,N,loves,VV2,a,ART,woman,N,with,P,long,ADJ,hair,N]
    -root(loves([man(the),woman(a)])). PARSED: [4,1,5,2,3]. DEP.list:
     [[5,4],[2,1],[3,5],[3,2],[0,3]]. NON_PROJ: _203
    -root(loves([man(the),woman(long)])). PARSED: [7,1,5,2,3]. DEP.list:
     [[5,7],[2,1],[3,5],[3,2],[0,3]]. NON_PROJ: _203
    -root(loves([man(the),woman([a,long])])). PARSED: [7,4,1,5,2,3].
     DEP.list: [[5,7],[5,4],[2,1],[3,5],[3,2],[0,3]]. NON_PROJ: _203
!!NON_PROJECTIVE!! [hair,a]
    -root(loves([man(the),woman(with(hair(a)))])). PARSED:
     [4,8,6,1,5,2,3]. DEP.list:
     [[8,4],[6,8],[5,6],[2,1],[3,5],[3,2],[0,3]]. NON_PROJ:  YES
    -root(loves([man(the),woman(with(hair(long)))])). PARSED:
     [7,8,6,1,5,2,3]. DEP.list:
     [[8,7],[6,8],[5,6],[2,1],[3,5],[3,2],[0,3]]. NON_PROJ: _203
!!NON_PROJECTIVE!! [hair,a]
+root(loves([man(the),woman(with(hair([a,long])))])). PARSED:
     [7,4,8,6,1,5,2,3]. DEP.list:
     [[8,7],[8,4],[6,8],[5,6],[2,1],[3,5],[3,2],[0,3]]. NON_PROJ:  YES
    -root(loves([man(the),woman(with(hair([a,long])))])). PARSED:
     [7,4,8,6,1,5,2,3]. DEP.list:
     [[8,7],[8,4],[6,8],[5,6],[2,1],[3,5],[3,2],[0,3]]. NON_PROJ:  YES
!!NON_PROJECTIVE!! [hair,a]
!!NON_PROJECTIVE!! [hair,a]
```

```
        -root(loves([man(the),woman(with(hair))])). PARSED: [8,6,1,5,2,3].
         DEP.list: [[6,8],[5,6],[2,1],[3,5],[3,2],[0,3]]. NON_PROJ: _203
        -root(loves([man(the),woman(with)])). PARSED: [6,1,5,2,3]. DEP.list:
         [[5,6],[2,1],[3,5],[3,2],[0,3]]. NON_PROJ: _203
     !!NON_PROJECTIVE!! [with,hair]
     !!NON_PROJECTIVE!! [hair,a]
     +root(loves([man(the),woman([long,with(hair(a))])])). PARSED:
          [4,8,6,7,1,5,2,3]. DEP.list:
          [[8,4],[6,8],[5,6],[5,7],[2,1],[3,5],[3,2],[0,3]]. NON_PROJ:  YES
        -root(loves([man(the),woman([long,with(hair(a))])])). PARSED:
          [4,8,6,7,1,5,2,3]. DEP.list:
          [[8,4],[6,8],[5,6],[5,7],[2,1],[3,5],[3,2],[0,3]]. NON_PROJ:  YES
     !!NON_PROJECTIVE!! [hair,a]
     !!NON_PROJECTIVE!! [hair,a]
     !!NON_PROJECTIVE!! [hair,a]
        -root(loves([man(the),woman([long,with(hair)])])). PARSED:
          [8,6,7,1,5,2,3]. DEP.list:
          [[6,8],[5,6],[5,7],[2,1],[3,5],[3,2],[0,3]]. NON_PROJ:  YES
        -root(loves([man(the),woman([long,with])])). PARSED: [6,7,1,5,2,3].
         DEP.list: [[5,6],[5,7],[2,1],[3,5],[3,2],[0,3]]. NON_PROJ: _203
     +root(loves([man(the),woman([a,with(hair(long))])])). PARSED:
          [7,8,6,4,1,5,2,3]. DEP.list:
          [[8,7],[6,8],[5,6],[5,4],[2,1],[3,5],[3,2],[0,3]]. NON_PROJ: _203

     yes
```

The parser has changed and grown quite complex after these changes. This is the full listing:

```
     ( 110) DEPP Version 0.5:
/* DEPENDENCY GRAMMAR */
/* Dependency Existence Parser DEPP V 0.5*/

:- use_module(library(lists)).

test1 :- dparse(['the','ART','man','N','loves','VV2','a','ART','woman','N']).
test2 :- dparse(['Peter','N','sleeps','VV1']).
test3 :-
     dparse(['the','ART','man','N','loves','VV2','a','ART','beautiful','ADJ','
     woman','N']).
test4 :-
     dparse(['the','ART','man','N','loves','VV2','a','ART','woman','N','with',
     'P','long','ADJ','hair','N']).
  /* These are test sentences */

dparse(Sent):-
     head(root,['ROOT'],Res,Sent,0,DPos,[],DList,FinDList,[],SList,FinSList,PF
     LAG),
     complete_check(Sent,FinSList,Marker),
     print(Marker),print(Res),
     print('. PARSED: '), print(FinSList),
     print('. DEP.list: '), print(FinDList),
     print('. NON_PROJ: '), print(PFLAG),
     nl,
     fail.

complete_check(Sent,PList,'+'):- %Complete
     length(Sent,SentL),
     length(PList,PListL),
     SentL is (PListL *2).
complete_check(_,_,'    -'). % Incomplete UN/COMMENT for NON/VERBOSE

cross_check([LF|LR],Res,PFLAG) :-
     cross_check_f(LF,LR,Res,PFLAG),!.

cross_check_f([_,_],[],_,_). % End of Recursion

cross_check_f([A,B],[[X,Y]|R],Res,' YES'):-
     nonvar(A),nonvar(B),nonvar(X),nonvar(Y),
     cross_checking(A,B,X,Y),
     print('!!NON_PROJECTIVE!! '), print(Res), nl.
```

```
cross_check_f([A,B],[[X,Y]|R],Res,PFLAG):-
     cross_check_f([A,B],R,Res,PFLAG).

cross_checking(A,B,X,Y):-
     A<X, B<Y, X<B.
cross_checking(A,B,X,Y):-
     B<X, A<Y, X<A.
cross_checking(A,B,X,Y):-
     A<Y, B<X, Y<B.
cross_checking(A,B,X,Y):-
     B<Y, A<X, Y<A.
cross_checking(A,B,X,Y):-
     X<A, Y<B, A<Y.
cross_checking(A,B,X,Y):-
     X<B, Y<A, B<Y.
cross_checking(A,B,X,Y):-
     X<B, Y<A, B<Y.
cross_checking(A,B,X,Y):-
     Y<B, X<A, B<X.

dep(_,[],[],_,_,_,_,DList,DList,SList,SList,_). % End of Recursion
dep(HWord,[DWordF|DWordR],[DTagF|DTagR],Res,Sent,HPos,[DPosF|DPosR],DList,AriD
     List, SList,AriSList,PFLAG):-
     /*Start searching whole sent*/
     finddep_adjac(DWordF,DTagF,Res,Sent,HPos,DPosF,DList,Sent,1,SList,NewSLis
     t),
     append([[HPos,DPosF]],DList,NewDList),
     cross_check(NewDList,[HWord,DWordF],PFLAG),
     dep(HWord,DWordR,DTagR,Res,Sent,HPos,DPosR,NewDList,AriDList,
     NewSList,AriSList,PFLAG).

finddep_adjac(DWordF,DTagF,Res,Sent,HPos,DPos,DList,
     Sent,DeltaSPos,SList,NewSList):-%FindLeft
     LeftSPos is (HPos-DeltaSPos),
     LeftSPos > 0,
     finddep_adjac_lr(DWordF,DTagF,Res,Sent,HPos,DPos,DList,
     Sent,LeftSPos,SList,NewSList).
finddep_adjac(DWordF,DTagF,Res,Sent,HPos,DPos,DList,
     Sent,DeltaSPos,SList,NewSList):-%FindRight
     RightSPos is (HPos+DeltaSPos),
     finddep_adjac_lr(DWordF,DTagF,Res,Sent,HPos,DPos,DList,
     Sent,RightSPos,SList,NewSList).
finddep_adjac(DWordF,DTagF,Res,Sent,HPos,DPos,DList,
     Sent,DeltaSPos,SList,NewSList):- %Search on
     length(Sent,Border),
     (HPos+DeltaSPos) < Border,
     NewDeltaSPos is (DeltaSPos+1),
     finddep_adjac(DWordF,DTagF,Res,Sent,HPos,DPos,DList,
     Sent,NewDeltaSPos,SList,NewSList).

finddep_adjac_lr(Word,Tag,Res,Sent,HPos,SPos,DPosList,Sent,SPos,SList,NewSList
     ):-
     SSPosM is ((SPos*2)-1),
     SSPos is (SPos*2),
     nth(SSPos,Sent,Tag),
     nth(SSPosM,Sent,Word),
     \+(member(SPos,SList)),
     append([SPos],SList,NewSList).

/* GRAMMAR TEMPLATES */

head(HWord,[HClass],Res,Sent,HPos,DPosL,DList,NewDList,FinDList,
     SList,NewSList,FinSList,PFLAG):-
     depgrammar(HClass,DClassL),
     dep(HWord,DWordL,DClassL,Res,Sent,HPos,DPosL,DList,NewDList,
     SList,NewSList,PFLAG),
     headarity(DWordL,DClassL,NewRes,Sent,DPosL,NewDPos,
     NewDList,NewNewDList,AriDlist,FinDList,
     NewSList,NewNewSList,AriSList,FinSList,PFLAG),
     %Is Dep. new local Head?
     Res =.. [HWord,NewRes].
     %print('      ? '), print(Res), print(' - '), print(NewSList),
```

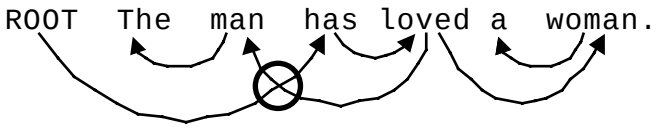
```
      %print(NewDList), print(PFLAG),nl.
      % UN/COMMENT above for NON/VERBOSE

head(HWord,_,HWord,_,_,_,DL,DL,DL,SL,SL,SL,_). % No dependents found -> copy
      Head to Res

headarity([],[],_,_,_,_,_,_,DL,DL,_,_,AriSList,AriSList,_).
headarity([HWordF|HWordR],[HClassF|HClassR], CoRes,Sent,[HPosF|HPosR],
      DPos,DList,NewDList,NewNewDList,FinDList,SList,NewSList,NewNewSList,FinSL
      ist,PFLAG) :-
      /* Several Args */
      head(HWordF,[HClassF],Res,Sent,HPosF,DPos,DList,NewDList,AriDList,
      SList,NewSList,AriSList,PFLAG),
      headarity(HWordR,HClassR,NewRes,Sent,HPosR,NewDPos,
      NewDList,NewNewDList,AriDList,FinDList,
      NewSList,NewNewSList,AriSList,FinSList,PFLAG),
      append_if_nonvar(Res,NewRes,CoRes).

append_if_nonvar(X,Y,XY) :-
      nonvar(Y),
      append([X],[Y],XY).
append_if_nonvar(X,Y,X) :-
      var(Y).


/* GRAMMAR RULES */

depgrammar('ROOT',['VV2']).
depgrammar('ROOT',['VV1']).
depgrammar('VV2',['N','N']).
depgrammar('VV1',['N']).
depgrammar('N',['ART']).
depgrammar('N',['ADJ']).
depgrammar('N',['ART','ADJ']).
depgrammar('N',['P']).
depgrammar('N',['ADJ','P']).
depgrammar('N',['ART','P']).
depgrammar('N',['ART','ADJ','P']).
depgrammar('P',['N']).
```

3.3.3.4 Argument Composition and Translation

Let us consider the following sentence:

(111) The man has loved a woman.

The most intuitive dependency parse links root to the inflected auxiliary verb, this in turn to the main verb, which links to the subject and object, since the main verb carries the valency information. This yields a non-projective structure, however:



(112)

The parser will report this structure, along with several other non-projective ones, in which the nouns link to the wrong articles. Since all parses are non-projective, however, it is difficult to exclude the incorrect parses at the first sight. It is also unsatisfactory that “The man has loved a woman” should be non-projective, while “The man loved a woman” is not.

We have seen in 2.3.2.5.5 that it is difficult to judge whether AUX (the syntactic core carrying INFL) or the main verb (the semantic core) should be head of the other. From a functional perspective, as I have discussed also in 2.3.2.5.5 and 2.3.8.3, the auxiliary verb is rather a marker on the main verb (HPSG [Pollard & Sag 1994]), or a part of it (Tesnière [1959]). A suggestion described in [Borsley 1996: 88-90], so-called *argument composition*, is to compose one semantic argument of both the auxiliary and main verb together.

3.3.3.4.1 Argument Composition

Only minimal changes are needed to implement the simplest form of argument composition: concomitance (cf. 2.2.3.2) instead of dependency for selected links. These links are licensed by `congrammar/2`, which works exactly like `depgrammar/2`. The call for `congrammar/2` comes from an additional and minimally modified `head/13` predicate:

```
     ( 113) Changes for Argument Composition:
/* GRAMMAR TEMPLATES */

head(HWord,[HClass],Res,Sent,HPos,DPosL,DList,NewDList,FinDList,
     SList,NewSList,FinSList,PFLAG):- %Normal Dependency
     depgrammar(HClass,DClassL),
     dep(HWord,DWordL,DClassL,Res,Sent,HPos,DPosL,DList,NewDList,
     SList,NewSList,PFLAG),
     headarity(DWordL,DClassL,NewRes,Sent,DPosL,NewDPos,
     NewDList,NewNewDList,AriDlist,FinDList,
     NewSList,NewNewSList,AriSList,FinSList,PFLAG),
     %Is Dep. new local Head?
     Res =.. [HWord,NewRes].
     %print('      ? '), print(Res), print(' - '), print(NewSList),
     %print(NewDList), print(PFLAG),nl.

head(HWord,[HClass],[HWord,NewRes],Sent,HPos,DPosL,DList,NewDList,FinDList,
     SList,NewSList,FinSList,PFLAG):- %Argument compostition
     congrammar(HClass,DClassL),
     dep(HWord,DWordL,DClassL,Res,Sent,HPos,DPosL,DList,NewDList,
     SList,NewSList,PFLAG),
     headarity(DWordL,DClassL,NewRes,Sent,DPosL,NewDPos,
     NewDList,NewNewDList,AriDlist,FinDList,
     NewSList,NewNewSList,AriSList,FinSList,PFLAG).
     %Is Dep. new local Head?
     %print('      ? '), print(Res), print(' - '), print(NewSList),
     %print(NewDList), print(PFLAG),nl.

head(HWord,_,HWord,_,_,_,DL,DL,DL,SL,SL,SL,_). % No dependents found -> copy
     Head to Res

[...]

/* GRAMMAR RULES */

depgrammar('ROOT',['VV2']).
depgrammar('ROOT',['VV1']).
depgrammar('ROOT',['VA']).
depgrammar('VV2',['N','N']).
depgrammar('VP2',['N','N']).
depgrammar('VV1',['N']).
depgrammar('VP1',['N']).
depgrammar('N',['ART']).
depgrammar('N',['ADJ']).
depgrammar('N',['ART','ADJ']).
depgrammar('N',['P']).
```

```
depgrammar('N',['ADJ','P']).
depgrammar('N',['ART','P']).
depgrammar('N',['ART','ADJ','P']).
depgrammar('P',['N']).

congrammar('VA',['VP2']).
congrammar('VA',['VP1']).
congrammar('NN',['NN']).
```

Accordingly, an output for test sentence 5 ("The man has loved a woman.") looks as follows. For some unknown reason, every reading is found twice.

```
)
| ?- test5.
[the,ART,man,N,has,VA,loved,VP2,a,ART,woman,N]
!!NON_PROJECTIVE!! [loved,man]
+root([has,loved([man(the),woman(a)])]). PARSED: [5,1,6,2,4,3]. DEP.list:
     [[6,5],[2,1],[4,6],[4,2],[3,4],[0,3]]. NON_PROJ:  YES
!!NON_PROJECTIVE!! [man,a]
!!NON_PROJECTIVE!! [woman,the]
+root([has,loved([man(a),woman(the)])]). PARSED: [1,5,6,2,4,3]. DEP.list:
     [[6,1],[2,5],[4,6],[4,2],[3,4],[0,3]]. NON_PROJ:  YES
!!NON_PROJECTIVE!! [woman,the]
!!NON_PROJECTIVE!! [woman,the]
!!NON_PROJECTIVE!! [woman,the]
!!NON_PROJECTIVE!! [man,a]
!!NON_PROJECTIVE!! [man,a]
!!NON_PROJECTIVE!! [man,a]
!!NON_PROJECTIVE!! [woman,the]
!!NON_PROJECTIVE!! [woman,the]
!!NON_PROJECTIVE!! [woman,the]
!!NON_PROJECTIVE!! [woman,the]
!!NON_PROJECTIVE!! [loved,man]
+root([has,loved([woman(a),man(the)])]). PARSED: [1,5,2,6,4,3]. DEP.list:
     [[2,1],[6,5],[4,2],[4,6],[3,4],[0,3]]. NON_PROJ:  YES
!!NON_PROJECTIVE!! [woman,the]
!!NON_PROJECTIVE!! [man,a]
+root([has,loved([woman(the),man(a)])]). PARSED: [5,1,2,6,4,3]. DEP.list:
     [[2,5],[6,1],[4,2],[4,6],[3,4],[0,3]]. NON_PROJ:  YES
!!NON_PROJECTIVE!! [man,a]
!!NON_PROJECTIVE!! [man,a]
!!NON_PROJECTIVE!! [man,a]
!!NON_PROJECTIVE!! [woman,the]
!!NON_PROJECTIVE!! [woman,the]
!!NON_PROJECTIVE!! [woman,the]
!!NON_PROJECTIVE!! [man,a]
!!NON_PROJECTIVE!! [man,a]
!!NON_PROJECTIVE!! [man,a]
!!NON_PROJECTIVE!! [man,a]
!!NON_PROJECTIVE!! [loved,man]
+root([has,loved([man(the),woman(a)])]). PARSED: [5,1,6,2,4,3]. DEP.list:
     [[6,5],[2,1],[4,6],[4,2],[3,4],[0,3]]. NON_PROJ:  YES
!!NON_PROJECTIVE!! [man,a]
!!NON_PROJECTIVE!! [woman,the]
+root([has,loved([man(a),woman(the)])]). PARSED: [1,5,6,2,4,3]. DEP.list:
     [[6,1],[2,5],[4,6],[4,2],[3,4],[0,3]]. NON_PROJ:  YES
!!NON_PROJECTIVE!! [woman,the]
!!NON_PROJECTIVE!! [woman,the]
!!NON_PROJECTIVE!! [woman,the]
!!NON_PROJECTIVE!! [man,a]
!!NON_PROJECTIVE!! [man,a]
!!NON_PROJECTIVE!! [man,a]
!!NON_PROJECTIVE!! [woman,the]
!!NON_PROJECTIVE!! [woman,the]
!!NON_PROJECTIVE!! [woman,the]
!!NON_PROJECTIVE!! [woman,the]
!!NON_PROJECTIVE!! [loved,man]
+root([has,loved([woman(a),man(the)])]). PARSED: [1,5,2,6,4,3]. DEP.list:
     [[2,1],[6,5],[4,2],[4,6],[3,4],[0,3]]. NON_PROJ:  YES
!!NON_PROJECTIVE!! [woman,the]
!!NON_PROJECTIVE!! [man,a]
```

```
+root([has,loved([woman(the),man(a)])]). PARSED: [5,1,2,6,4,3]. DEP.list:
     [[2,5],[6,1],[4,2],[4,6],[3,4],[0,3]]. NON_PROJ:  YES
!!NON_PROJECTIVE!! [man,a]
!!NON_PROJECTIVE!! [man,a]
!!NON_PROJECTIVE!! [man,a]
!!NON_PROJECTIVE!! [woman,the]
!!NON_PROJECTIVE!! [woman,the]
!!NON_PROJECTIVE!! [woman,the]
!!NON_PROJECTIVE!! [man,a]
!!NON_PROJECTIVE!! [man,a]
!!NON_PROJECTIVE!! [man,a]
!!NON_PROJECTIVE!! [man,a]

no
```

3.3.3.4.2 Tesnière's Translation

The semantic structure built up in the above extension features a most simple form of argument composition, but the syntactic structure reported remains unchanged and hence non-projective. Only the mapping between syntax and semantics has been altered. If we follow Tesnière, then also the syntactic structure has to change. *AUX+main verb* should be syntactically combined into a something that behaves like a single word, by a translation (cf. 2.1.4).

This, too, is easy to implement. When parsing, the main verb is taken up into the search list `SList` which is used for checking completeness, but not into the list of dependencies `DList`, which is used to check for projectivity. The valencies from the main verb are used for the composed *AUX+main verb* word, as described in 2.1.4.2.

```
( 114) Translation and Argument Composition in head/13:
head(HWord,[HClass],Res,Sent,HPos,DPosL,DList,NewDList,FinDList,
     SList,NewSList,FinSList,PFLAG):- %Argument compostition1
     congrammar(HClass,[ConClass1]),
     dep(HWord,[ConWord1],[ConClass1],Res,Sent,HPos,ConPos,DList,ConDList
     1, SList,ConSList1,PFLAG),
     name(ConWord1,ConResASCII),
     name(HWord,HWordASCII),
     append(ConResASCII,[45],ConResASCIIDash),
     append(ConResASCIIDash,HWordASCII,AllASCII),
     name(ConWord,AllASCII),
     depgrammar(ConClass1,DClassL),
     dep(HWord,DWordL,DClassL,NewRes,Sent,HPos,DPosL,ConDList1,NewDList,
     ConSList1,NewSList,PFLAG),
     headarity(DWordL,DClassL,NewRes,Sent,DPosL,NewDPos,
     NewDList,NewNewDList,AriDlist,FinDList,
     NewSList,NewNewSList,AriSList,FinSList,PFLAG),
     Res =.. [ConWord,NewRes].
     %Is Dep. new local Head?
     %print('      ? '), print(Res), print(' - '), print(NewSList),
     %print(NewDList), print(PFLAG),nl.
```

Now the correct reading of test sentence 5 ("The man has loved a woman") is analyzed projectively again. The non-verbose output:

```
( 115) Parses of "The man has loved a woman":
| ?- test5.
[the,ART,man,N,has,VA,loved,VP2,a,ART,woman,N]
+root(loved-has([man(the),woman(a)])). PARSED: [5,1,6,2,4,3]. DEP.list:
     [[6,5],[2,1],[3,6],[3,2],[3,4],[0,3]]. NON_PROJ: _187
+root(loved-has([man(a),woman(the)])). PARSED: [1,5,6,2,4,3]. DEP.list:
     [[6,1],[2,5],[3,6],[3,2],[3,4],[0,3]]. NON_PROJ:  YES
+root(loved-has([woman(a),man(the)])). PARSED: [1,5,2,6,4,3]. DEP.list:
     [[2,1],[6,5],[3,2],[3,6],[3,4],[0,3]]. NON_PROJ: _187
```

```
+root(loved-has([woman(the),man(a)])). PARSED: [5,1,2,6,4,3]. DEP.list:
     [[2,5],[6,1],[3,2],[3,6],[3,4],[0,3]]. NON_PROJ:  YES
+root(loved-has([man(the),woman(a)])). PARSED: [5,1,6,2,4,3]. DEP.list:
     [[6,5],[2,1],[3,6],[3,2],[3,4],[0,3]]. NON_PROJ: _187
+root(loved-has([man(a),woman(the)])). PARSED: [1,5,6,2,4,3]. DEP.list:
     [[6,1],[2,5],[3,6],[3,2],[3,4],[0,3]]. NON_PROJ:  YES
+root(loved-has([woman(a),man(the)])). PARSED: [1,5,2,6,4,3]. DEP.list:
     [[2,1],[6,5],[3,2],[3,6],[3,4],[0,3]]. NON_PROJ: _187
+root(loved-has([woman(the),man(a)])). PARSED: [5,1,2,6,4,3]. DEP.list:
     [[2,5],[6,1],[3,2],[3,6],[3,4],[0,3]]. NON_PROJ:  YES

no
```

3.4 The Real Challenge: Constraining Non-Projectivity

The real challenge of non-projective parsing is to find out when to allow non-projective parses. It is e.g. obviously ridiculous to allow articles to be bound across the entire sentence. Here locality constraints in the form of a projectivity test are very reliable and traditional and inherently constituent-based measures.

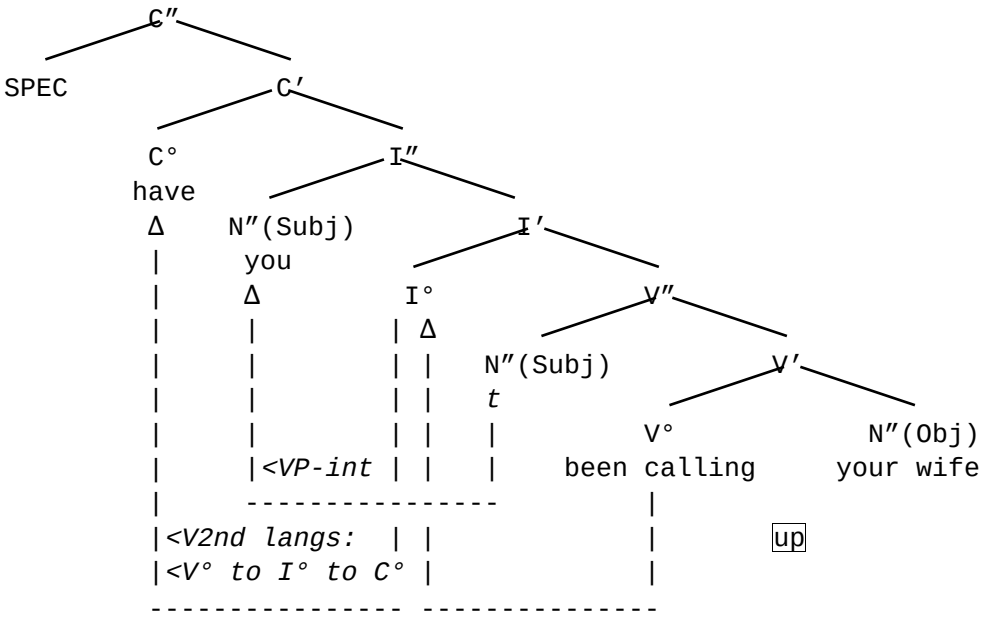
It is probably a safe procedure to allow only projective parses in a first step, and then permit transgressions in a well-defined way. Where and in which way to permit non-projectivity will be the really interesting question, which due to my limited time and resources I can not answer here. My suggested `fastparse/1` is only a very crude solution.

As far as I can see now, some of the reasons allowing non-projective parses are:

3.4.1 Verb Chains and Argument Composition

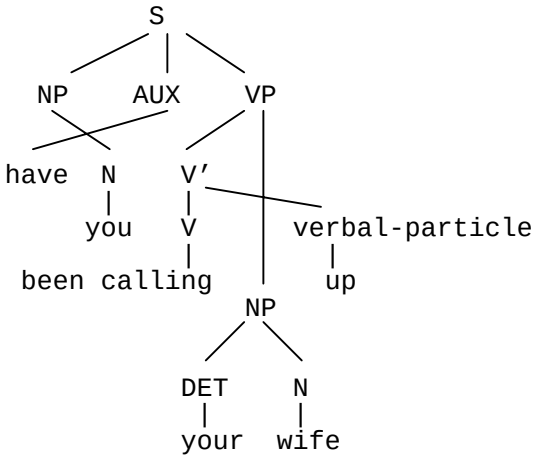
The distance between the inflected auxiliary, the main verb and verbal particles can be very big. Since these components of the verb have a very close semantic relation, however, they should ideally be very local in parsed structure. This is hardly possible in a projective analysis. GB uses a complex machinery of transformations to keep the structure projective. Let us look at this sentence (cf. 5.1.2.1)

(157) Have you been calling your wife up ?



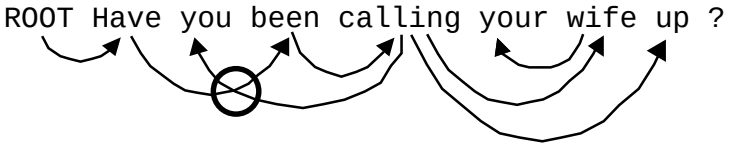
This fairly standard GB structure is unable to accommodate the verbal particle. A more intuitive and traditional constituent analysis is no longer projective. Note that the rule `VP -> V′ (Obj)NP` applies before `V′ -> V verbal-particle` in order to express the closer semantic proximity of the latter to the verb. The verbal component AUX is still separated very early, which is unsatisfactory.

(157 non-projective constituent analysis)



Dependency analysis manages to keep the semantic main predicate, i.e. the verb components together in a chain. The analysis is non-projective, but this does not transgress the principles of the theory.

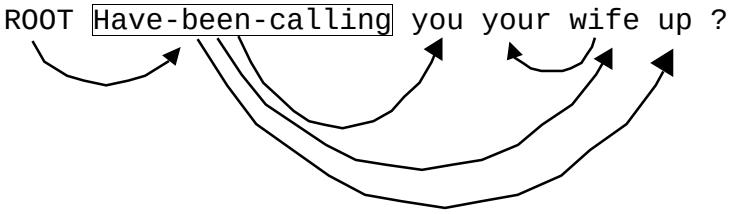
(157 non-projective dependency analysis)



Attaching the verbal particle is projective in dependency because, as we have seen in 2.4.5, dependency cannot express a distinction between X" and X'. Both *wife* and *up* are dependents of *calling*, one cannot specify that one comes first – or if we did, as we have seen suggested, wrong, projective, conversions would result.

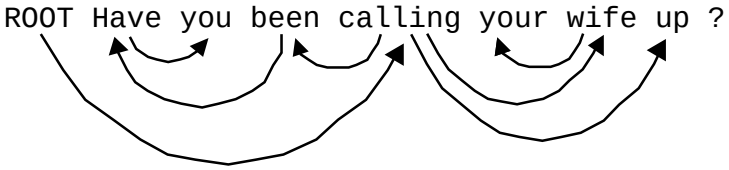
But this sentence (157) need not necessarily be non-projective. If we use argument composition or Tesnière's translation, we arrive at a projective analysis, however, as implemented in 3.3.3.4.

(157 projective dependency analysis with translation)



We have seen in 2.3.2.5.5 that it is debatable if AUX or VP should be the head. While AUX carries INFL and is thus the morphological core of the sentence, the main verb in VP contributes most to verb semantics and is the semantic head. If we aim at a semantically appropriate functional dependency structure, it is even more convincing to have the main verb as the head and AUX as its dependent:

(157 projective dependency analysis with main verb as head)



This is in fact the analysis suggested by the *Functional English Parser* [Järvinen & Tapanainen 1997, 1998]. The fact that on the one hand *ROOT* connects to the main verb and the main verb connects to the object, but on the other hand the subject is connected to AUX confirms that, internally, some sort of Tesnière's translation is used: "Internally in the parser, the verb chain is considered as one syntactic element" (Järvinen & Tapanainen 1997: 25).

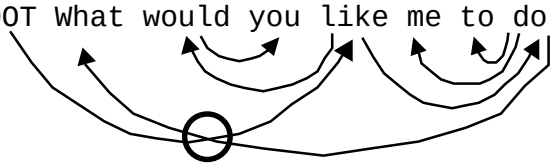
3.4.2 Morphological Dependency

In many highly inflectional languages extraposed adjectives etc. are a well-known poetical device. Covington's non-projective parser [1990] parses examples from Latin poetry. There is always morphological agreement between the head and the extraposed dependent. The extracted elements are always highly marked.

3.4.3 Long-distance Dependencies

Even if we use argument composition or translation as described in 3.4.1, even if extraposed adjectives like in Latin do not exist in English, at least in one case non-projectivity persists. Argument composition can hardly apply across subordinate clauses because of the intervening arguments (me in sentence (116) below). If in addition a WH-element is fronted in such a structure (long-distance WH-movement), it becomes non-projective.

```
( 116) What would you like me to do ?
ROOT What would you like me to do ?
```



3.5 Outlook

So far the parsers presented in this chapter had no practical application. I would like to extend and adapt them for the following tasks:

- **Dependency Parser on Functional Tagger**: The English constraint grammar tagger (ENGCG) yields functional rather than grammatical output. Basing a functional dependency parser on this instead of on Gertwol output will be a future task.
- **Dependency Parser on German NP-tool**: If an external NP recognizer delivers the NPs, a shallow dependency parser could be used to yield the functional relations between the main verb and the NPs.
- **Dependency Postprocessor for German Tagger**: One of our projects is training and evaluating taggers for German [Schneider and Volk 1998, Volk and Schneider 1998]. It will be interesting to see if a dependency-based post-processor can correct some of the remaining tagging errors.

3.6 Conclusions

I hope to have shown that the intuitive concept of dependency is not more difficult to implement than that of constituency. I also hope to have shown that any grammar needs either some sort of Chomskyan transformations, or it has to allow restricted non-projectivity, whether by the use of HPSG argument composition [Borsley 1996] or Tesnière's translation [Tesnière 1959] or explicitly use the concept of non-projectivity [Järvinen & Tapanainen 1997,1998].

4. Link Grammar vs. Dependency Grammars

The only descriptions of the relationship between Link Grammar and dependency grammar available come from the authors of Link Grammar themselves [Sleator & Temperley 1991, 1993]. [Sleator & Temperley 1993: 11] mention works by Gaifman, Mel'čuk, Fraser and Hudson as representatives of dependency grammar (see chapter 2.2). They also mention [Sleator & Temperley 1993: 12] a slight resemblance of Link Grammar to Categorial Grammar (see chapter 4.2.5)..

[Sleator & Temperley 1993] describe their Link Grammar as follows:

> A sequence of words is a *sentence* of the language defined by the grammar if there exists a way to draw *links* among the words so as to satisfy the following conditions:
>
> | | |
> |---|---|
> | Planarity: | The links do not cross (...) |
> | Connectivity: | The links suffice to connect all the words of the sequence together. |
> | Satisfaction: | The links satisfy the linking requirements of each word in the sequence. |
>
> The linking requirements of each word are contained in a *dictionary*.
>
> (Sleator & Temperley 1993: 1)

4.1 Differences between Link Grammar and Classical Dependency

Sleator & Temperley [1993.12] point out five areas of difference between Link Grammar and classical dependency:

4.1.1 Labeled Links

Unlike in dependency grammar, according to Sleator & Temperley [ibid.], links are labeled. E.g. a link called *S* connects the subject and the main verb, or a *D* link connects a determiner and a noun. This is not really a difference, as many variants of dependency grammar, perhaps the majority (e.g. Word Grammar (cf. 4.2.1), Dependency Parser for English (cf. 4.2.4), TRs in the Prague framework (cf. 2.2.5)) also use labeled links. Covington's [1994a] reinterpretation of dependency, which was discussed in chapter 2.4.5, also introduces labeled links, indicating if the dependent is *complement, adjunct* or *specifier* to the head. In this sense, labeled links should facilitate a mapping from link structures to X-bar structures or to a functional or theta-role structure.

4.1.2 Undirected Links

Link Grammar abandons the idea of dependency between *heads* and *dependents*. Since links are undirected, the two linked words appear on the same level.

This is a major difference, because dependency is what dependency grammar owes its name to. In 2.3.2 we have seen how central the notion of *head* is, and what a big part of dependency research deals with the question which of the participants in a connection should be head. One of the criteria considered there was that dependency is an existence relation, and that the head is compulsory:

> In der gröbsten Form des Dependenzmodells besteht Dependenz zwischen den Elementen A und B, "wenn das Auftreten von A syntagmatische Vorkommensbedingung für B ist" [Korhonen 1977: 40]. Mit anderen Worten handelt es sich bei Dependenz "um eine Vorkommensrelation, mit deren Hilfe aus dem Vorkommen eines Elementes auf das Vorkommen anderer Elemente geschlossen werden kann" [Korhonen 1977: 40, Baumgärtner 1977: 54, Engel 1971: 122] (Jung 1995: 21)

This suggests that from the existence of a dependent A we can conclude about the existence of the head B, as also the widely used elimination test (described e.g. in [Weber 1997:45] predicts.

By the use of curly brackets, Link Grammar allows for optional links, much as we know from many PSGs. While this has the obvious advantage of being able to discern between compulsory *arguments* [Radford 1988: 371-2] and optional *adjuncts* [Radford 1988: 175-179], Dependency relations are also implicitly expressed by them: Setting a link as *optional* on the *head side* - if the link is optional -, and as *compulsory* on the *dependent side* ensures correct emulation of the dependency even in a non-dependent system like Link Grammar. In other words a dependent, by virtue of its compulsory link, can only attach to its appropriate head, while a head may well exist without dependents.

Such an emulation of dependency by marking the link as optional on the head perfectly corresponds to [Fraser: 1996]'s description of dependency grammar:

> Dependency is an asymmetrical relation between two sentence elements, usually single words. One of the elements is defined as the 'governor', 'regent', or 'head' (...); the other element is defined as the 'dependent' or 'modifier'. A governor is distinguished from its dependent in a number of ways. These include: (a) the governor determines whether a dependent is optional or obligatory, and not vice versa; (b) the governor *subcategorizes* for its dependents, and not vice versa; (c) the *governor* determines which inflectional form of a dependent occurs, and not vice versa; (d) the governor identifies a semantic object which a dependent further specifies, and not vice versa. Other criteria for identifying governors have been proposed, but no single criterion provides a necessary and sufficient condition for dependency. (Fraser 1996: 71)

It has to be noted, however, that these 'implicit' dependency relations can only be derived in the case of optional dependents, when the head *optionally* offers a link. In the case of compulsory dependents, both the head and the dependent require a link – accordingly they appear on the same level in a hierarchical representation. Link Grammar is not only as monostratal as most dependency grammars, as lexical as Word Grammar, on top of that its structures are extremely flat, if somebody ever cared to represent Link Grammar analyses in Tesnière's *stemmas*.

Except for these 'implicit' and only partly derivable dependencies, the notion of *head*, a notion which is absolutely central in dependency, has been abandoned in Link Grammar. This is a crucial difference. If we want to convert link structures to X-bar or functional structures or topic-focus articulation structures, the latter two being important steps towards a semantic representation (cf. 6.2), we need head information, however. In 5.4.1, I therefore investigate if for each link type one of the participants can safely be interpreted as head, while the other would function as dependent. This would allow us to reintroduce dependencies and test parser outputs in this more widely used format and allow conversions.

On the other hand, we may argue that the notion *head* itself, although central to dependency and nowadays also to many PS style grammars like HPSG (hence the name: *head-driven* phrase-structure grammar) or X-bar Theory, is far from being uncontroversial (cf. 2.3.2.) If the direction of the dependency is so debatable, should we specify it? On the syntactical level, Link Grammar proves that it is not necessary. In unification-based grammars, it may be left unspecified too, because the direction of the variable binding is undefined. But it seems hardly possible to do semantics without a specification of direction, without applying one participant to the other as in *functional application* or in TFA (cf. 6.2).

4.1.3 Word Order and Projectivity

Instead of dependency, Link Grammar grammar rules, which – in accordance to the extremely lexical character of the formalism, in accordance to the projective character of most modern grammars, in accordance with the fact that dependency structures only contain lexical nodes – are in fact lexical entries, contain information on word order, i.e. whether the word to be linked to occurs before (expressed by '-') or after (expressed by '+') the word described in the lexical entry. The lexical entry for e.g. the article 'the'

```
(1.)   the:  D+;
```

would have to change to

```
(2a.)  the:  D+ or D-;
```

in a free word-order language. The representation internal to Link Grammar, the so-called *disjunctive form*, enumerates all ways in which a formula could be satisfied,

i.e. it solves all brackets, and lists all possible links on the left and on the right hand side of the word (the word itself is represented by the central blank) yielding for (2a.)

```
(2b.)  (D)   ()
        ()   (D)
```

Therefore, Sleator & Temperley [1993] conclude that "[t]he number of disjuncts [i.e. a line in *disjunctive form*] in the resulting link grammar is at most quadratic in the number of rules in the dependency grammar." (Sleator & Temperley 1993: 12)

Unlike many dependency grammars (cf. 2.4.6 and 2.4.7) the *planarity* constraint of Link Grammar makes it fully projective and context-free.

4.1.4 Root Word

There is no explicit notion of a root word in Link Grammar. The root word is the topmost head in dependency grammar. Because a hierarchical dependency ordering is only implicitly expressed in Link Grammar (the dependency emulation described in 4.1.2 above), and because the notion *head* seems to have been abandoned in Link Grammar, marking an element as the top head does probably not suit the philosophy of Link Grammar either.

Most sentence types, however, have a so-called *wall* link (*Wd* here) from an artificial word inserted at parse time before the beginning of the sentence to the subject.

```
( 117)
  +------------Xp------------+
  +---Wd---+--Spx--+--Pa-+   |
  |        |       |     |   |
///// elephants.n are grey.a .
```

This means that in Link Grammar the discussion whether the subject or the main verb constitutes the core still continues. When converting a link structure to dependency or a semantic representation, this poses unexpected problems, because the main verb first has to be found via the subject.

The *wall* link is absent in yes-no questions, and replaced by a *Q* link:

```
( 118)
  +------------Xp------------+
  |    +--------Pa-------+   |
  +-Qd-+--SIpx-+         |   |
  |    |       |         |   |
///// are elephants.n grey.a ?
```

This gives the impression that Link Grammar makes a uniform distinction between questions (*Q* link) and assertions (*W* link). This is not true, however, as most questions are also introduced by a wall link:

```
( 119)
  +--------------Xp--------------+
  |        +--------Pa-------+   |
  +-Wq-+-Q-+--SIpx-+         |   |
  |    |   |       |         |   |
///// why are elephants.n grey.a ?
```

And on the other hand, the *Q* link type is not only used to attach to the wall (“/////”), but as we can also see in (119) it attaches question words to verbs.

We can see two things:

- Link Grammar does not have a uniform root link, but two, one of them only being root link in some cases.
- Link Grammar is caught in the same configurational cage as constituency. Yes-no questions are the only kind of sentence type with an empty first position. English is generally a verb second (V2) language (cf. Vikner [1995], Schneider [1996]), but in yes-no questions this position either has to be declared empty, or yes-no questions are the only case of a verb first (V1) sentence type.

4.1.5 Cycles

Unlike in dependency grammar, a linkage in a Link Grammar may have cycles. Such analyses violate the very principles of dependency grammars. In Tesnière's stemma form (cf. 2.1.1), they can simply not be expressed. But the link that introduces circularity is the anaphoric binding link for the relative pronoun (*Bp*), i.e. a semantic link, which is either not directed even in Dependency (as Mel’čuk [1988] suggests), or whose direction is the same as that of the links it parallels. For anaphoric binding links, if we want to include them in the dependency structure, it also makes sense to relax the condition that every dependent can only have one head, as also Hudson [1990] suggests.

```
( 120)
  +----------------------Xp---------------------+
  |        +------------Spx-----------+         |
  |        +-----Bp----+----Pv---+    |         |
  +---Wd---+---R---+-RS+-N-+     |    +--Pa-+   |
  |        |       |   |   |     |    |     |   |
///// elephants.n who are not washed are grey.a .
```

4.1.6 Lexicalism

Link Grammar is indeed more lexical than most other dependency frameworks. While this allows a lexicon-driven projective character that takes care of the idiosyncratic valencies of each word, we remember that one of the forces of Tesnière's dependency is that we can take functional words up into the nucleus and form new multi-word nuclei by means of translations. Link Grammar's extreme lexicalism is probably rather a mistake than an advance.

4.1.7 Conversions between Link and Dependency Grammar

While [Sleator & Temperley 1993:12] are pessimistic about converting Link Grammar grammars to a dependency grammar, they are quite positive about the opposite conversion:

> It is easy to take a dependency grammar in Gaifman's notation and generate a link grammar that accepts the same language. In this correspondence, the linkage that results from parsing a sentence is the same as the corresponding dependency structure. This means that our algorithm for link parsing can be easily applied to Dependency Grammars. (Sleator & Temperley 1993: 12)

Their optimism culminates in the following statement: "We are not aware of an implementation of a dependency grammar for any natural language that is nearly as sophisticated as ours." (ibid.)

In order to convert Link Grammar to a dependency grammar, we would need a consistent system to find out which of the linked words is the head and which the dependent. I address this question in 5.4.1.

4.2 Differences between Link Grammar and Dependency-Related Grammars

4.2.1 Word Grammar

Word Grammar [Hudson 1984, 1990, 1996] is a grammar system which encompasses both a syntactic and a semantic theory. While nothing will be said about its semantic part here, its syntactic part is "explicitly and consistently based on syntactic dependencies" (Mel'čuk 1988: 7):

> Word Grammar (WG) is a theory of language structure. Its most distinguishing characteristics are the following:
>
> (a) that knowledge of language is assumed to be a particular case of more general types of knowledge, distinct only in being about language; and
>
> (b) that most parts of syntactic structure are analyzed in terms of dependency relations between single words, and constituency analysis is applied only to coordinate structures. (Hudson 1996: 368)

While (a) refers to the semantic part of Word Grammar, (b) addresses its syntactic part. In at least three ways, Word Grammar's syntax seems to be even closer to Link Grammar than other variants of dependency grammar:

4.2.1.1 Lexicalism

"We are not aware of any notation for dependency systems that is lexical ... as link grammars" (Sleator & Temperley 1993: 12). It is surprising to read this from the developers of Link Grammar, who mention [Hudson 1984] in the same text. He has paved the road towards lexicalism for (at least his version of) dependency grammar a long time ago. But Link Grammar is at least as lexical as Word Grammar, so much lexical that the disadvantages this brings are probably bigger than the projective lexicon-driven character lexicalism brings on the positive side.

About Word Grammar, Hudson [1980] states:

> I have also argued (...) that 'dependency' is just another name for the relation between a 'frame' and its 'slots' – in other words, for the relation of 'strict subcategorisation', which in transformational grammar is relegated to the lexicon. If the claims of this paper are right, then, they constitute strong further support for the 'pan-lexical' model in which the grammar is virtually identified with the lexicon. ... Like the traditional lexicon, the pan-lexicon refers only to words. (Hudson 1980: 196)

From [Hudson 1990] onwards, the strongly lexical character of Word Grammar is being stressed. "WG is **lexicalist** because the word is central – hence the name of the theory." (Hudson 1990: 10)

4.2.1.2 Labeled Arcs

Like many but not all dependency grammars, Word Grammar uses labeled dependents or arcs, so-called "enriched dependency structures" (Hudson 1996: 369):.

> Dependency theory has always allowed one word to have more than one dependent, in contrast with its single head. Different dependents of a single word or word-type often have different characteristics, all of which need to be defined in rules, so it is necessary to distinguish one dependent from another – hence the traditional set of 'grammatical relation' categories like 'subject', 'object', 'complement' and so on. (Hudson 1990: 120)

The Word Grammar label categories form a much smaller set than the one used by Link Grammar. Word Grammar categories are much more functionally oriented than those in Link Grammar.

4.2.1.3 Multiple Heads

Word Grammar deviates in a number of exceptions from the dependency rule that words are allowed to have only one head. This allows Word Grammar to deal with discontinuous sentences, e.g. in raising constructions [Hudson 1990: 111 ff.]. Because heads are at best inherently to be found in Link Grammar, it is difficult to assess if

similar extensions to dependency grammar could be realized, or have already been so, within Link Grammar.

4.2.2 Valency Grammar

Chapters 2.2.3 and 2.3.1 have already discussed how central valency is to dependency grammar. It is difficult to judge whether Valency Grammar [Allerton 1996: 359-368] merits its own subchapter among the list of theories closely related to Link Grammar, or whether valency grammar is just an instance of a link grammar or a dependency grammar. If we choose the former option, it is possible to build up an argument that there may be relevant differences between valency and dependency. We may also choose the latter and agree with Schubert [1988:56], who says in his discussion on dependency grammar: "Ich glaube auch nicht, dass es vielversprechend wäre, eine Art 'Valenzgrammatik' um einen unabhängig definierten Begriff der Valenz herum zu konstruieren."

"The invention of the notion of *valency* is often credited to Lucien Tesnière" (Allerton 1996: 359), but it is already inherently mentioned in Karl Bühler's *Sprachtheorie* [1934]: " '... words of a particular word-class open up around them one or several "empty places", which have to be filled by words of certain other word-classes [1934: 173]." (ibid.). Valency is typically just another term for subcategorisation – hence e.g. the German translation "Verbvalenz" for "verb subcategorisation". Valency theory started also with research on verbs, in Tesnière's tradition, and then tried to apply similar subcategorisation mechanisms to other word classes.

> Valency is ... seen as the capacity a verb (or noun, etc.) has for combining with particular patterns of other sentence constituents, in a similar way to that in which the valency of a chemical element is its capacity for combining with a fixed number of atoms of another element. (Allerton 1996: 359)

Many adjectives select prepositions, often in an idiosyncratic way, which therefore has to be noted in the lexicon. It is fair to say that the adjective subcategorizes for the appropriate preposition. "As in the case of prepositional verbs, the adjectives each select a particular preposition, in an often arbitrary way" (Allerton 1996: 366). In constituency terms, the adjective subcategorizes for a prepositional phrase (PP), in dependency terms it subcategorizes for a prepositions which in turn requires, 'subcategorizes' for a noun. Similarly, many nouns select specific prepositions, which are an object of valency research.

But valency theory is generally more conservative than dependency. Valency sees itself only as a part of a grammar theory. While "[v]erbs, adjectives, and nouns are the three major lexical word classes for which the concept of valency is clearly appropriate and for which detailed language studies have been made and preliminary

dictionaries compiled" (Allerton 1996: 367), he apprehends limits in other areas, which in his view rule out a grammar based on valency only:

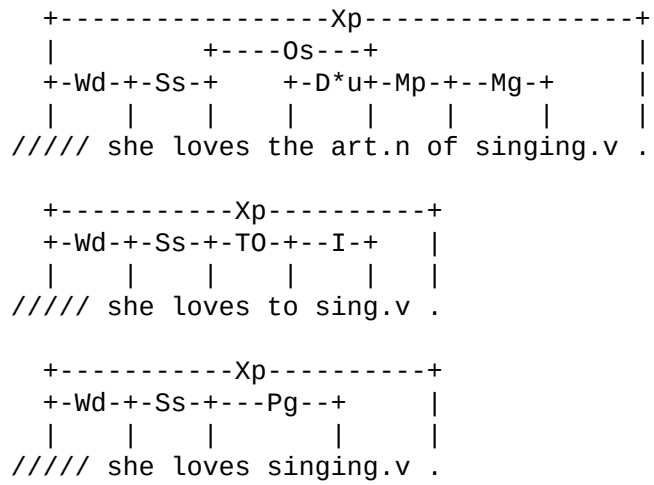
> Although the application of valency can be thus extended, it is not really intended to account for the whole of a syntactic system ... . Valency is a matter of the subcategorization of lexical categories, and therefore in a strict sense would exclude:
>
> (a) coordinate structures, which are treated separately both by Tesnière (under the heading of French *jonction*) and in Hudson's *Word Grammar* (where they are basic and not derived from word-word relations);
>
> (b) grammatical 'specifiers', such as auxiliaries and (nonpossessive) determiners, which although treated as dependents (or even governors) by many dependency grammarians, were regarded by Tesnière as involving a different relationship to main verbs and nouns respectively – a view still valid today;
>
> (c) 'subordinators' in Tesnière's interpretation, which embraced prepositions and subordinating conjunctions, these being for him converters (French *translatifs*), or noun phrases or clauses to adverbial or adnominal function ... – again a theoretically tenable position.
>
> (Allerton 1996: 368)

The quoted enumeration clearly shows the problems discussed in dependency grammar in chapter 2, namely

(a) the need for a constituency concept within dependency to deal with coordination. Link Grammar deals with coordination in the same way as with ambiguity. In a construction *A and B* it first reports a linkage of *A*, and then a separate linkage for *B*, as if they were two ambiguous readings. Since complex sentences often contain complex coordination, we get an explosion of this artificial kind of ambiguity.

(b) Tesnière takes functional heads and verbal parts *up* into the semantic nucleus (like e.g. the phrasal particle *up* in this sentence!). Link Grammar is absolutely word-based and projective and therefore unable to perform this. It means that Link Grammar links are considerably less functional and closer to the syntactic surface.

(c) translations facilitate functionalism. Without a concept similar to them, and because Link Grammar is completely word-based, it cannot make functional generalizations. In the following sentences, the generalization that the element dependent on the main verb *love* cannot be made:



One interesting question to ask is whether the very closely related concepts valency and dependency always have to mean the same. In other words: Is dependency the most appropriate way to express valency? "Valency can ... evidently be included in a dependency grammar; for Engel (1977: 116), in fact, valency is simply a matter of dependency on subclasses."

Modern PS grammars like HPSG or PATR express agreement and valency by means of typed features. Their use has been suggested by [Zwicky 1986], he calls them 'tagged features'. In his research on lexical integrity, [Anderson 1992] bases his *revised lexical integrity principle* on them (cf. [Schneider 1997: 7]). [Covington 1994b]'s extension to PROLOG, *Graph Unification Logic Programming* (GULP) implements typed features.

The direction of a unification, unlike in dependency, but like in Link Grammar, is unspecified. While information can only monotonously increase, it is unknown which of the participants in the unification constrains (in fact both can constrain). E.g. in the sentence

(3) She walks

the lexical entries of both 'she' and 'walks' will have a typed feature such as `(PERS:3 .. NUM:sg)`. The conception of the verb selecting a 3rd person singular noun is as correct as the conception of the noun selecting a 3rd person singular verb. Thinking of agreement in terms of dependency can be misleading.

In the case of subcategorisation, the case for a possible absence for a clear dependency is much harder to make. After all, only a verb subcategorizes say for an object, while the fact that objects need verbs is contained in a PS rule. Apparently a clear case of dependency: "(b) the governor *subcategorizes* for its dependents, and not vice versa" (Fraser 1996: 71).

However, in systems which are as lexical as Link Grammar or Word Grammar, no PS rules or equivalent general dependency or link rules exist. The lexical entry of the object in question requires, 'subcategorizes for' a verb. In this view, again, it is just as correct to conceive of a verb selecting an object it subcategorizes for as an object selecting a verb it subcategorizes for. And accordingly, in this view valency can be thought of as non-dependent, suitable for expression by a link system as easily as by a dependency system. This could be another possible answer to chapter 2.2.2.2, which addressed one of the major differences between dependency and link: while Link Grammar does not abide to this central concept of dependency grammar, it perfectly abides to the corresponding principle of the extremely closely related Valency Grammar.

Dependency Unification Grammar DUG [Hellwig 1986] has taken up the idea of building up a dependency grammar which uses typed features (cf. 3.2). [Steimann & Brzoska 1995] is a toy implementation of this idea. For simplicity's sake, instead of typed features they simply use argument positions in Prolog, i.e. the devices usually containing agreement or subcategorisation information. "The terms n(.,.) provide space for feature structures commonly employed to capture syntactic and semantic properties of words [Shieber 1986, Knight 1989]" (Steimann 1995: 96). Again, as always, the direction of the unification is in principle free, which means that the dependency can be seen as bidirectional.

4.2.3 X-bar Grammar

Unexpectedly at first sight, X-bar grammar is also related to Link Grammar due to its alleged equivalence to dependency grammar, which is claimed in [Hudson 1990: 111 ff.] and [Covington 1994a]. See chapter 2.4.5 for a more detailed discussion and 2.4.6 for the rebuttal of their claim.

Because Link Grammar abandons heads and dependencies, links are bi-directional, with participants on the same hierarchical level. Link Grammar is still context-free.

4.2.4 Functional Dependency Parser of English from Helsinki

Although there might be more than one *functional dependency parser of English*, this is the official name of the dependency parser developed at Helsinki University [Järvinen & Tapanainen 1997, 1998]. The system can be tested online at

http://www.ling.helsinki.fi/~tapanain/dg/

or

http://www.conexor.fi/analysers.html

At the second address, an attractive and extremely entertaining graphical output is also available in the form of quite an ingenious Java applet. My subjective impression is that the coverage is broad, but slightly below Link Grammar. But the dependency analyses are indeed much more functional than those from Link Grammar – as the name correctly suggests.

The functional terms *subject*, *object*, *attribute, modifier*, and *complement* are used very consistently and, as far as I could see, in correspondence with Quirk et al. [1985], which the grammar model aims to follow: "The inventory of structures which are dealt with in our description are based largely on the Quirk et al.'s (1985) *Comprehensive Grammar of the English Language*" (Järvinen & Tapanainen 1997: 7). I did not have the time, unfortunately, to test this claim in detail, but it will be rewarding to do so.

Let us consider some functionally related sentences, and the reported dependency structures (in textual representation):

```
( 121) She loves the art of singing.

OUTPUT:
0
1         She       she       subj:>2   @SUBJ PRON
2         loves     love      main:>0   @+FMAINV V
3         the       the       det:>4    @DN> DET
4         art       art       obj:>2    @OBJ N
5         of        of        mod:>4    @<NOM-OF PREP
6         singing   sing      pcomp:>5  @<P-FMAINV ING
          $.
```

As expected, *she* is subject and *art* object, both dependents of the main verb at position 2.

```
( 122) She loves to sing.

OUTPUT:
0
1         She       she       subj:>2   @SUBJ PRON
2         loves     love      main:>0   @+FMAINV V
3         to        to        pm:>4     @INFMARK> INFMARK>
4         sing      sing      obj:>2    @-FMAINV V
          $.
```

In functional terms, the non-finite verb sing is also an object, as Tesnière's translation allows (cf. 2.1.4) . Note that to is not a functional head, but only a marker.

```
( 123) She loves her boyfriend singing.

OUTPUT:
0
1         She       she       subj:>2   @SUBJ PRON
2         loves     love      main:>0   @+FMAINV V
3         her       she       attr:>4   @A> PRON
4         boyfriend boyfriend subj:>5   @OBJ N
5         singing   sing      obj:>2    @-FMAINV ING
          $.
```

Again, *singing* - or rather *singing* plus all its dependents - is the object of the main verb. While Tesnière would use a translation to convert the non-finite clause *her boyfriend singing* into a nominal nucleus in object function, this grammar prefers to mark the object function on the head of the non-finite clause. *Boyfriend* is a subordinate subject of the object, which keeps its valency information despite the translation to object.

```
( 124) She hates that he cannot sing.

OUTPUT:
0
1         She       she       subj:>2   @SUBJ PRON
2         hates     hate      main:>0   @+FMAINV V
3         that      that      pm:>7     @CS CS
4         he        he        subj:>5   @SUBJ PRON
5         can+      can       v-ch:>7   @+FAUXV V
6         not       not       neg:>5    @ADVL NEG-PART
7         sing      sing      obj:>2    @-FMAINV V
          $.
```

Even if a finite subordinate clause is introduced, the functional relations remain unchanged despite the different syntactic means – *sing* plus its dependents stays object. Like *to* in (122) *that* is only a marker of the subordinate verb instead of a functional head.

```
( 125) She hates it that he cannot sing.

OUTPUT:
0
1         She       she       subj:>2   @SUBJ PRON
2         hates     hate      main:>0   @+FMAINV V
3         it        it        obj:>2    @OBJ PRON
4         that      that      obj:>8    @OBJ PRON
5         he        he        subj:>6   @SUBJ PRON
6         can+      can       v-ch:>8   @+FAUXV V
7         not       not       neg:>6    @ADVL NEG-PART
8         sing      sing      mod:>3    @-FMAINV V
          $.
```

Sing and its dependents is correctly analyzed as a modifier of the *it* pronoun. Otherwise the functional analysis is the same as for (124).

Some of the functionally closely related structures that receive very different linkages in Link Grammar (cf. 5.1.1.2) , e.g. *dative shift*, receive closely related analyses by the functional dependency parser - although others, namely *passive (127)*, *cleft* and *pseudo-cleft*, do not, however:

```
( 126a) Peter gives the book to Mary.

OUTPUT:
0
1         Peter     Peter     subj:>2   @SUBJ N
2         gives     give      main:>0   @+FMAINV V
3         the       the       det:>4    @DN> DET
4         book      book      obj:>2    @OBJ N
5         to        to        dat:>2    @ADVL PREP
6         Mary      Mary      pcomp:>5  @<P N
          $.

( 126b) Peter gives Mary the book.

OUTPUT:
0
1         Peter     Peter     subj:>2   @SUBJ N
2         gives     give      main:>0   @+FMAINV V
3         Mary      Mary      dat:>2    @I-OBJ N
4         the       the       det:>5    @DN> DET
5         book      book      obj:>2    @OBJ N
          $.

( 127) The book is given to Mary by Peter.

OUTPUT:
0
1         The       the       det:>2    @DN> DET
2         book      book      subj:>3   @SUBJ N
3         is        be        v-ch:>4   @+FAUXV V
4         given     give      main:>0   @-FMAINV EN
5         to        to        dat:>4    @ADVL PREP
6         Mary      Mary      pcomp:>5  @<P N
7         by        by        agt:>4    @ADVL PREP
8         Peter     Peter     pcomp:>7  @<P N
          $.
```

The deep subject is not recognized, but the thematic *agent* role assigned to Peter easily allows to reconstruct it.

Although the coverage seems to be slightly lower than in Link Grammar, in terms of linguistic adequacy the Functional Dependency Parser is certainly superior.

4.2.5 Other Related Grammar Systems

Sleator & Temperley [1993] themselves point out that "[a]nother grammatical system, known as *categorial grammar* ... bears some resemblance to link grammars." (Sleator & Temperley 1993: 12). Link Grammar and Categorial Grammar seem to be further apart than Link Grammar and dependency grammar. Valency plays a central role in

categorial grammar, so much so that – loosely speaking – word classes are not named by what they are, but by what they lack, and by what they will be once their valency is satisfied. *V/N* e.g. means 'something which misses a noun to become a complete verb, a verb phrase'. However, I have not investigated Categorial Grammar.

Mel'čuk [1988: 7] mentions *Lexical-Functional Grammar* [Bresnan 1982] (see 2.3.8.2), *Case Grammar* [Fillmore 1968], *Relational Grammar* [Perlmutter 1983], and Hudson's Word Grammar (see chapter 4.2.1) as important stepping stones towards dependency.

4.3 Link Grammar's Post-Processing

It is important to stress that Link Grammar uses extensive post-processing in order to filter its results. As I have learnt just before completing this paper, post-processing is used to identify clauses. It seems that clause-boundary detection may indeed be a major problem for dependency-related theories. "In fact, the domains of post-processing essentially identify clauses" (Sleator & Temperley, 1998b).

5. A Closer Look at Link Grammar Link Types

First time users of Link Grammar are always surprised about the plethora of different labels used. Only some of them are intuitive, such as *S* for subject or *O* for object. But Link Grammar uses many more labels than e.g. Word Grammar (cf. 4.2.1). It often seems questionable whether all those labels still bear any functional meaning, and if they will turn out to be useful for the semantic analysis. This chapter does not aim to be an introduction to Link Grammar, for which [Sleator & Temperley 1991, 1993, 1998a, 1998b] are recommended.

During the completion of this chapter, the new version 3.0 of Link Grammar just became available. The main changes involve robustness, but also the grammar has been extended. I have therefore rewritten parts of this chapter. Chapters 5.1, 5.2, and 5.3 are now consistently based on version 3.0, while chapter 5.4 is still based on the older version 2.1.

5.1 Comparing Link to A Standard English Grammar

It is rewarding to compare the grammar provided in the Link Grammar distribution to a standard grammar of English, e.g. Quirk et al. [1985]. This task would easily merit a finals paper of its own, but it is possible to make some observations and give an overview in short.

When I speak about Link Grammar here, I am referring to the Link Grammar grammar provided for English, which is delivered together with the system. It would be possible to develop a grammar of one's own, with partly different characteristics. Some of the limitations, however, e.g. context-freeness, a monostratal approach or the problems with coordination are limitations of the system.

[Quirk et al. 1985] is built up in three cycles, each dealing with topics in more detail than the former [ibid. 2:37]. The first cycle (chapter 2 of he book) offers an overview. I will follow this overview in 5.1.1.

The second cycle (chapters 3 to 11), which will be taken up in 5.1.2., investigates the basic constituents of a simple sentence. They are the verb phrase - or in dependency terms the verb and its dependents, the noun phrase, adjectives and adverbs, adverbials, and finally prepositions and prepositional phrases.

The third cycle (chapters 12 to 19) deals with more complex sentence structure and the functional relations between the clauses. I will briefly summarize the problems encountered in 5.1.3.

Many of the topics discussed in [Quirk et al 1985] are of semantic rather than syntactic character, they can not be addressed by a syntactic parser. I will therefore only discuss some selected topics which are especially challenging in syntactic parsing. Neither was it possible, in the short space provided, to test idiomatic usage or lexical idiosyncrasies.

5.1.1 An Overview

Following the tradition, Quirk et al. introduce constituents as elements that can be identified through relationships of choice or substitution [ibid. 2.3-11:38-48] they introduce the important distinction between form and function.

> In order to state more complicated facts of constituency ..., it is important to distinguish two ways of classifying constituents. We may classify a unit either on the base of its FORM (e.g. its internal structure, as a noun phrase, or as a verb phrase), or on the basis of its FUNCTION (e.g. as a subject or an object of a clause). (Quirk et al. 1985 2.12:48)

5.1.1.1 Functions of Clauses and Phrases

They stress that functional classification is important on the clause and phrase level [ibid. 2.13:49], where they distinguish the following elements: Subject(S), Object(O), Complement(C) and Adverbial (A).

Some of these functional elements, especially adverbials are often optional. But after eliminating optional elements we arrive at the following seven core clause types [ibid. 2.16:53]

	S(ubject)	**V(erb)**	**O(bject)**	**C(omplement)**	**A(dverbial)**
Type *SV*	Someone	was laughing			
Type *SVO*	My mother	enjoys	parties		
Type *SVC*	The country	became		totally independent	
Type *SVA*	I	have been			in the garden
Type *SVOO*	Mary	gave	the visitor		
			a glass of milk		
Type *SVOC*	Most people	consider	these books	rather expensive	
Type *SVOA*	You	must put	all the toys		upstairs

Table 4: Clause types [Quirk et al. 1985 2.16:53]

Let us take a look at the Link Grammar analyses of these example sentences:

- **Type SV**:

```
( 128) Someone was laughing.

  Linkage 1, cost vector = (UNUSED=0 DIS=0 AND=0 LEN=3)

  +-------------Xp-------------+
  +--Wd--+--Ss--+---Pg---+     |
  |      |      |        |     |
///// someone was.v laughing.v .

  Linkage 2, cost vector = (UNUSED=0 DIS=4 AND=0 LEN=3)

  +-------------Xp-------------+
  +--Wd--+--Ss--+---Ost--+     |
  |      |      |        |     |
///// someone was.v laughing.v .
```

As in every Link structure, main link types are in capitals, subtypes or features (so-called subscripts) in lower case letters. *W* links the beginning of the sentence to the subject, *X* connects the beginning to the end. The *S* link correctly connects the subject and the inflected verb. So far everything is as expected.

Comment:

The first reading is correct. One may expect *P* to stand for *Participle*. Looking up the *P* link in [Sleator & Temperley 1998a] reveals, however: "P is used to link forms of the verb "be" to various words that can be its complements: prepositions, adjectives, and passive and progressive participles". *P* is a very broad and unspecific category.

It is surprising, however, that Link Grammar brings up two parses (so-called linkages). In the second one, which is correctly dispreferred to the first one, *laughing* acts as a special type of object (*O*). If we look up the *Ost* link [Sleator & Temperley 1998a] we find that it should only occur in connection to expletive *there* as subject, in sentences like:

```
( 129) There is someone laughing.

  +----------------Xp---------------+
  +--Wd-+-SFst+-Ost-+----Mg---+     |
  |     |     |     |         |     |
///// there is.v someone laughing.v .
```

We have seen in the GB VP-internal subject hypothesis (2.3.9.1) that it is justified to suggest the same structure for sentences with and without expletive there ("Someone is laughing" and "There is someone laughing"). The *Ost* link in (129),

however, links the Object to the main verb, while in the second reading of (128) it links the auxiliary to the gerund.

Something has gone completely wrong in the second reading of (128). First the *Ost* link should not occur in this environment, secondly it makes the wrong statement that laughing can be some kind of object.

If we take a closer look at (129), we notice that someone has to be subject, at least on a deep or functional level, not object. We get a first glimpse of one of the problems of Link Grammar: it is extremely surface-oriented.

- **Type SVO:**

```
 ( 130) My mother enjoys parties.

  +----------------Xp----------------+
  +----Wd----+                       |
  |    +--Ds-+---Ss---+---Op---+     |
  |    |     |        |        |     |
///// my mother.n enjoys.v parties.n .
```

Comment:

One linkage, which is correct. The subscripts on *O* and *S* express singular (*s*) and plural (*p*).[10]

- **Type SVC:**

```
( 131) The country became totally independent.

  Linkage 1, cost vector = (UNUSED=0 DIS=0 AND=0 LEN=7)

  +-----------------------Xp-----------------------+
  +-----Wd----+         +--------Pa--------+       |
  |    +--Ds--+----Ss---+--MVa--+          |       |
  |    |      |         |       |          |       |
///// the country.n became.v totally independent.a .

  Linkage 2, cost vector = (UNUSED=0 DIS=0 AND=0 LEN=7)

  +-----------------------Xp-----------------------+
  +-----Wd----+         +--------Pa--------+       |
  |    +--Ds--+----Ss---+       +----EA----+       |
  |    |      |         |       |          |       |
///// the country.n became.v totally independent.a .
```

[10] Subscripts have very different meanings according to different link types. The *p* subscript of *Xp*, e.g., does not express plurality.

Comment:

Both readings are reported as equally preferable by the Link Parser. The first linkage, in which totally modifies the copula, appears hardly acceptable, but it cannot be completely ruled out.

As we have seen above, the *P* link is a very broad category, which heavily relies on subscripts to become more specific. But *P* generally links to (obligatory) complements. Let us assume that *P* stands for complement function. *Pa* links the verb to predicative adjectives:

> Pa connects certain verbs to predicative adjectives:
>
> ```
>       +-S--+-Pa-+
>       |    |    |
>  The dog was black
> ```
>
> Only certain verbs carry Pa+ connectors ("be", "seem", "look", "taste"). (Sleator & Temperley 1998a)

Let us consider this sentence [QAL 2.17:55]:

```
( 132) The country became a separate nation.

  +------------------------Xp-----------------------+
  |                     +----------Os---------+     |
  +-----Wd----+         |    +-------Ds-------+     |
  |    +--Ds--+----Ss---+    |      +----A----+     |
  |    |      |         |    |      |         |     |
///// the country.n became.v a separate.a nation.n .
```

Nation is treated as an object instead of a complement. In terms of Case this is fundamentally wrong, at least for most Romance or Germanic languages which mark Case. Link Grammar does not make a proper distinction between objects and complements.

- **Type SVA:**

```
( 133) I have been in the garden.

  Linkage 1, cost vector = (UNUSED=0 DIS=0 AND=0 LEN=7)

  +------------------Xp-----------------+
  |                      +----Js---+    |
  +-Wd-+-Sp*i+--PPf-+-Pp-+  +--Ds--+    |
  |    |     |      |    |  |      |    |
///// I.p have.v been.v in the garden.n .

  Linkage 2, cost vector = (UNUSED=0 DIS=2 AND=0 LEN=7)

  +------------------Xp-----------------+
  |                      +----Js---+    |
  +-Wd-+-Sp*i+--PPf-+-MVp+  +--Ds--+    |
  |    |     |      |    |  |      |    |
///// I.p have.v been.v in the garden.n .
```

Comment:

Absolutely correct. The first linkage is marked as preferable. In it, the prepositional phrase is an obligatory complement, while in the second, it is an optional verb modifier.

- **Type SVOO**:

```
( 134) Mary gave the visitor a glass of milk.

  +------------------------Xp----------------------+
  |           +----------Osn---------+              |
  |           +-----Os----+          |              |
  +--Wd-+--Ss-+    +--Ds--+     +-Dsu+--Mp-+-Jp-+   |
  |     |     |    |      |     |    |     |    |   |
///// Mary gave.v the visitor.n a glass.n of milk.n .
```

Comment:

Absolutely correct. Note, however, that the functionally equivalent sentence

```
 ( 135) Mary gave a glass of milk to the visitor
```

receives a different analysis:

```
  Linkage 1, cost vector = (UNUSED=0 DIS=0 AND=0 LEN=15)

  +-------------------------Xp-------------------------+
  |           +-----------MVp----------+               |
  |           +---Os---+               +----Js---+     |
  +--Wd-+--Ss-+   +-Dsu+--Mp-+-Jp-+    |  +--Ds--+     |
  |     |     |   |    |     |    |    |  |      |     |
///// Mary gave.v a glass.n of milk.n to the visitor.n .

  Linkage 2, cost vector = (UNUSED=0 DIS=1 AND=0 LEN=11)

  +-------------------------Xp-------------------------+
  |           +---Os---+               +----Js---+     |
  +--Wd-+--Ss-+   +-Dsu+--Mp-+-Jp-+-Mp-+  +--Ds--+     |
  |     |     |   |    |     |    |    |  |      |     |
///// Mary gave.v a glass.n of milk.n to the visitor.n .

  Linkage 3, cost vector = (UNUSED=0 DIS=1 AND=0 LEN=13)

  +-------------------------Xp-------------------------+
  |           +---Os---+-------Mp------+----Js---+     |
  +--Wd-+--Ss-+   +-Dsu+--Mp-+-Jp-+    |  +--Ds--+     |
  |     |     |   |    |     |    |    |  |      |     |
///// Mary gave.v a glass.n of milk.n to the visitor.n .
```

Linkage 2 expresses something like "Mary gave a glass of visitor's milk" and linkage 3 "Mary gave a visitor's glass of milk". Linkage 1 corresponds more to what we were looking for. But the recipient prepositional phrase is linked as an optional

prepositional verb modifier instead of an obligatory indirect object. This is a questionable analysis, because it suggests that (134) and (135) have a different argument structure and the main verb a different arity. Link Grammar is too much surface-oriented to be truly functional here.

- **Type SVOC**:

```
( 136) Most people consider these books rather expensive.

  +------------------------------Xp-----------------------------+
  |                      +--------------Paf--------------+      |
  +-----Wd-----+         +------Op------+                |      |
  |     +--Dmc-+----Sp---+       +--Dmc-+       +---EA---+      |
  |     |      |         |       |      |       |        |      |
///// most people.p consider.v these books.n rather expensive.a .
```

Comment:

The linkage seems to be correct. The complement is treated as a special kind of predicative adjective. Checking *P* with subscript *f* in Sleator & Temperley [1998a] reveals, however, that this refers to 'filler' expletives:

```
        +SFsi+--Paf-+-THi+--Ce+-S(e)-+I(e)+
        |    |      |    |    |      |    |
        It  is   likely that Jane  will  go
```

(Sleator & Temelrey 1998a)

While the noun on which the predication is made is indeed in unusual position, marked as object instead of the more usual subject, it is questionable if expletive structures and *SVOC* sentences should be given the same link type. Especially the semantic link between the object and the complement is missing.

Let us consider this sentence [Quirk et al. 2.17:55]:

```
 ( 137) Most people consider Picasso a genius.

  +-----------------------Xp----------------------+
  +-----Wd-----+         +--------Osn--------+    |
  |     +--Dmc-+----Sp---+---Os---+    +-Dsu-+    |
  |     |      |         |        |    |     |    |
///// most people.p consider.v Picasso a genius.n .
```

As in (132), Link Grammar does not make a proper distinction between complements and objects.

- **Type SVOA**:

```
( 138) You must put all the toys upstairs.

  +----------------------Xp----------------------+
  |                +-----------Pp----------+     |
  |                |    +----Jp---+        |     |
  +-Wd-+--Sp-+--I--+--O-+ALx+-Dmc-+        |     |
  |    |     |     |    |   |     |        |     |
///// you must.v put.v all the toys.n upstairs.e .
```

Comment:

Correct, although the linking of *all* is unusual (see 5.1.1.2). *Upstairs* is compulsory, but its functional status as adverbial is not recognized. Link Grammar thus fails to make a distinction between objects, complements and adverbials.

But it clearly distinguished obligatory dependents (complements, objects, some adverbials) from all optional ones (modifiers). In (138), the adverbial is part of the verb valency, in (139) it is optional:

```
( 139) You must like all the toys upstairs.

  Linkage 1, cost vector = (UNUSED=0 DIS=1 AND=0 LEN=8)

  +----------------------Xp----------------------+
  |                 +----------MVp----------+     |
  |                 |    +----Jp---+        |     |
  +-Wd-+--Sp-+---I--+--O-+ALx+-Dmc-+        |     |
  |    |     |      |    |   |     |        |     |
///// you must.v like.v all the toys.n upstairs.e .

  Linkage 2, cost vector = (UNUSED=0 DIS=1 AND=0 LEN=8)

  +----------------------Xp----------------------+
  |                      +----Jp---+             |
  +-Wd-+--Sp-+---I--+--O-+ALx+-Dmc-+---Mp---+    |
  |    |     |      |    |   |     |        |     |
///// you must.v like.v all the toys.n upstairs.e .
```

5.1.1.2 Functional Correspondences

In (138) and(139) the quantifier *all* is linked in a surprising way. In addition to be linked to the determiner - as expected - it also functions as the head of the object noun phrase. *All* is linked to the noun both by a *J* connection and via the determiner - we get a cyclic structure, which could not be expressed in a stemma (cf. 2.1.1). Normally, *J* connects prepositions to objects. Since there is no preposition here, this is surprising. (139) is in fact analyzed similarly to (140):

```
( 140) You must like all of the toys upstairs.

  Linkage 1, cost vector = (UNUSED=0 DIS=1 AND=0 LEN=9)

  +-----------------------Xp-----------------------+
  |                 +------------MVp-----------+    |
  |                 |        +---Jp---+        |    |
  +-Wd-+--Sp-+---I--+--O-+-Mp+  +-Dmc-+        |    |
  |    |     |      |    |   |  |     |        |    |
///// you must.v like.v all of the toys.n upstairs.e .

  Linkage 2, cost vector = (UNUSED=0 DIS=1 AND=0 LEN=9)

  +-----------------------Xp-----------------------+
  |                          +---Jp---+            |
  +-Wd-+--Sp-+---I--+--O-+-Mp+  +-Dmc-+---Mp---+    |
  |    |     |      |    |   |  |     |        |    |
///// you must.v like.v all of the toys.n upstairs.e .

  Linkage 3, cost vector = (UNUSED=0 DIS=1 AND=0 LEN=12)

  +-----------------------Xp-----------------------+
  |                      +----------Mp---------+    |
  |                      |   +---Jp---+        |    |
  +-Wd-+--Sp-+---I--+--O-+-Mp+  +-Dmc-+        |    |
  |    |     |      |    |   |  |     |        |    |
///// you must.v like.v all of the toys.n upstairs.e .
```

Although Link Grammar fails to express functional correspondences in many cases where one may expect it (as we will see now), here it uses a functional correspondence in a case one hardly thinks of [Quirk et al. 5.16: 258]

We have seen in the VP-internal hypothesis in 2.3.9.3 that some quantifiers like *all* or *both* can “strand”, because the quantifier may remain in the determiners D-position inside the VP:

```
( 141) They all have helped her.

  Linkage 1, cost vector = (UNUSED=0 DIS=0 AND=0 LEN=6)

  +---------------Xp--------------+
  |     +----Sp---+               |
  +--Wd-+   +--E--+---PP--+--Ox-+  |
  |     |   |     |       |     |  |
///// they all have.v helped.v her .

  Linkage 2, cost vector = (UNUSED=0 DIS=3 AND=0 LEN=8)

  +---------------Xp--------------+
  |     +--------Sp-------+       |
  |     +----Bp---+       |       |
  +--Wd-+-R-+-Sp*t+       +--Ox-+  |
  |     |   |     |       |     |  |
///// they all have.v helped.v her .


( 142) They have all helped her.

  +---------------Xp--------------+
  |           +-----PP----+       |
  +--Wd-+--Sp-+    +---E--+--Ox-+  |
  |     |     |    |      |     |  |
///// they have.v all helped.v her .
```

Apart from the wrong linkage in one case, both sentences get very similar linkages indeed, which nicely express the functional relation. But semantically it is hard to account for the fact that all is linked to the verb. (The *E* link is usually used for verb-modifying adverbs).

In their discussion of functional correspondence, Quirk e al. [1985] mention the following structures:

- **Active and passive: [Quirk et al. 2.21:57]**

```
( 143) A number of people saw the accident

  +-----------------------Xp----------------------+
  +----Wd---+---------S--------+-----Os-----+     |
  |   +--Ds-+--Mp-+--Jp-+      |    +---Ds--+     |
  |   |     |     |     |      |    |       |     |
///// a number.n of people.p saw.v the accident.n .
```

```
( 144) The accident was seen (by a number of people)

  +----------------------------Xp---------------------------+
  +-----Wd-----+                   +---J---+                |
  |    +---Ds--+---Ss--+--Pv--+-MVp+ +--Ds-+--Mp-+--Jp-+    |
  |    |       |       |      |    | |     |     |     |    |
///// the accident.n was.v seen.v by a number.n of people.p .
```

While the parses are correct, we can see that Link Grammar is too surface-oriented to express any functional correspondence between the sentences.

- **Copular and complex transitive structures [Quirk et al. 2.22:58]**

```
( 145) I considered her beautiful

  Linkage 1, cost vector = (UNUSED=0 DIS=0 AND=0 LEN=5)

  +-----------------Xp-----------------+
  |             +-------Pa------+      |
  +-Wd-+--Sp*i--+---Ox--+       |      |
  |    |        |       |       |      |
///// I.p considered.v her beautiful.a .

  Linkage 2, cost vector = (UNUSED=0 DIS=2 AND=0 LEN=5)

  +-----------------Xp-----------------+
  |             +-------Op------+      |
  +-Wd-+--Sp*i--+       +---DD--+      |
  |    |        |       |       |      |
///// I.p considered.v her beautiful.a .  [This linkage is wrong]
```

```
( 146) I considered her to be beautiful

  Linkage 1, cost vector = (UNUSED=0 DIS=0 AND=0 LEN=7)

  +---------------------Xp---------------------+
  |             +----TOo----+                  |
  +-Wd-+--Sp*i--+---Ox--+   +-Ix+---Pa--+      |
  |    |        |       |   |   |       |      |
///// I.p considered.v her to be.v beautiful.a .

  Linkage 2, cost vector = (UNUSED=0 DIS=1 AND=0 LEN=7)

  +---------------------Xp---------------------+
  |             +----MVi----+                  |
  +-Wd-+--Sp*i--+---Ox--+   +-Ix+---Pa--+      |
  |    |        |       |   |   |       |      |
///// I.p considered.v her to be.v beautiful.a .

  Linkage 3, cost vector = (UNUSED=0 DIS=3 AND=0 LEN=10)

  +---------------------Xp---------------------+
  |             +----------Pa----------+      |
  |             +----MVi----+          |      |
  +-Wd-+--Sp*i--+---Ox--+   +-Ix+      |      |
  |    |        |       |   |   |       |      |
///// I.p considered.v her to be.v beautiful.a .   [This linkage
is wrong]
```

Note that in the first linkage the subscript *o* in the *TOo* link expresses object control, thus conveying the semantic connection. In the second (questionable) linkage, the non-finite clause is an optional modifier.

```
( 147) I considered that she was beautiful.

  +----------------------Xp----------------------+
  +-Wd-+--Sp*i--+---TH---+-Ce+-Ss-+---Pa---+      |
  |    |        |        |   |    |        |      |
///// I.p considered.v that she was.v beautiful.a .
```

Except for the object control subscript in (146), the functional resemblance between these sentences is not expressed. The only function the *TH* link has - one may guess it - is to link *that* to the verb. This seems a trivial statement, which does not express the fact that the *that*-clause is a sentential complement.

- **Indirect object and prepositional phrases [Quirk et al. 2.23:59]**

This is the distinction between

```
( 148) She gave him an apple
```

and

```
( 149) She gave an apple to him
```

We have already seen in 5.1.1.1 that Link Grammar is unable to express the functional correspondence.

- **Assertions, questions, negations [Quirk et al. 2.49-50:80]**

```
( 150) They often go abroad.

  +------------Xp------------+
  |     +----Sp----+         |
  +--Wd-+    +--E--+-MVp-+   |
  |     |    |     |     |   |
///// they often go.v abroad .


( 151) Do they often go abroad ?

  +---------------Xp--------------+
  |     +------I*d------+         |
  +--Qd-+-SIp+    +--E--+-MVp-+   |
  |     |    |    |     |     |   |
///// do.v they often go.v abroad ?


( 152) They do not often go abroad.

  +-----------------Xp----------------+
  |          +------I*d-----+         |
  +--Wd-+-Sp-+-N-+    +--E--+-MVp-+   |
  |     |    |   |    |     |     |   |
///// they do.v not often go.v abroad .
```

Link Grammar connectors in Link Grammar grammar rules have to specify if the connector they connect to occurs to the left (-) or to the right (+). The English Link Grammar grammar provided with Link Grammar never uses the same link types in both directions. Accordingly, the inverted subject-verb order in question has to be expressed by a different link, suitably called *SI* for subject inversion. Except for this, the functional correspondence is expressed, with an *I* link keeping the verb chain together.

- **Cleft sentences [Quirk et al. 2.59, 18.25]:**

I have discussed in 2.3.6 how a functional dependency grammar could be able to assign the same structure cleft- and non-cleft sentences.

```
( 153) It is John who loves Mary

  +---------------Xp--------------+
  |        +------Bs-----+        |
  |        +----R---+    |        |
  +-Wd-+SFs+-Osi+   +-RS-+--Os-+  |
  |    |   |    |   |    |     |  |
///// it is.v John who loves Mary .
```

Comment:

This is even questionable as a surface structure. The *B* and *R* links are usually employed for relative clauses, to which cleft sentences certainly resemble. But unlike in relative clauses (cf. (166)), *B* and *R* link to the verb instead of to the relativized noun *John,* which can only be accessed indirectly through an *O*i* link. *O*i* is indeed reserved for cleft sentences in [Sleator & Temperley 1998a], although they do not use this term. Even at the surface level, John should be complement rather than object, and subject at deep level.

Although the fact that this is a cleft sentence is expressed by the *O*i* link, the functional relations can only be derived indirectly, and also the connection between the deep subject and the deep main verb is indirect.

- **Pseudo-cleft sentences [Quirk et al. 4.4, 18.25]:**

While the subject is topicalized in cleft sentences, the object is topicalized in pseudo-cleft. Accordingly, in Link Grammar the connection between the verb and the object is expected not to be directly expressed:

```
( 154) What he likes is singing.

Linkage 1, cost vector = (UNUSED=0 DIS=0 AND=0 LEN=8)

  +----------------Xp----------------+
  |     +------Ss*t-----+            |
  |     +---Bsd--+      |            |
  +--Wd-+   +-Ss-+      +--Pg--+     |
  |     |   |    |      |      |     |
///// what he likes.v is.v singing.v .

  Linkage 2, cost vector = (UNUSED=0 DIS=4 AND=0 LEN=8)

  +----------------Xp----------------+
  |     +------Ss*t-----+            |
  |     +---Bsd--+      |            |
  +--Wd-+   +-Ss-+      +--Ost-+     |
  |     |   |    |      |      |     |
///// what he likes.v is.v singing.g .
```

Comment:

The connection between verb and object has to be derived across three links. At least *Bsd* and *Ss*t* were not originally intended to express pseudo-cleft:

Bsd is used for words like "whatever" and "whoever". These words may take object-type relative clauses: however, they simultaneously serve as the subject or object of a main clause.

...

```
    +---------Ss---------+
    +------Bsd--------+  |
    |                 |  |
 Whatever you want to do is fine
```

(Sleator & Temperley 1998a)

Ss#t is used for a few nouns that can take "be+that" as predicates:

The idea was that we would go to London

(Sleator & Temperley 1998a)

- **Fronting [Quirk et al. 2.59]:**

The most simple way to topicalize objects is called fronting:

```
( 155) Bananas I like.

No complete linkages found.
Found 1 linkage (1 had no P.P. violations) at null count 5
  Unique linkage, cost vector = (UNUSED=6 DIS=0 AND=0 LEN=0)

[bananas] [I] [like] [.]
```

Comment:

Finally a grammar that bluntly refuses to parse every linguist's favourite sentence! It has to be added that fronting is conceived of as ungrammatical by many native speakers - but linguists still refuse to notice.

5.1.2 Simple sentences

5.1.2.1 Verbs

Link Grammar manages to parse very complex verb phrases correctly, even if many incorrect ambiguities are reported [Quirk et al. 1985 3.56:153].

```
( 156) I like to have been being examined.

  Linkage 1, cost vector = (UNUSED=0 DIS=0 AND=0 LEN=7)

  +------------------------Xp-----------------------+
  +-Wd-+-Sp*i+-TO-+-If-+--PPf-+--Pg--+----Pv---+     |
  |    |     |    |    |      |      |         |     |
///// I.p like.v to have.v been.v being.v examined.v .

  Linkage 2, cost vector = (UNUSED=0 DIS=3 AND=0 LEN=10)

  +------------------------Xp-----------------------+
  |          +-----------Pg----------+               |
  +-Wd-+-Sp*i+-MVi+-If-+--PPf-+      +----Pv---+     |
  |    |     |    |    |      |      |         |     |
///// I.p like.v to have.v been.v being.v examined.v .

  Linkage 3, cost vector = (UNUSED=0 DIS=4 AND=0 LEN=7)

  +------------------------Xp-----------------------+
  +-Wd-+-Sp*i+-TO-+-If-+--PPf-+--Ost-+----Mv---+     |
  |    |     |    |    |      |      |         |     |
///// I.p like.v to have.v been.v being.v examined.v .

  Linkage 4, cost vector = (UNUSED=0 DIS=9 AND=0 LEN=10)

  +------------------------Xp-----------------------+
  |          +----------Osn----------+               |
  +-Wd-+-Sp*i+-MVi+-If-+--PPf-+      +----Mv---+     |
  |    |     |    |    |      |      |         |     |
///// I.p like.v to have.v been.v being.v examined.v .
```

Also verbal dependencies across the whole sentence, due to question inversion or phrasal verbs pose no problems:

```
( 157) Have you been calling your wife up ?

  +----------------------Xp----------------------+
  |                          +---------K--------+ |
  |      +----PPf---+        +------Os-----+    | |
  +--Qd--+-SIp+     +---Pg--+       +--Ds-+    | |
  |      |    |     |       |       |     |    | |
///// have.v you been.v calling.v your wife.n up ?
```

While the verb phrase performance of Link Grammar is generally quite impressive, there are a number of less satisfactory points, which follow as unordered observations. This list is by no means exhaustive:

- **The analyses are rather crude and hardly semantic**:

They blur important semantic distinctions. In the case of the semi-auxiliary *to have*, e.g., we get two linkages for

```
( 158) I have to win.

  Linkage 1, cost vector = (UNUSED=0 DIS=0 AND=0 LEN=4)

  +-----------Xp----------+
  +-Wd-+-Sp*i+-TO-+-I-+   |
  |    |     |    |   |   |
///// I.p have.v to win.v .

  Linkage 2, cost vector = (UNUSED=0 DIS=3 AND=0 LEN=4)

  +-----------Xp----------+
  +-Wd-+-Sp*i+-MVi+-I-+   |
  |    |     |    |   |   |
///// I.p have.v to win.v .
```

The first linkage is the same as for

```
( 159) I demand to win.

  +------------Xp-----------+
  +-Wd-+-Sp*i-+--TO-+-I-+   |
  |    |      |     |   |   |
///// I.p demand.v to win.v .
```

Which is not really the same. The second linkage is the same as for

```
( 160) I came to win.

  +-----------Xp----------+
  +-Wd-+-Sp*i+-MVi+-I-+   |
  |    |     |    |   |   |
///// I.p came.v to win.v .
```

Which is an unlikely reading for (158), in which the non-finite clause is adverbial.

But at least Link Grammar conveys whether the non-finite clause is obligatory (*TO*) or optional (*MVi*).

- **The coverage of the grammar is not always sufficient**:

E.g. the progressive passive with get [QAL 3.66:161] is not supported:

```
( 161) Things are going.

  Linkage 1, cost vector = (UNUSED=0 DIS=0 AND=0 LEN=3)

  +------------Xp------------+
  +---Wd--+--Spx-+--Pgf-+    |
  |       |      |      |    |
///// things.n are.v going.v .

  Linkage 2, cost vector = (UNUSED=0 DIS=4 AND=0 LEN=3)

  +------------Xp------------+
  +---Wd--+--Spx-+--Ost-+    |
  |       |      |      |    |
///// things.n are.v going.v . [This linkage does not express
passive]


( 162) Things get going.

  +------------Xp------------+
  +---Wd--+--Sp--+--Os--+    |
  |       |      |      |    |
///// things.n get.v going.v .
```

- Link Grammar copes with **subject-verb contractions** in a very peculiar way:

Instead of separating the contracted words, it keeps them together, but manages to parse them nevertheless. At first sight one gets the impression of a subjectless sentence, since there is no *S* link:

```
( 163) I'm quite ready.

  +----------Xp---------+
  |    +-----Pa----+    |
  +-Wd-+    +--EA--+    |
  |    |    |      |    |
///// I'm quite ready.a .
```

5.1.2.2 Nouns, Pronouns, Articles

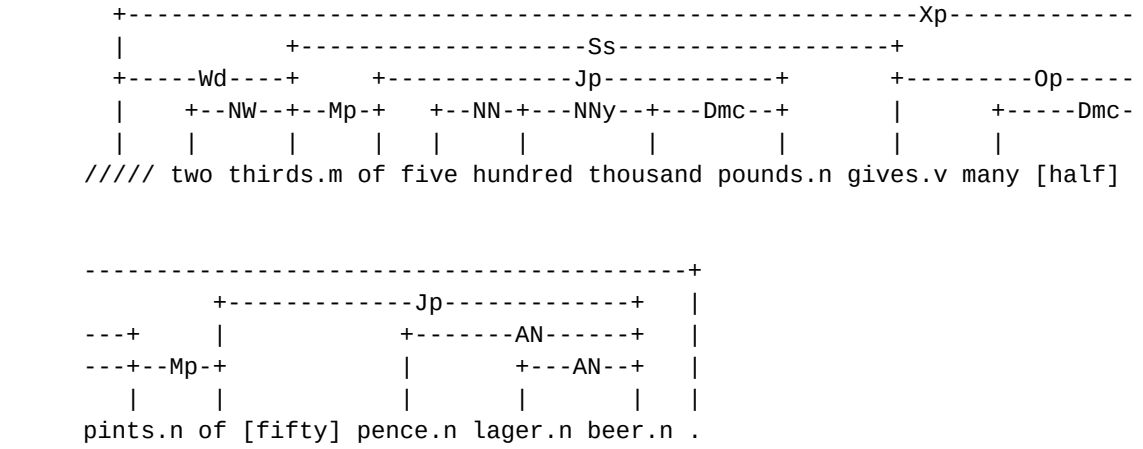
The following observations are again not ordered or exhaustive. They are a selection of reccurring topics in syntactic parsing.

- Link Grammar has an **elaborate system for articles and numerals**:

Often it works and sometimes it doesn't, but I will not describe it in detail here.

(164) Two thirds of five hundred thousand pounds gives many half pints of fifty pence lager beer.

parses, among others, as

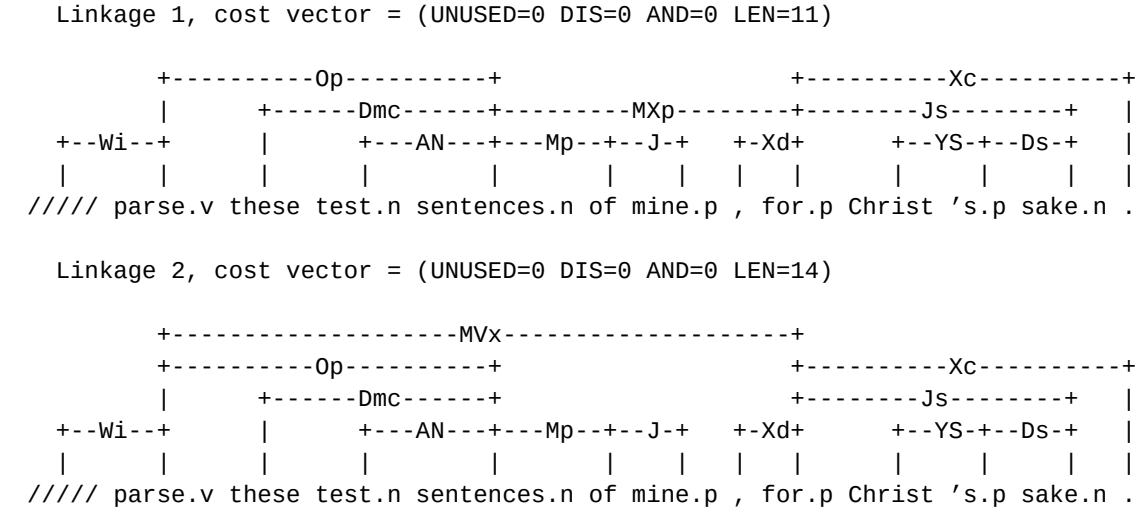


which is not complete, but a good guess.

We have seen in sentence (138) how the syntactically difficult quantifiers *both* and *all* are treated.

- **Genitive constructions** are parsed immaculately:

(165) Parse these test sentences of mine, for Christ's sake.



- Link Grammar does not try to resolve **anaphoric links**:

In a monostratal and context-free system, this would also be difficult, since anaphoric links could cross syntactic links. There is one exception, however, in which anaphoric links are given: relative clauses, the anaphoric link having the label *R*:

```
( 166) The man in the garden whom you know walks off.

  +----------------------------Xp---------------------------+
  |         +------------------Ss------------------+        |
  +----Wd---+    +----Js---+-------Bs-------+      |        |
  |    +-Ds-+-Mp-+  +--Ds--+---R--+-Cr+--Sp-+      +--K--+  |
  |    |    |    |  |      |      |   |     |      |     |  |
///// the man.n in the garden.n whom you know.v walks.v off .

  Linkage 2, cost vector = (UNUSED=0 DIS=1 AND=0 LEN=26)

  +----------------------------Xp---------------------------+
  |         +------------------Ss------------------+        |
  |         +---------------Bs--------------+      |        |
  |         +----------R----------+         |      |        |
  +----Wd---+    +----Js---+      |         |      |        |
  |    +-Ds-+-Mp-+  +--Ds--+      +-Cr+--Sp-+      +--K--+  |
  |    |    |    |  |      |      |   |     |      |     |  |
///// the man.n in the garden.n whom you know.v walks.v off .
```

This creates cyclic links, which could not be represented in a lemma structure.

- Link Grammar copes surprisingly well with **long-distance dependencies**:

Here it can fully profit from its relation to dependency grammar:

```
( 167) Which house did you think I like ?

  +----------------------Xp----------------------+
  |            +-------------Bsm-------------+   |
  |            |      +----I*d---+           |   |
  +--Wq-+-Ds*w-+      +-SIp+     +--Ce-+-Sp*i+   |
  |     |      |      |    |     |     |     |   |
///// which house.n did.v you think.v I.p like.v ?
```

B links link the verb to the raised WH-subject or object and generally express long-distance dependencies:

> B serves various functions involving relative clauses and questions. It connects transitive verbs back to their objects in relative clauses, questions, and indirect questions ...; it also connects the main noun to the finite verb in subject-type relative clauses. (Sleator & Temperley 1998a)

In more complex cases of WH raising, which are also grammatically less acceptable, Link Grammar fails, however:

```
( 168) Which house did you think my father said I like ?

No complete linkages found.
Found 4 linkages (2 had no P.P. violations) at null count 2
  Linkage 1, cost vector = (UNUSED=2 DIS=0 AND=0 LEN=17)

  +---------------------------------Xp-------------------------------+
  |            +-----------------Bsm-----------------+              |
  |            |      +----I*d---+-----Ce----+       |              |
  +--Wq-+-Ds*w-+      +-SIp+     |     +--Ds-+---Ss--+              |
  |     |      |      |    |     |     |     |       |              |
///// which house.n did.v you think.v my father.n said.v [I] [like] ?

  Linkage 2, cost vector = (UNUSED=2 DIS=3 AND=0 LEN=16)

  +---------------------------------Xp---------------------------------+
  |                                              +--------Bs--------+   |
  |                   +-----------Os-----------+------R-----+     |   |
  +--Ws-+-Ds*w-+--Ss--+                 +--Ds-+---Mv--+    +-Sp*i+   |
  |     |      |      |                 |     |       |    |     |   |
///// which house.n did.v [you] [think] my father.n said.v I.p like.v ?


( 169) To whom did the man say you spoke ?

  +----------------------Xp----------------------+
  |             +------I*d------+                |
  |    +---Qd---+---SIs---+     |                |
  +-Wj-+-Jw+    |    +-Ds-+     +-Ce-+--Sp-+    |
  |    |   |    |    |    |     |    |     |    |
///// to whom did.v the man.n say.v you spoke.v ?
```

The linkage in (169) is complete, but incorrect. Especially, there is no link between *spoke* and *to whom*.

```
( 170) Who did the man say you spoke to ?

No complete linkages found.
Found 4 linkages (2 had no P.P. violations) at null count 1
  Linkage 1, cost vector = (UNUSED=1 DIS=0 AND=0 LEN=16)

  +----------------------Xp----------------------+
  |    +--------------B*w--------------+         |
  |    |    +------I*d------+          |         |
  |    |    +---SIs---+     |          |         |
  +-Wq-+    |    +-Ds-+     +-Ce-+--Sp-+         |
  |    |    |    |    |     |    |     |         |
///// who did.v the man.n say.v you spoke.v [to] ?

  Linkage 2, cost vector = (UNUSED=1 DIS=2 AND=0 LEN=13)

  +----------------------Xp----------------------+
  |         +----Os---+----------Bs----------+ |
  +-Ws-+S**w+    +-Ds-+-----R----+--Sp-+-MVp-+ |
  |    |    |    |    |          |     |     | |
///// who did.v the man.n [say] you spoke.v to ?
```

5.1.2.3 Adjectives

Predicative adjectives are linked with a *Pa* link, attributive adjectives with *A*:

```
( 171) The tall woman is strong.

  +-----------------Xp-----------------+
  +--------Wd-------+                  |
  |    +-----Ds-----+                  |
  |    |     +---A--+--Ss--+--Pa--+    |
  |    |     |      |      |      |    |
///// the tall.a woman.n is.v strong.a .
```

- More challenging grammatical phenomena are the rare English **adjectives in postposition** [Quirk et al. 1985 7.21:418]:

Most of them can be regarded as a reduced relative clause. In Link Grammar, they all fail to parse correctly.

```
( 172) Something interesting will happen.

  +---------------------Xp--------------------+
  |       +----------Ss----------+            |
  +---Wd--+----Paf----+          +---I---+    |
  |       |           |          |       |    |
///// something interesting.a will.v happen.v .
```

This looks acceptable at first sight, but the *Paf* link is involved with expletives only - a clear miss.

```
( 173) The boats afloat were not seen by the bandits.

  +------------------------------Xp-----------------------------+
  +----Wd----+                +----Pv----+    +----Jp---+     |
  |    +-Dmc-+-------Spx------+--N-+     +-MVp+  +--Dmc-+     |
  |    |     |                |    |     |    |  |      |     |
///// the boats.n [afloat] were.v not seen.v by the bandits.n .


( 174) At the time appointed they met

  +-------------------Xp-------------------+
  +---------------Wd--------------+        |
  |    +------------CO------------+        |
  |    +---Jp---+                 |        |
  |    |  +-D*u-+---Mv---+        +-Sp-+   |
  |    |  |     |        |        |    |   |
///// at the time.n appointed.v they met.v .


( 175) The person opposite is her father.

  +----------------------Xp---------------------+
  +----------Wd---------+                       |
  |    +-------Ds-------+       +----Ost---+    |
  |    |      +----AN---+---Ss--+   +--Ds--+    |
  |    |      |         |       |   |      |    |
///// the person.n opposite.n is.v her father.n .
```

- **Adjective complementation** [Quirk et al. 1985 7.22:420]:

This parses properly, but the complements are not always recognized as obligatory complements, sometimes only as optional modifiers.

```
( 176) I am able to walk.

  Linkage 1, cost vector = (UNUSED=0 DIS=0 AND=0 LEN=5)

  +--------------Xp-------------+
  +-Wd-+-SX-+--Pa-+-TO-+--I-+   |
  |    |    |     |    |    |   |
///// I.p am.v able.a to walk.v .

  Linkage 2, cost vector = (UNUSED=0 DIS=1 AND=0 LEN=5)

  +--------------Xp-------------+
  +-Wd-+-SX-+--Pa-+-MVi+--I-+   |
  |    |    |     |    |    |   |
///// I.p am.v able.a to walk.v .
```

The *TO* link marks obligatory complements: "TO connects verbs and adjectives which take infinitival complements" (Sleator & Temperley 1998a). In the following example, however, the prepositional complement is marked as an optional modifier.

```
( 177) I know an actor suitable for the part.

  +-------------------------Xp-------------------------+
  |          +----Os---+                +----Js----+   |
  +-Wd-+-Sp*i+    +-Ds-+----Ma---+--MVp--+    +--Ds-+   |
  |    |     |    |    |         |       |    |     |   |
///// I.p know.v an actor.n suitable.a for.p the part.n .
```

5.1.2.4 Adverbs

Although Link Grammar is a syntactic parser, some semantic decisions are needed, e.g. where adverbs should be attached. Link Grammar does not always, but in surprisingly many cases, manage to attach adjectives to the semantically proper place. Sometimes it correctly reports an ambiguity, as in:

```
( 178) She is amazingly intelligent.

  +-----------------Xp-----------------+
  |         +--------Paf-------+       |
  +-Wd-+-Ss-+      +-----EA----+       |
  |    |    |      |           |       |
///// she is.v amazingly intelligent.a .

  Linkage 2, cost vector = (UNUSED=0 DIS=0 AND=0 LEN=5)

  +-----------------Xp-----------------+
  |         +--------Paf-------+       |
  +-Wd-+-Ss-+--EBm-+           |       |
  |    |    |      |           |       |
///// she is.v amazingly intelligent.a .
```

In other cases the attachment is disambiguated:

```
( 179) The car is probably new.

  +----------------Xp---------------+
  +----Wd---+     +------Pa-----+   |
  |    +-Ds-+--Ss-+--EBm-+      |   |
  |    |    |     |      |      |   |
///// the car.n is.v probably new.a .

( 180) The car is extremely new.

  +----------------Xp----------------+
  +----Wd---+     +------Pa------+   |
  |    +-Ds-+--Ss-+      +---EA--+   |
  |    |    |     |      |       |   |
///// the car.n is.v extremely new.a .
```

If adverbs are raised to sentence-initial position, however, Link Grammar links them to the surface subject. This is semantically incorrect.

```
( 181) Apparently, they went abroad.

  +-----------------Xp----------------+
  +--------Wd--------+                |
  |        +----CO---+                |
  |        +--Xc-+   +--Sp-+--MVp-+   |
  |        |     |   |     |      |   |
///// apparently , they went.v abroad .
```

If Link Grammar was verb-centered, like most dependency grammars, this mistake would probably not happen. It shows one of the disadvantages of taking the subject as the root of the sentence (in Link Grammar the *wall* links to the subject and not to the inflected verb.

5.1.3 Functions in Complex Clauses

The "third cycle" of [Quirk et al. 1985], esp. chapters 15-18, deals with functions of complex clauses (like e.g. subordination). Again a couple of observations:

- **Sentential relative clauses:**

Link Grammar is strictly word-based. Like in every dependency theory, it is not possible to modify a constituent as a whole, but only its head, as I have discussed in 2.2.6. Accordingly, it will be difficult to represent sentential relative clauses in Link Grammar [Quirk et al. 1985: 15.57: 1118]. The best solution would be to modify the main verb in the subordinate clause.

```
( 182) He walks for an hour each morning, which would bore me.
[Quirk et al. 1985: 1118]

                      +---------MXsr---------+
             +----Js---+-----Mp-----+         +----------Xc---------+
 +-Ss-+--MVp-+    +-Ds-+     +--DTn-+     +-Xd+-S**w-+---I---+-Ox-+ |
 |    |      |    |    |     |      |     |   |      |       |    | |
he walks.v for.p an hour.n each morning.t , which would.v bore.v me .
```

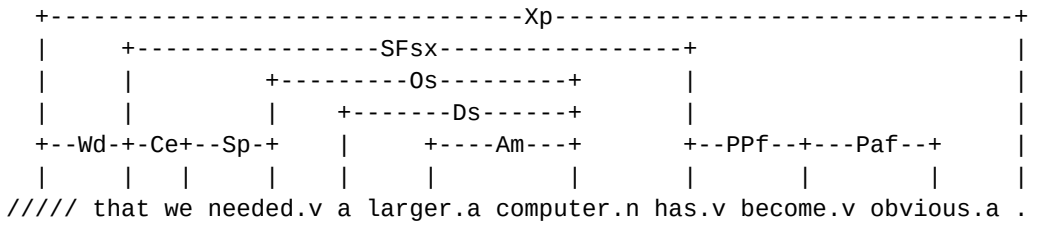
Hour is modified instead of the main verb *walk.* Syntactically this is also possible, but the sentential reading is lacking. Apparently, Link Grammar cannot yet cope with sentential relative clauses:

```
( 183) He walks, which I find boring.

      +-------------MVx-------------+
 +-Ss-+    +-----------Xd-----------+-Xc-+
 |    |    |                        |    |
he walks.v , [which] [I] [find] boring.g .
```

- **Functions of clausal elements:**

Just like on the phrase level, clausal elements have the functions subject, object, complement or adverbial. I have already shown in 5.1.1 that Link Grammar is too surface-oriented and not functional enough to express syntactic functions. A link called complement is unknown, Link Grammar mostly uses grammatical (non-functional) classes, the use of the functional *S* (subject) and *O* (object) links is usually restricted to nouns in surface positions. There are some exceptions, however, in which Link Grammar has a functional touch even at clause level:

(184) That we need a larger computer has become obvious. [Quirk et al. 1985 15.1:1047]



But it is unsatisfactory that sentential subjects (*SF*x*) should be a subclass of filler subjects (*SF*) like existential *it* or *there*.

Functionalism is rather an exception in Link Grammar, not the rule. The use of *object* and *subject* links for nouns mainly creates an illusion, because

- Only the surface functions are recognized. E. g. in passive, cleft- or pseudo-cleft sentences the functional subject is marked as object because at the surface it appears in an object position.
- For clausal elements this terminology is not used, except for the *SF*x* link in (184).
- A complement function is not known either. In most cases, however, optional modifiers and obligatory arguments (objects or complements) are distinguished.
- There are many very specific links which are only used in one particular construction. The over 100 different link types are sometimes loose categories whose subclasses (expressed by subscripting) have little in common.

5.1.4 Conclusions

Link Grammar is not functional (cf. 6.4) As a syntactical, purely grammatical (I mean non-functional) parser, however, Link Grammar is probably faster than any constituent parser and it has a surprisingly large coverage. For most simple "school-grammar" sentences, and still for a majority of complex real-world sentences, it brings up correct linkages.

The fact that over a hundred different link types are used, each of them further complicated by a complex subscripting system, confuses the user. Many link types are too broad categories, on the other hand some link types seem to overlap with others.

5.2 Different Kinds Of Link Types

Link Grammar uses at least two non-standard syntactic links: The anaphoric B link to bind relative pronouns and the wall link, which is present in most sentences. While a root link is present in most dependency grammars, the inconsistent way in which it is used in Link Grammar is unusual (cf. 4.1.4).

A minority of link types is labeled according to their syntactic function (object, subject etc.), but the majority are grammatical, i.e. based on part-of-speech.

5.3 Semantic Dependencies

Some dependencies are semantically hard to account for.

- Commas: Because Link Grammar is completely word-based, every word, including punctuation, has to be linked. In e.g. (181), linking the comma seems to be an ad hoc solution:

```
  +-----------------Xp----------------+
  +--------Wd--------+                |
  |        +----CO---+                |
  |        +--Xc-+   +--Sp-+--MVp-+   |
///// apparently , they went.v abroad .
```

- Staying with the same sentence, as discussed in 5.1.2.4, it is semantically inaccurate to modify the subject by the adverb *apparently*.

- As we have seen in sentence

  (142) They have all helped her.

  the stranded quantifier all is linked to the main verb instead of to the article.

```
  +---------------Xp---------------+
  |           +-----PP----+        |
  +--Wd-+--Sp-+    +---E--+--Ox-+  |
  |     |     |    |      |     |  |
///// they have.v all helped.v her .
```

5.4 Converting Linkages to Constituency

This subchapter would also merit a separate finals paper. In the short space given here I can only give some hints, without making practical texts or attempting an automatic conversion.

We have seen in 2.4.6 that dependency cannot be mapped onto constituency because dependency is not necessarily context-free. But Link Grammar allows no crossing links and thus stays context-free. It should therefore be possible to convert any linkage to a constituent PSG style structure.

Constituents also exist in dependency, as a derived concept (cf. 2.3.13). A dependency constituent consists of a head and all its dependents. In order to recognize all the dependents of a head we need to introduce a principled distinction between head and dependents to Link Grammar, in which linkages are undirected, like in concomitance (cf. 2.2.3.2). Chapter 5.4.1 will address this topic.

The documentation of Link Grammar Version 3.0 [Sleator & Temperley 1998b], which has just become available, also contains some hints on how to convert. Above all, it points out that post-processing uses many constituent concepts. Chapter 5.4.2 will quote them. Unfortunately, it is too late to comment or test the authors' suggestions.

5.4.1 Heads

We have seen in 2.3.2 that the concept of heads is central for dependency. It is as central in X-bar grammar, which is a dependency grammar that lacks non-projectivity but allows intermediate (X') nodes. We will see in 6.2, that in semantics, or at least in the semantic framework TFA I will discuss there, the head notion is just as central. Yet heads are absent from Link Grammar. If we want to be able to convert Link Grammar linkages to dependency or TFA or constituency, recognition of heads is crucial. I have tried to find out if for each kind of link a head can be established. The following

alphabetical list of the about 90 links of Link Grammar Version 2.1 is an excerpt form the documentation enriched by my suggestions. It will only make sense to people acquainted with Link Grammar. All others are strongly advised to refer to this documentation, which is provided with the system ([Sleator & Temperley 1998a] for Version 3.0).

Because links are dependent on the direction in the text it suffices to indicate for any link type whether the head is to the LEFT (indicated by '\') or the RIGHT (indicated by '/') end of the link. In addition, the head is **bold** in the text. For coordinating links and other non-dependent relations I suggest to use Engel's concomitance relations (cf. 2.2.3.2) , which is indicated by '-':

Chart I. LINK TYPES AT A GLANCE, enriched by head-dependent direction

A / connects pre-noun ("attributive") adjectives to following nouns:
"The BIG **DOG** chased me", "The BIG BLACK UGLY DOG chased me".

AA / is used in the construction "How [adj] a [**noun**] was it?". It
connects the adjective to the following "a".

AF / connectives adjectives to verbs in cases where the adjective
is fronted, such as questions and indirect questions: "How
BIG **IS** it?"

AL / connects a few determiners like "all" or "both" to following
determiners: "ALL **THE** people are here".

AN / connects noun-modifiers to following nouns: "The TAX **PROPOSAL**
was rejected".

AZ \ connects the word "as" back to certain verbs that can take
"[obj] as [adj]" as a complement: "He **VIEWED** him AS stupid".

B \ serves various functions involving relative clauses and questions.
It connects transitive verbs back to their objects in cases
like relative clauses and questions ("WHO did you **HIT**?"); it
also connects the main noun to the finite verb in subject-type
relative clauses ("The **DOG** who CHASED me was black").
THIS IS A PROBLEM CASE, AS SOMETIMES – SEEMS TO FIT BETTER

BI \ connects form of the verb "be" to certain idiomatic expressions:
for example, cases like "He **IS** PRESIDENT of the company".

BT / is used with time expressions acting as fronted objects: "How many
YEARS did it **LAST**?".

BW / connects "what" to various verbs like "think", which are not really
transitive but can connect back to "what" in questions: "WHAT
do you **THINK**?"

C / links conjunctions to subjects of subordinate clauses ("He left
WHEN HE saw me"). it also links certain verbs to subjects
of embedded clauses ("He **SAID** HE was sorry").
FIRST MAY BE FUNCTIONAL HEAD

CC - connects clauses to following coordinating conjunctions ("SHE left
BUT we stayed").

CO / connects "openers" to subjects of clauses: "APPARENTLY / ON Tuesday ,
THEY went to a movie".
SHOULD RATHER MODIFY THE VERB

CQ - connects to auxiliaries in comparative constructions involving

s-v inversion: "SHE has more money THAN DOES Joe".

CX - is used in comparative constructions where the right half of the comparative contains only an auxiliary: "She has more money THAN he DOES".
EITHER INTRODUCES CLAUSE \ (LIKE COORD) OR SUBSTITUTES OBJ /

D / connects determiners to nouns: "THE **DOG** chased A CAT and SOME BIRDS".
UNLESS UNDER DP HYPOTHESIS

DD / connects definite determiners ("the", "his") to number expressions certain things like number expressions and adjectives acting as nouns: "THE **POOR**", "THE TWO he mentioned".
UNLESS UNDER DP HYPOTHESIS

DG / connects the word "The" with proper nouns: "the **Riviera**", "the Mississippi".

DP / connects possessive determiners to gerunds: "YOUR **TELLING** John to leave was stupid".

DT / connects determiners to nouns in idiomatic time expressions: "NEXT **WEEK**", "NEXT THURSDAY".

E / is used for verb-modifying adverbs which precede the verb: "He APPARENTLY not **COMING**".

EA / connects adverbs to adjectives: "She is a VERY **GOOD** player".

EB \ connects adverbs to forms of "be" before an object or prepositional phrase: "He IS APPARENTLY a good programmer".
SHOULD PERHAPS MODIFY NOUN

EC / connects adverbs to comparative adjectives: "It is MUCH **BIGGER**"

EE / connects adverbs to other adverbs: "He ran VERY **QUICKLY**".

EF / connects the word "enough" to preceding adjectives and adverbs: "He didn't run QUICKLY **ENOUGH**".

EI / connects a few adverbs to "after" and "before": "I left SOON **AFTER** I saw you".

EN / connects certain adverbs to expressions of quantity: "The class has NEARLY **FIFTY** students".

ER - is used the expression "The x-er..., the y-er...". it connects the two halfs of the expression together, via the comparative words (e.g. "The FASTER it is, the MORE they will like it").

FM \ connects the preposition "from" to various other prepositions: "We heard a scream **FROM** INSIDE the house".

G - connects proper noun words together in series: "GEORGE HERBERT WALKER BUSH is here."
- OR / OR \ : UNIMPORTANT

GN - (stage 2 only) connects a proper noun to a preceding common noun which introduces it: "The ACTOR Eddie MURPHY attended the event".

H / connects "how" to "much" or "many": "HOW **MUCH** money do you have".

I \ connects certain words with infinitive verb forms, such as modal verbs and "to": "You **MUST** DO it", "I want **TO** DO it".

IN \ connects the preposition "in" to certain time expressions: "We did it **IN** DECEMBER".

J \ connects prepositions to their objects: "The man **WITH** the HAT is here".

JG \ connects certain prepositions to proper-noun objects: "The Emir

OF KUWAIT is here".

JQ / connects prepositions to question-word determiners in "prepositional questions": "IN **WHICH** room were you sleeping?"

JT \ connects certain conjunctions to time-expressions like "last week": "**UNTIL** last WEEK, I thought she liked me".

K / connects certain verbs with particles like "in", "out", "up" and the like: "He STOOD **UP** and WALKED **OUT**".

L / connects certain determiners to superlative adjectives: "He has THE **BIGGEST** room".

LE \ is used in comparative constructions to connect an adjective to the second half of the comparative expression beyond a complement phrase: "It is more **LIKELY** that Joe will go THAN that Fred will go".

M \ connects nouns to various kinds of post-noun modifiers: prepositional phrases ("The **MAN** WITH the hat"), participle modifiers ("The **WOMAN** CARRYING the box"), prepositional relatives ("The **MAN** TO whom I was speaking"), and other kinds.

MG \ allows certain prepositions to modify proper nouns: "The **EMIR** OF Kuwait is here".

MV \ connects verbs and adjectives to modifying phrases that follow, like adverbs ("The dog **RAN** QUICKLY"), prepositional phrases ("The dog RAN IN the yard"), subordinating conjunctions ("He LEFT WHEN he saw me"), comparatives, participle phrases with commas, and other things.

MX - OR \ connects modifying phrases with commas to preceding nouns: "The DOG, a **POODLE**, was black". "JOHN, **IN** a black suit, looked great".

N \ connects the word "not" to preceding auxiliaries: "He **DID** NOT go".

ND / connects numbers with expressions that require numerical determiners: "I saw him THREE **WEEKS** ago".

NF / is used with NJ in idiomatic number expressions involving "of": "He lives two **THIRDS** OF a mile from here".

NI \ OR - is used in a few special idiomatic number phrases: "I have BETWEEN 5 AND 20 dogs".

NN - or / connects number words together in series: "FOUR HUNDRED THOUSAND people live here".

NR / connects fraction words with superlatives: "It is the THIRD **BIGGEST** city in China".

NS / connects singular numbers (one, 1, a) to idiomatic expressions requiring number determiners: "I saw him ONE **WEEK** ago".

NW / is used in idiomatic fraction expressions: "TWO **THIRDS** of the students were women".

O \ connects transitive verbs to their objects, direct or indirect: "She **SAW** ME", "I **GAVE** HIM the BOOK".

OD \ is used for verbs like "rise" and "fall" which can take expressions of distance as complements: "It **FELL** five FEET".

OF / connects certain verbs and adjectives to the word "of": "She **ACCUSED** him OF the crime", "I'm **PROUD** OF you".

OT \ is used for verbs like "last" which can take time expressions as objects: "It **LASTED** five HOURS".

P \ connects forms of the verb "be" to various words that can be its complements: prepositions, adjectives, and passive and progressive participles: "He **WAS** [ANGRY / IN the yard / CHOSEN / RUNNING]".

PF / is used in certain questions with "be", when the complement need of "be" is satisfied by a preceding question word: "WHERE **ARE** you?", "WHEN will it **BE**?"

PP \ connects forms of "have" with past participles: "He **HAS** GONE".

Q / is used in questions. It connects the wall to the auxiliary in simple yes-no questions ("///// **DID** you go?"); it connects the question word to the auxiliary in where-when-how questions ("WHERE **DID** you go").

QI \ connects certain verbs and adjectives to question-words, forming indirect questions: "He **WONDERED** WHAT she would say".

R \ connects nouns to relative clauses. In subject-type relatives, it connects to the relative pronoun ("The **DOG** WHO chased me was black"); in object-type relatives, it connects either to the relative pronoun or to the subject of the relative clause ("The DOG THAT we chased was black", "The DOG WE chased was black").

RS \ is used in subject-type relative clauses to connect the relative pronoun to the verb: "The dog **WHO** CHASED me was black".

RW - connects the right-wall to the left-wall in cases where the right-wall is not needed for punctuation purposes.

S / connects subject nouns to finite verbs: "The DOG CHASED the cat": "The DOG [**IS** chasing / HAS chased / WILL chase] the cat".
ROOT LINK TO SUBJ MAY NECESSITATE \

SF / is a special connector used to connect "filler" subjects like "it" and "there" to finite verbs: "THERE **IS** a problem", "IT IS likely that he will go".

SFI \ connects "filler" subjects like "it" and "there" to verbs in cases with subject-verb inversion: "**IS** THERE a problem?", "**IS** IT likely that he will go?"

SI \ connects subject nouns to finite verbs in cases of subject-verb inversion: "**IS** JOHN coming?", "Who **DID** HE see?"

TA / is used to connect adjectives like "late" to month names: "We did it in LATE **DECEMBER**".

TD \ connects day-of-the-week words to time expressions like "morning": "We'll do it **MONDAY** MORNING".

TH \ connects words that take "that [clause]" complements with the word "that". These include verbs ("She **TOLD** him THAT..."), nouns ("The **IDEA** THAT..."), and adjectives ("We are **CERTAIN** THAT").

TI \ is used for titles like "president", which can be used in certain cirumstances without a determiner: "**AS** PRESIDENT of the company, it is my decision".

TM \ is used to connect month names to day numbers: "It happened on JANUARY 21".

TO \ connects verbs and adjectives which take infinitival complements to the word "to": "We **TRIED** TO start the car", "We are **EAGER** TO do it".

TQ / is the determiner connector for time expressions acting as fronted objects: "How MANY **YEARS** did it last".

```
TS \ connects certain verbs that can take subjunctive clauses as
       complements - "suggest", "require" - to the word that: "We
       SUGGESTED THAT he go".

TY \ is used for certain idiomatic usages of year numbers: "I saw him on
       January 21 , 1990 ". (In this case it connects the day number to
       the year number.)

U / is a special connector on nouns, which is disjoined with both the
       determiner and subject-object connectors. It is used in idiomatic
       expressions like "What KIND_OF DOG did you buy?"

UN \ connects the words "until" and "since" to certain time phrases like
       "after [clause]": "You should wait UNTIL AFTER you talk to me".

V \ connects various verbs to idiomatic expressions that may be non-adjacent:
       "We TOOK him FOR_GRANTED", "We HELD her RESPONSIBLE".

W \ connects the subjects of main clauses to the wall, in ordinary
       declaratives, imperatives, and most questions (except yes-no
       questions). It also connects coordinating conjunctions to
       following clauses: "We left BUT SHE stayed".

WN \ connects the word "when" to time nouns like "year": "The YEAR WHEN
       we lived in England was wonderful".

WR / connects the word "where" to a few verbs like "put" in questions like
       "WHERE did you PUT it?".

X - is used with punctuation, to connect punctuation symbols either to
       words or to each other. For example, in this case, POODLE
       connects to commas on either side: "The dog , a POODLE , was
       black."

Y / is used in certain idiomatic time and place expressions, to connect
       quantity expressions to the head word of the expression: "He
       left three HOURS AGO", "She lives three MILES FROM the station".

YP / connects plural noun forms ending in s to "'" in possessive
       constructions: "The STUDENTS ' rooms are large".

YS / connects nouns to the possessive suffix "'s": "JOHN 'S dog is black".
      BOTH ABOVE LIKE PREP.

Z \ connects the preposition "as" to certain verbs: "AS we EXPECTED, he was
       late".
```

5.4.2 Hints from the Makers of Link Grammar

The authors of Link Grammar suggest a different method for converting linkages to constituent structures. Since this document [Sleator & Temperley 1998b] appeared only a month before the completion of my paper it is too late to test their ideas. I quote them uncommented:

```
The constituent structure of sentences, while not absolutely explicit,
is also quite "close to the surface" in linkage structures. This
requires some explanation. Imagine a linkage as a graph through which
paths can be traced, similar to a street map. Constituents can be
defined as sets of words which can be reached from certain links,
tracing in a certain direction. ...

A fairly robust system for identifying constituents from linkages can
be quite easily specified in this way. Such a system is outlined below
for the most common constituent types. (Some further details would be
```

needed to identify these constituents in all cases.) In this table, "X-r" means "everything reachable from an X link tracing to the right": "X-l" means the same, tracing to the left.

> VP (Verb Phrase): S-r, Mv-r, Mg-r
> PP (Prepositional Phrase): MVp-r, Mp-r, Pp-r, CO-r
> NP (Noun Phrase): O-r; J-r; SI-r; MX#*-r; S-l (but not tracing through W, CC, CO, R, or C from the left end of the S)
> S (Clause): C-r, W-r, R-r, QI*d-r
> AdjP (Adjective Phrase): A-l, Pa-r, Ma-r

In all cases do not trace through B when it extends to the left of the starting link.

These constituent link-groups correspond very closely to the domains used in post-processing (...). In fact, the domains of post-processing essentially identify clauses, in a very similar way. The difference is that post-processing domains are sets of links, not words.

(Sleator & Temperley 1998b)

6. Syntactic vs. Semantic: The Semantic Interface

The core of this paper is syntax. But because one of the aims of syntactical representation is to constitute a bridgehead to semantics – especially so functional structures like syntactic dependency structures, I cannot completely neglect it.

6.1 Truth Theory

6.1.1 Stepping Up the Meta-Levels

In order to assess whether an utterance is true, some sort of correspondence between language and reality has to be found. This, however, as e.g. the Vienna Positivists have shown, is simply impossible. Tarski is criticized, it is shown that the gap between model and reality cannot be bridged, only model levels are being shifted as we transgress from meta- to metametalevel, and so on, but these metalevels are as unrelatable to reality as language itself. Ayer [1936] suggests that the only possible way to bridge the gap is our *knowledge*.

Given the nature of the ExtrAns project, one might argue that the UNIX commands belong to a semantically controlled language which is clearly defined and not subject to any deconstruction. Moreover, the UNIX commands constitute a model world in which the impossible mapping to an outside world can be neglected. This is true, but the manpages describing the commands are written in natural language, which is saturated with the problems outlined above, i.e. they are undefinable, fail theta theory, and a "correspondence" to the outside world, which would prove or disprove truth, cannot be established.

Because the meaning of any word is essentially undefinable, it does not even matter if meaning can be compositionally derived – it is unattainable anyway. We may argue that if meaning is undefinable it is still experienceable. To experience means to bridge the gap between model and outer world, it is the only way to establish a correspondence between them. I agree with Ayer's [1936] argument that knowledge can be the only bridge between them. Accordingly, I maintain that derivational semantics is illusory until *machine knowledge* and *machine experience* make progress.

6.1.2 After Babel

No matter how far we step up through the meta-levels we can never reach reality. Therefore, the tower of Babel was never completed, because, as the legend

says, nobody was able to understand anybody else. But its story survived – a fact which contradicts the legend. The other contradicting fact is hat we build towers much higher than Babel would have been. Since Wittgenstein's *Sprachspiel* we know that it suffices to identify a brickstone to be able to put them on top of each other, even if we do not know all the details about a brickstone. A functional, process-oriented *knowledge* suffices. Similarly, it may suffice for a Q&A system to identify predicates and arguments. If the logical conversion of a query leads to e.g. copy(file-1, file-2) a simple match with the logical conversions of the answer candidates may suffice to produce the correct result.

The unbridgeable gap between reality and model seems to shrink down to a simple pattern-matching or a Prolog unification. This trick out of the magician's hat has of course nothing to do with semantics or meaning, but it works and users will think that semantics is involved. In fact it is just a series of complex conversions between complex strings and bracketed structures. Because these conversions have nothing to do with meaning, they can on the one hand be calculated, on the other hand they will sometimes produce funny results – but sometimes work very nicely too.

This suggests that a functional dependency structure, a tectogrammatical structure as described by Sgall, Hajičová, Panevová [1986] (cf. 2.2.5.1) may indeed be sufficient for many tasks, e.g. for ExtrAns.

On the lexical meaning level, what we can do in addition is to convert synonyms or near-synonyms or hypo- and hyperonyms into each other, with increasing number of hits and errors i.e. increasing *recall* and decreasing *precision*. For this purpose one uses thesauri, semantic nets or hierarchies like *WordNet*. I will not deal with this topic at all here.

To summarize, we have reduced truth theory to pattern-matching and take words or a class of near-synonyms or whatever as meaning atoms. Because utterances consist of several words all that remains to do is to express the relations between them in a principled way. Here we follow the well-established tradition of using predicate-argument structures. Inherently, syntactical dependency structures are predicate-argument structures, in the form

(185) $\mathrm{Head(dependent_1, \ldots, dependent_n)}$.

with recursions, e.g.

(186) Head(Subhead(Subsubhead, ... Subsubhead), ..., Subhead).

Which leaves us with the following questions:

(I.) Which word forms have such a *pred(arg)* relation, i.e. are linked with each other ? And, in order to answer this question, what does A(B) express ? And in which sense does it express something different from B(A) ? We have already outlined an answer in 2.3.2.2, which we will take up below.

(II.)Which is the head, which are the dependents ? This is the question of 2.3.2 again, only on a semantic level. The fact that Zwicky [1985] does not achieve the same results with the semantic arguments and all other arguments, or that Jung [1995: 85] decides to exclude this argument for the definition of syntactic heads may suggest that dependency structures on these two levels are not identical. Therefore,

(III.)Is there a difference between semantic and syntactical dependency ? If so, can they be mapped onto each other in a principled way ? We hope that they are as close to each other as possible – here expressed for Word Grammar:

> A WG [Word Grammar] grammar generates a semantic structure which parallels the syntactic structures ... The parallels are in fact very close, ... virtually every word is linked to a single element in the semantic structure, and the dependency relations are typically matched by one of two relations between their meanings: dependency or identity. Moreover, if word A depends on word B, and the semantic relation between them is dependency, then the dependency nearly always goes in the same direction as in the syntax – the meaning of A depends on that of B.
> (Hudson 1990: 123)

6.2 Topic-Focus Articulation (TFA)

TFA goes back to the Prague School (cf. 2.2.5) and was first described in Sgall, Hajičová, Panevová [1986] and developed by many others, e.g. Hajičová, Skoumalová, Sgall [1995] and Peregrin [1996].

Let us address the questions posed above. First, in (I.), the meaning of *A(B)* or *B(A).* As we have seen in 2.3.2.2, *sleeps(Peter)* means that there is a set of sleepers, and that Peter is one of them. It means that 'sleep-ness' is a property of Peter. It is generally assumed that this should be the logical representation of the unmarked sentence *Peter sleeps.*

The question is then, what *Peter(sleeps)* could mean, if anything. It generally means that there is set of possible Peters, one of them is a sleeper, i.e. that sleeping is one of the activities of the set of activities of Peter. It means that 'Peter-ness' is the property of the sleeper. If we know that there is a sleeper, but we do not know about the sleeper's identity, the property 'Peter-ness' is what we want to know. In terms of topic and focus, *Peter* becomes the focus, whereas *sleeps* is the focus in the non-marked

sentence. If we ask the question *WHO sleeps ?* or utter the marked assertion *It is PETER who sleeps*, then *Peter(sleeps)* is semantically more appropriate.

Let us consider adjectives. *Tall(Peter)* means that of all things tall, Peter is one. We are asserting tall-ness to *Peter*. The argument *Peter* modifies the predicate *tall*. In opposition, *Peter(tall)* means that of all possible Peters one is tall. The predicate *Peter* is modified by the argument *tall*.

These findings correspond to the distinction between predicative and attributive adjectives: For predicative adjectives, *Peter* modifies the predication (as we have seen in 2.2.3.1.4), just like in the usual analysis for verbs: *tall(Peter)*. In attributive adjectives, however, *Peter* is modified by *tall*, and the appropriate logical representation is *Peter(tall)*, because *tall Peter* is a kind of *Peter*.

6.2.1 Presupposition and Assertion

If we think of *presupposition* theory, Peter's tallness is asserted in *tall(Peter)* and presupposed in *Peter(tall)*. A predication can be negated, but not an argument.

The debate about presupposition and assertion originates from the dispute between Russell [1905] and Strawson [1952]. Considering the classical example sentence

(187) The King of France is bald.

the problem is that France does not have a king any more, the king of France does hence not exist and presupposition fails. Note that the assertion, i.e. that what the sentence is *about*, is about *Baldness*, i.e. *bald(King)*. The king is presupposed. But what happens if the presupposition is false ?

6.2.1.1 Russell's Wide Scope

According to Russell, the sentence is simply FALSE [I shall write Boolean values in capitals henceforth]. The logical representation for (187) is, in his view:

(187a) ∃(king)∧bald(king)

a composed statement , which is negateable in two ways. Let us say that [A] =(187), [C]= ∃(king) and [D]= bald(king). If (187) is false, then either because [D] is false, or because [C] is false. In a truth-table this looks as follows:

Russell's Truth Values				"narrow scope"	"wide scope"
	[C]	[D]	[A]=[C]∧[D]	[B1]=[C]∧¬[D]	[B2]=¬[C]∧¬[D]
α.	T	T	T	F	F
β.	T	F	F	T	F
γ.	F	T	F	F	F
δ.	F	F	F	F	T

Table 5: Russell's Truth Values

As we see, [A] can be wrong or negated in two ways, in [B1] (which is called *narrow scope negation*) because the king exists, but is not bald, in [B2*] (wide scope negation*) because he does not exist. Theoretically, there is a third possibility, in which [D] but ¬ [C]. The fact that this system could express such semantic nonsense is a first indication of its flaws. The natural language representation for [B2] is

(187b1) The King of France is not bald, because he does not exist.

This shows some more problems. Accordingly, (187) should really be

(187a) The King of France exists and is bald.

But (187) and (187a) are not the same statements. (187a) rather contains an assertion, a reference to the king's existence. Russell's postulation is also unfortunate in the sense that even if we accept wide scope negation of [B2], it is far more unlikely and should not appear on the same level as [B1]. Using a two-bit truth-value, with the assertion bit *low* and the presupposition bit *high*, would be more satisfactory there. Assuming *11* for TRUE, we recursively decrement for each negation interpretation, first reaching the far more plausible narrow scope at *10*, then if the interpretation fails we re-iterate and come to the logically impossible *01* and finally wide-scope *00*.

6.2.1.2 Strawson's presupposition failure

(187a) contains an assertion, a reference to the king's existence. Strawson [1952] points out that an utterance presupposes its truth. If we explicitly say that "it is true that X" we refer to it. This is a statement at a different level. In analogy, (187) and (187a) are on different levels. For Strawson, presupposition failure is a failure of Austin's *felicity* conditions.

6.2.1.3 Presupposition as Topic

Peregrin [1996] uses the same argument in a Montagovian context. Let us return to our above example of *tall(Peter)* and *Peter(tall)* in order to answer question (II), at least for *subject+verb* and *adjective+noun* structures. Only predicates can make assertions. If the predication *tall(Peter)* is TRUE then it asserts the assignment of the property *tall* to the

object *Peter*. The existence of Peter is presupposed. *tall(Peter)* corresponds to the natural language statement

(188) Peter is tall.

and *bald(king)* to (187), or *sleeps(Peter)* to *Peter sleeps*.

> [W]hat really makes a sentence into a predication is the fact that one of its parts is "about" the other part. The (semantic) subject is what the sentence is about, predicate is what it says about the subject. What does this "*about*" mean? Well, it, first and foremost, means that the subject is taken for granted for the whole sentence, its existence is not being disputed. This is to say that the subject is connected with a presupposition.
>
> (Peregrin 1996: 239)

6.2.2 Attributive and Predicative Adjectives

If *tall(Peter)* is a comment about Peter, then *Peter(tall)* is a comment about tallness. If we answer a question like *Who is tall?* or focus on Peter in any other way, e.g. by saying *It is Peter who's tall* we speak about tallness, with Peter as a focus: *Peter(tall).*

In *Tall Peter sleeps* the focus is *sleep*, but what is the topic, what are we commenting about, what is presupposed ? First and foremost, *Peter*, of course. That he should be tall is a presupposition at second level. *(sleeps(Peter(tall)).*

We can therefore – provisionally – suggest that the difference between attributive and predicative adjectives is expressed in the direction of the semantic dependency. As the name *attributive* suggests, we are really talking about the noun, which is hence in focus, to which we attribute something: *noun(adjective).* But in predicative adjectives we make a predication: *adjective(noun).* This perfectly corresponds to Tesnière's syntactical treatment of adjectives. He takes the predicative adjective up into the verb nucleus:

(189) Sleepy Peter is tall vs. Tall Peter sleeps

This means that – as a first approximation – syntactical dependency and semantic dependency are in parallel.

These findings also mean that a common of treating adjectives as intersection, e.g.

(190) exist(X) ∧ tall(X) ∧ peter(X).

is fundamentally flawed, for three reasons:

- A predication does not make any statement about the existence of the object in question, regardless if this object has an assertive or a predicative adjective.
- Distinctions between predicative and attributive adjectives are completely blurred. This is a substantial imprecision.
- If we intersect properties instead of modifying them, a *big mouse* will be bigger than a *small elephant*.

There are cases where adjectives behave differently. In e.g. our example of *the French house is big* the adjective *big* does not modify *house* only, but the entire N' *French house*, as we have discussed in 2.2.6.1.1. Here we need take French house into one single node by a Lexical Rule as in LFG or by a translation rule (cf. 2.1.4).

6.2.3 TFA in A Formal Framework

This subchapter mainly follows Peregrin [1996]. We have seen that *tall(Peter)* is a comment about Peter, and *Peter(tall)* is a comment about tallness with *Peter* as focus. While it is generally unusual to assert nouns, it is possible to do so in special cases. When asking *Who is tall?* We are likely to answer with a verbless clause, in which the predicate is *Peter*, anyway. The claim made by Peregrin [ibid.:238] is simply that Peter remains predicate even if we mention the redundant predicative adjective, or, alternatively, the argument *(tall)* is inherently present even in a verbless clause. If one is still skeptical about nouns as predicates, as soon as one lambda-abstracts them they conform to our expectations.

> [I]f we say ([JOHN walks]), then what we express seems to be not the property of walking assigned to the individual John, but rather the property of being John assigned to an anonymous walker. ... One might here evoke the idea that the power of TFA is reminiscent of lambda abstraction: what we do when focusing *John* resembles what we do when making a predicate λf.f(**John**), out of **John** and then applying it to **Walks**.
>
> (Peregrin 1996: 238)

But unfortunately *λf.f(John)(Walks)* lambda-converges to *Walks(John)*. But because of the presupposition arguments presented above, because the focus is asserted while the topic is presupposed, Peregrin stresses that one should retain the possibility to make distinction between *John(Walks)* and *Walks(John)*.

He therefore suggests the following semantic definition: Assuming that ||X|| is the extension of X and |X| the corresponding proposition, and

```
( 191) |X| = ||X|| if X is a sentence,
           = ||∃y.y=X|| if X is a term,
           = ||∃y.X(y)|| if X is a unary predicate
```

then we can define P{S} or "Real" predication:

(192) ||P{S}|| = TRUE iff |S|=TRUE ∧ ||P(S)||=TRUE
= FALSE iff |S|=TRUE ∧ ||P(S)||=FALSE
= NIL iff |S|=FALSE

Every sentence requires at least one element which is not presupposed. For *John loves Mary* we now get 5 readings, depending on focus as e.g. expressed by intonation. For *John loves MARY* we e.g. have

(193) λf.f(**Mary**){λy.**love**(**John**,y)}.

6.2.3.1 Scope of Negation

Sgall, Hajičová, Panevová [1986] show that TFA can be used to disambiguate natural language considerably. In their discussion of negation scope [ibid.:244-249] they show that in most cases, negation negates the focus.

6.2.3.2 Quantifier Scope

Also Peregrin [1996: 241] shows that TFA can help to find out the most probable reading. For e.g. ambiguous quantifier scope sentence

(194) Every man loves a woman

we get the following TFA forms (focus capitalized, as e.g. in pronunciation):

(194a) Every man LOVES A WOMAN:
λM.M(λx.∃y.(**woman**(y)∧**love**(x,y))){λQ.∀x.(**man**(x)→Q(x))}
(194b) EVERY MAN LOVES a woman:
λM.M(λy.∀x.(**man**(x)→**love**(x,y))){λQ.∃y.(**woman**(y)∧Q(y))}
(194c) Every man loves A WOMAN:
λQ.∃y.(**woman**(y)∧Q(y)){ λy.∀x.(**man**(x)→**love**(x,y))}
(194d) EVERY MAN loves a woman:
λQ.∀x.(**man**(x)→Q(x)){λx.∃y.(**woman**(y)∧**love**(x,y))}

If felicity conditions are met, (192) applies, and (194a) and(194c) reduce to (194a+c'), while (194b) and (194d) reduce to (194b+d'):

(194a+c'): ∃y.(**woman**(y)∧∀x.(**man**(x)→**love**(x,y))}

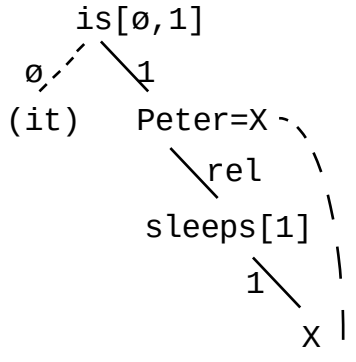
(194b+d'): ∀x.(**man**(x)→∃y.(**woman**(y)∧**love**(x,y))}

These are exactly the two semantic interpretations we know. TFA allows us to attribute them to the corresponding intended meaning, as e.g. expressed by articulation or by the assumption that the egining of a sentence is more likely to be the focus: reading (194b+d') will be more probable.

6.3 Semantic Heads

While a text-based system like e.g. ExtrAns has no access to articulation, TFA still has two crucial benefits:

- We can easily postulate and code a favourite reading. For *subject+verb* constructions, the reading with verb=predicate=focus will be the favourite reading, so if we copy the syntactic dependency structure *verb(subject)* unchanged to the semantic level, most cases will make correct predictions. In (194), (194b+d) is preferred. If we e.g. use a rule that only builds up logical representations which first quantify the first argument (the subject) we will only get the preferred reading.
- There are many syntactic means for expressing topic and focus. Let us take *cleft-sentences* as an example. We have seen in 6.2.2 that TFA allows us to distinguish elegantly between predicative and assertive adjectives. Similarly, we may suggest that relative clauses modify nouns, i.e. the relative clause is topic to the local focus noun. Accordingly, for *Peter, who sleeps, breathes* we get *Breathes(Peter(Sleeps)),* while for *Peter, who breathes, sleeps* we get *Sleeps(Peter(Breathes)).* If we compare a typical topicalisation like It is Peter who sleeps we get a syntactical dependency which roughly looks like



(195)

Irrespective of whether we want to keep the dummy-*it* at the syntactic level or not, the directly derivable semantic dependency is *is(Peter(sleeps)),* or if we include a routine for deleting the dummy-*is* in the mapping to semantics, *Peter(sleeps),* as discussed in 6.2.3.

6.3.1 Fronting, Non-Projectivity and TFA

In 2.3.6 I have shown how a dependency parser can assign the same structure to unmarked and marked versions (cleft, pseudo-cleft, etc.) of a sentence. We have just seen, however, that a TFA dependency analysis even *should* report different parses.

Because fronting is a typical method of topicalisation, word-order has to be taken into account again.

In 6.2 we have seen that if we are interested in the identity of somebody who does something (e.g. *PETER sleeps*) the correct TFA dependency analysis is *Peter(sleeps).* When we ask a question about the identity of something, this something immediately becomes the focus and should be the sentence predicate (e.g. *who(sleeps)*).

In 3.4.3 I have discussed that, as far as I can see, after implementing argument composition for the verb chain the last major remaining non-projective structure for English are long-distance WH-movements. I provisionally suggest that if we parse for TFA dependency structures and implement argument composition for verbs we no longer need non-projective parsers, even if we do not use Chomskyan transformations.

In such a conception, we only need a non-projective element to build up the verb chain and the argument composition. The rest remains projective and is equally suited to be parsed by dependency or constituency. A top rewrite rule might look like

```
( 196) S ->   TOPIC,
              BEFORE_VERB,
              VERB-CHAIN-with-argument-composition,
              AFTER_VERB.
```

The functional roles would have to be attributed later, like in LFG.

One might claim that extraposed Latin adjectives are a poetical device affecting topic and focus, i.e. putting the adjective into focus, which may also render the structure projective again.

The idea presented in this subchapter is only an idea and will need further substantiation. Comments are welcome.

6.3.2 Are Preferred Topics always Semantic Heads ?

Whenever we can assess which of the two nuclei taking part in a connection is more likely to function as a local topic, we know the unmarked direction of the dependency, according to TFA. To illustrate this, let us consider Zwicky's [1985] head criteria constructions from 2.3.2.1 again, only for the (a.) Semantic Argument, juxtaposed to Hudson's [1987]:

(a) Semantic Argument	V+NP	P+NP	NP+VP	DET+N	AUX+VP	COMP+S
Hudson[1987]	V	P	VP	DET	AUX	COMP
Zwicky [1985]	NP	NP	NP	N	VP	S

Table 6: Semantic Head Arguments

We have discussed functional categories in several places in 2.3.2, especially in 2.3.2.5.6. For all functional arguments, it seems very unlikely to conceive of the functional head as a topic and therefore semantic head. It seems highly advisable to follow Tesnière and HPSG here and to take up the functional element into the nucleus. This is also more appropriate at the semantic level because e.g. prepositions and case express functional relations in the same way. They are just relational markers and select the dependency type. For Tesnière, they are the *translatives* for translations (cf. 2.1.4). Functional heads can be found in the following constructions:

- P+NP
- COMP+S
- AUX+VP

If we employ a syntax in Tesnière's intention, the syntactical structure constitutes a functional semantic structure already, which is a good starting point. If we want to use a Deep Case representation the functional structure is only a starting point and we can expect to find the very serious and perhaps unsolvable mapping problems a transformation from f- structure to Deep Case always pose, no matter what theory is employed.

Let us consider the remaining structures:

- V+NP, NP+VP: As discussed in 6.2, also on the semantic level V as the focus and predication over the subject is the rule. In marked structures, expressed by fronting, cleft and pseudo-cleft (cf. 6.3.1) subjects or objects can be focus.
- DET+N: Syntactically, they were one of the most difficult cases. One possibility is to treat them as functional markers, i.e. take them up to the nucleus. But semantically, that is not the end of the road. As far as I can say there is no reason why Montague's PTQ (proper treatment of quantification) should not be applicable to functional dependency structures. In any case functional dependency structures are at best intermediate pre-logical structures, to which PTQ has to be applied at a later stage.

6.4 Functionalism and Semantics in Link Grammar

While Sleator & Temperley [1991] are quite positive about their system, the lack of functionalism makes it unlikely that it is still easy to build useful tectogrammatical of TFA or other logical representations.

> Our current system only uses the linkages it computes for checking if the sentence is grammatical. Are these linkages useful for doing anything else? We have reason to believe that they are. First of all, after seeing several linkages produced by our system, one soon gains an intuition about how the meaning of a sentence is related to the structure of the linkage. This, and the connection to dependency structures [Mel'čuk1988, Hudson 1984] are encouraging signs that the semantics of a sentence can be analyzed by means of the linkages our system produces. (Sleator & Temperley 1991: 56)

I have discussed in chapter 5 that the functional information necessary to build semantic representations is often only contained implicitly, and the functional links between the participants are often very indirect across several linkages. Building semantic representations is still possible, but a considerable effort and a complex mapping between Link linkages and a functional structure or semantic representation is needed.

7. Conclusions

Chapter 2 has introduced dependency as a very intuitive concept, then it elaborates in detail on the differences between dependency and constituency and especially GB as an instance of a constituent grammar theory. I have discussed why dependency is not equivalent to a context-free PSG. The chapter concludes with a summary of the few remaining differences.

It is also claimed that, in order to adequately recognize functionally related structures, any grammar will have to include one of the following devices:

- **Transformations and empty categories**, as in GB. Due to the unnecessary complexity and the problems they create in parsing (backward generation, cf. 2.1.4.3) this device cannot be recommended.
- **Translations** (Tesnière 1959, cf. 2.1.4) or argument composition (cf. 2.3.2.5.5 and 3.3.3.4), in connection with a formalism with free word-order.
- **Parsing of non-projective structures** (cf. 2.4.7), (in addition to free word-order)
- I tentatively suggest in 6.3.1 that parsing for **topic-focus articulation structures** (TFA, cf. 2.2.5.2 and 6.2) may be a fourth alternative, in connection with argument composition for verbs (but non-projectivity may not be necessary, and word-order also becomes more important again)

Link Grammar fails to be functional (cf. chapter 5) because it does not include any of the above.

Chapter 3 shows how to practically write a non-projective dependency parser, both in an imperative and a descriptive language. In both cases, a subroutine (or clause, respectively) for the head which calls a subroutine (clause) for the dependent(s) constitute the core of the program. Because a dependent is a new head, it recursively calls the head subroutine (clause) again. I hope to have shown that writing dependency-based parsers is not more difficult than writing constituent-based ones.

Chapter 4 finally addresses Link Grammar and discusses the differences between 'classical' dependency and Link Grammar. While they are certainly related concepts, the differences are considerable. Like a constituent grammar, Link Grammar is context-free. Unlike any other 'modern' grammar, Link Grammar does not recognize heads.

Chapter 5 offers a first glance at the grammatical coverage of Link Grammar, by means of comparison to a standard English grammar. It is concluded that the coverage is surprisingly high, but some important distinctions, above all the one between *object* and *complement* are not made, and the expected functionalism is largely absent. Much of the linguistic information and many of the functional word-relations are contained in the reported linkage, but a big part of it comes in a very unwieldy format, and functional relations are often indirect across several links with only subscripts identifying special types of sentences. The over one hundred link types are further complicated by a complex subscripting system. Despite its missing functionalism, despite its unwieldy huge set of link types, the incredible speed and surprisingly broad coverage make it a system recommendable for the task at hand in the Zürich ExtrAns project, although a lot of post-processing on the path to truly functional structures is needed.

Chapter 6 mainly introduces topic-focus articulation, which is only one of many relevant topics in semantics. Besides that, it suggest that functional dependency structures (e.g. in the TFA framework) may be a suitable pre-logical intermediate representation.

Besides summarizing or explaining views expressed by researchers, I have made three unusual observations, i.e. observations I have not found elsewhere in the relevant literature.

1. **Minimalist Program is a dependency-based theory**: In 2.3.12 I have discussed that Chomsky's minimalist programme takes up fundamental dependency concepts like free word-order or X' and X" as derived categories. Minimalism approaches dependency theory to such an extent that it can be called a dependency-based theory.
2. **With TFA and argument composition for the verb neither transformations nor non-projectivity are needed**. In 6.3.1 I have suggested that parsing for TFA dependency structures renders one of the major non-projective English structures, long-distance WH movement, projective. If we implement argument composition for verbs to build a single nucleus consisting of the possibly non-projective elements auxiliary verb (AUX), main verb (VP) and verbal particle (in phrasal verbs) all English sentences are projective, as far as I can see. It remains to be checked if this method could solve weak cross-over phenomena in other Indo-European languages.
3. **Covert functional preposition hypothesis.** Based on the parallelism between NPs and sentences, and because prepositions are a functional category much like I(NFL) or C(OMP) dominating the VP it is only logical to suggest that in

addition to the functional category D(ET) a second functional category P(REP) should dominate the NP. This suggestion finally allows to assert the same D-structure to *Peter gives the book to Mary* and *Peter gives Mary a book*. I am surprised that such a hypothesis is not yet a compulsory part of GB theory.

I have been fascinated by dependency and I hope that there will be a possibility for me to be able to continue exploring it.

8. Bibliography

Allerton, David, 1996. "Valency and Valency Grammar". In: [Brown 1996], pp. 359-368.

Anderson, Stephen, 1992. *A-Morphous Morphology*. Cambridge Studies in Linguistics 62. Cambridge: CUP.

Baumgärtner, K., 1970. "Konstituenz und Dependenz. Zur Integration der beiden grammatischen Prinzipien". In: Steger, H. (ed.): *Vorschläge für eine strukturale Grammatik des Deutschen,* pp. 52-77. Darmstadt: Wissenschaftliche Buchgesellschaft.

Borsley, Robert D., 1996. *Modern Phrase Structure Grammar*. Oxford: Blackwell.

Bouchard, Denis, 1995. *The Semantics of Syntax. A Minimalist Approach to Grammar*. Chicago: The University of Chicago Press.

Bresnan, Joan, 1982. "The Passive in Lexical Theory". In: Joan Bresnan (ed.), *The Mental Representation of Grammatical Relations,* pp. 1-59. Cambridge, MA: MIT Press

Brown, Keith, and Jim Miller, 1996. *Concise Encyclopedia of Syntactic Theories.* Oxford: Elsevier.

Bühler, Karl, 1934. *Sprachtheorie.* Jena: Fischer.

Chomsky, Noam ,1965. *Aspects of the Theory of Syntax*. Cambridge, Mass.: MIT Press.

Chomsky, Noam, 1986. *Knowledge of Language. Its Nature, Origin, and Use.* New York: Praeger.

Chomsky, Noam, 1991. “Some notes on economy of derivation and representation”. In: R. Freidin (ed.), *Principles and Parameters in Comparative Grammar*. Cambridge, Mass.: MIT Press.

Chomsky, Noam, 1995. “Bare phrase structure” In: [Webelhuth 1995].

Cook, Vivian and Mark Newson, 1996. *Chomsky’s Universal Grammar, 2nd. Ed.* Oxford: Blackwell.

Covington, Michael, 1990. "Parsing Discontinuous Constituents in Dependency Grammar". *Computational Linguistics 16,* pp. 234-237.

Covington, Michael, 1992. "GB Theory as Dependency Grammar." *Research Report AI1992-03.* Athens, GE: University of Georgia.

Covington, Michael, 1994a. “An Empirically Motivated Reinterpretation of Dependency Grammar”. *Research Report AI1994-01.* Athens, GE: University of Georgia.

Covington, Michael, 1994b. “Gulp 3.1: An Extension of Prolog for Unification-Based Grammars”. *Research report AI-1994-06.* Athens, GE: The University of Georgia.

Cowper, Elisabeth A., 1992. *A Concise Introduction to Syntactic Theory. The Government-Binding Approach.* Chicago: Chicago UP.

Culicover, Peter W., 1997. *Principles and Parameters. An Introduction to Syntactic Theory*. Oxford: OUP.

Dahl, Öster, 1980. "Some Arguments for Higher Nodes in Syntax: A Reply To Hudson's 'Constituency and Dependency' ". *Linguistics 18,* pp. 484-488.

Dalrymple, Mary, Ronald M. Kaplan, John T. Maxwell III and Annie Zaenen, ed. 1995. *Formal Issues in Lexical-Functional Grammar*. Stanford: CSLI.

Engel, Ulrich, 1994. *Syntax der deutschen Gegenwartssprache.* 3. Auflage. Berlin: Schmidt.

Engel, Ulrich, 1996a. “Tesnière missverstanden.” In: [Gréciano 1996].

Engel, Ulrich, 1996b. *Deutsche Grammatik.* 3. Auflage. Heidelberg: Julius Groos

Eroms, Hans-Werner, 1987. "Passiv und Passivfunktionen im Rahmen einer Dependenzgrammatik". In: Centre de Recherche en Linguistique Germanique (ed.): Akten des Kolloquiums über das Passiv im Deutschen, Nizza 1986. *Das Passiv im Deutschen*. Tübingen.

Fasold, Ralph, 1990. *Sociolinguistics of Language*. Oxford: Blackwell.

Feuillet, Jack, 1996. “Les types de fonctions”. In: [Gréciano 1996].

Fillmore, Charles, 1968. "The Case for Case". in: Bach, Emon and R.T. Harms (eds.), *Universals in Linguistic Theory,* pp. 1-88. London.

Finegan, Edward, 1989. *Language. Its structure and use.* 2nd edition. Fort Worth, TX: Harcourt Brace.

Fraser, N., 1989. “Parsing and Dependency Grammar”. In Carston, Robyn (ed.) *UCL Working Papers in Linguistics 1,* pp. 296-338. London: UCL

Fraser, N., 1996. "Dependency Grammar". In [Brown 1996], pp. 71-75.

Gaifman, H., 1965. "Dependency systems and phrase-structure systems". *Information and Control 8,* pp. 304-337.

Gréciano, Gertrud and Helmut Schumacher, ed., 1996. *Lucien Tesnière – Syntaxe structural et opérations mentales. Akten des deutsch-französiaschen*

Kolloquiums anlässlich der 100. Weiderkehr seines Geburtstages. Stasbourg 1993. Tübingen: Niemeyer.

Grover, Claire, John Carroll and Ted Briscoe, 1993. "The Alvey Natural Language Tools grammar (4th release)". *Technical report, Human Communication Research Centre.* Edinburgh: University of Edinburgh

Hajičová, Eva, Hana Skoumalová and Petr Sgall, 1995. "An Automatic Procedure for Topic-Focus Identification". In: *Computational Linguistics 21*.

Hays, David G. 1964. "Dependency theory: A formalism and some observations." *Language*, 40, pp. 511-525.

Helbig, Gerhard (ed.), 1988. *Linguistische Studien*. Arbeitsberichte 180. Berlin: Akademie der Wissenschaften der DDR. Zentralinstitut für Sprachwissenschaft.

Helbig, Gerhard, 1992. *Probleme der Valenz- und Kasustheorie*. Konzepte der Sprach- und Literaturwissenschaft. Tübingen: Niemeyer.

Helbig, Gerhard, 1996. "Zur Rezeption und Weiterentwicklung des Tesnièreschen Valenzkonzepts". In: [Gréciano 1996].

Hellwig, P., 1986. "Dependency Unification Grammar." In: *Proceedings, 11th International Conference on Computational Linguistics (COLING 1986)*, pp. 195-198. Bonn: University of Bonn.

Horrocks, Geoffrey, 1987. *Generative Grammar*. Harlow: Longman.

Hudson, Richard, 1980a. "Constituency and Dependency". *Linguistics 18*, pp. 179-198.

Hudson, Richard, 1980b. "A Second Attack on Constituency". *Linguistics 18*, pp. 489-504.

Hudson, Richard, 1984. *Word Grammar*. Oxford: Basil Blackwell.

Hudson, Richard, 1987. "Zwicky on Heads". *Journal of Linguistics 23*, pp. 109-132.

Hudson, Richard, 1990. *English Word Grammar*. Oxford: Basil Blackwell.

Hudson, Richard, 1996. "Word Grammar". In: [Brown 1996], pp. 368-372.

Hudson, Richard, 1997. *Functional Control with and without Structure-Sharing*. Manuscript.

Järvinen, Timo and Pasi Tapanainen, 1997. *A Dependency Parser for English*. Department of Linguistics Technical Report TR-1. Helsinki: University of Helsinki.

Järvinen, Timo and Pasi Tapanainen, 1998. “Towards an Implementable Dependency Grammar”. Accepted for COLING-ACL ’98 Workshop: *Processing of Dependency-Based Grammars*.

Jung, Wha-Young, 1995. *Syntaktische Relationen im Rahmen der Dependenzgrammatik*. Hamburg: Buske.

Karlsson, Fred, 1990. “Constraint grammar as a framework for parsing running text”. In: Hans Karlgren, ed., *Papers presented to the 13th International Conference on Computational Linguistics Vol. 3*, pp. 168-173. Helsinki.

Knight, K., 1989. "Unification: A multidisciplinary survey." *ACM Computing Surveys* 21(1), pp. 105-113.

Korhonen, Jarmo, 1977. *Studien zur Dependenz, Valenz und Satzmodell, Teil 1. Theorie und Beschreibung der deutschen Gegenwartssprache. Dokumentation, kritische Besprechung, Vorschläge*. Bern: Peter Lang.

Lingsoft Oy, 1994. “Gertwol. Questionnaire for Morpholympics 1994”. *LDV-Forum* 11(1), pp. 17-29.

Matthews, P.H., 1981. *Syntax*. Cambridge: CUP

Mel’čuk, Igor A., 1988. *Dependency Syntax: Theory and Practice*. New York: State University of New York Press.

Mollá Aliod, Diego, 1997. "Link Grammar vs. Alvey". Internal Research Report. University of Zürich.

Peregrin, Jaroslav, 1996. “Topic and focus in a formal framework”. In: *Discourse and Meaning: Papers in Honour of Eva Hajičová*, ed. Barbara H. Partee and Petr Sgall. Amsterdam: John Benjamins.

Perlmutter, David M. (ed.), 1983. *Studies in Relational Grammar I*. Chicago: Chicago University Press.

Pollard, Carl, and Ivan A. Sag, 1994. *Head-Driven Phrase Structure Grammar*. Chicago: Chicago UP.

Quirk, Randolph, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik, 1985. *A Comprehensive Grammar of the English Language*. London: Longman.

Radford, Andrew, 1988. *Transformational Grammar. A First Course*. Cambridge: CUP.

Radford, Andrew, 1997. *Syntactic Theory and the Structure of English. A Minimalist Approach*. Cambridge: CUP.

Rambow, Owen, and Aravind Joshi, 1994. “A Formal Look at Dependency Grammars and Phrase-Structure Grammars, with Special Consideration of Word-Order Phenomena”. In: Wanner, Leo, ed., 1994. *Current Issues in Meaning-Text Theory*. London: Pinter.

Russell, Bertrand. 1905. "On denoting". *Mind 14*: 479 - 93.

Schneider, Gerold, 1996. "An LFG Grammar for German, Danish and English." Seminar Paper at the Universities of Zurich and Copenhagen.

Schneider, Gerold, 1997. "Lexikalische Integrität in implementations–orientierten Grammatiken." Seminar Paper at Zurich University.

Schneider, Gerold, and Martin Volk, 1998. “Adding Manual Constraints and Lexical Look-Up to a Brill-Tagger for German”. To appear in: *Proceedings of the ESSLLI ’98 Workshop: Recent Developments in Corpus Annotation*. Saarbrücken.

Schubert, Klaus, 1988. "Zu einer sprachübergreifenden Definition der Valenz". In [Helbig 1988].

Sells, Peter, 1987. *Introduction to Syntactic Theories*. Stanford: CSLI.

Shieber, S. M., 1986. "An introduction to unification-based approaches to grammar." *CSLI Lecture Notes,* No. 4. Stanford, CA: Stanford University.

Sgall, Petr, Eva Hajičová and Jarmilla Panevová, 1986. *The Meaning of the Sentence in Its Semantic and Pragmatic Aspects*. Dordrecht: Reidel.

Sleator, Daniel, and Davy Temperley, 1991. "Parsing English with a Link Grammar." Technical report CMU-CS-91-196, Carnegie Mellon University, School of Computer Science.

Sleator, Daniel, and Davy Temperley, 1993. "Parsing English with a Link Grammar." Proceedings of *IWPT ’93*.

Sleator, Daniel, and Davy Temperley, 1998a. “Guide to Links”. Provided together with the system or available online at *http://www.link.cs.cmu.edu/link/index.html*

Sleator, Daniel, and Davy Temperley, 1998b. “An Introduction to the Link Grammar Parser”. Provided together with the system or available at *http://www.link.cs.cmu.edu/link/dict/introduction.html*

Sleator, Daniel, and Davy Temperley, 1998c. “Link Grammar Version 3.0 Improvements”. Provided together with the system or available at *http://www.link.cs.cmu.edu/link/dict/improvements.html*

Steimann, Friedrich, and Christoph Brzoska, 1995. "Dependency Unification Grammar for Prolog". *Computational Linguistics* 21, pp. 95-103.

Storrer, Angelika, 1996. “Wie notwendig sind obligatorische Valenzstellen ? – Faktoren der Weglassbarkeit von Valenzstellen im Text”. In: [Gréciano 1996].

Strawson, Peter, 1952. *Introduction to Logical Theory*. London: Methuen

Sutcliffe, Richard, Tom Brehony and Annette McElligott, 1995. "The Grammatical Analysis of Technical Texts Using a Link Parser." Proceedings of *PACLING-II.*

Sutcliffe, Richard, Heinz-Detlev Koch and Annette McElligott, 1997. *Industrial Parsing of Software Manuals.* Amsterdam: Rodopi.

Tapanainen, Pasi and Timo Järvinen, 1997. “A non-projective dependency parser”. In: *Proceedings of the 5th Conference on Applied Natural Language Processing.* Washington, DC: Association for Computational Linguistics.

Tarvainen, Kalevi, 1981. *Einführung in die Dependenzgrammatik*. Reihe Germanistische Linguistik 35. Tübingen: Niemeyer.

Tesnière, Lucien, 1959. *Eléments de syntaxe structurale.* Paris: Librairie Klincksieck

Valkonen, K., H. Jäppinen, A. Lehtola and M. Ylilammi, 1987. “Declarative Model for Dependency Parsing - A View into Blackboard Methodology”. In: *Papers Presented to The Association of Computational Linguistics. Copenhagen*, pp. 218-225.

Vikner, Sten, 1995. *Verb Movement and Expletive Subject in the Germanic Languages.* Oxford: OUP.

Volk, Martin, and Gerold Schneider, 1998. “Comparing a statistical and a rule-based tagger for German”. To appear in: *Proceedings of Konvens-98.* Bonn.

Webelhuth, Gert (ed.), 1995. *Government and Binding Theory and the Minimalist Program*. Oxford: Blackwell.

Weber, Heinz J., 1996. “Translation und Rekursivität bei Lucien Tesnière”. In: [Gréciano 1996].

Weber, Heinz J., 1997. *Dependenzgrammatik.* Ein interaktives Arbeitsbuch. 2. Auflage. Tübingen: Gunter Narr Verlag.

Wittgenstein, Ludwig, 1918. *Tractatus logico-philosophicus*. in: *Werkausgabe Band 1*. 1990. Suhrkamp Taschenbuch Wissenschaft. Frankfurt a. M.: Suhrkamp.

Wotjak, Gerd, 1996. “Actants und circonstants. Tesnières Pionierleistung in semantischer Sicht: Zu Funktoren, Argumenten und Modifikatoren”. In: [Gréciano 1996].

Zwicky, Arnold, 1985. “Heads”. *Journal of Linguistics 21*, pp. 1-30.

Zwicky, Arnold, 1986. "Agreement Features: Layers or Tags". *Ohio State University Working Papers in Linguistics* 32, pp. 146-8