Chapter 3 💬

The dependency grammar

The Stanford dependency analysis of a given text constitutes the input for the algorithm developed in the current work. It provides the foundation to build the syntactic backbone ~~used~~ adopted here. This chapter offers an overview of the grammar and the parser developed at ~~the~~ Stanford university. In the last part of the chapter ~~is discussed~~ the cross theoretical connection between ~~the~~ dependency and systemic functional grammars.

3.1 A ~~briefing on~~ the theory of dependency grammar

Traditionally, Latin language tended to be analysed with the *dependency model* based on theories of what was called *government*. It explains the syntagmatic relations of the subtype Firth called *colligation* (i.e. relations between grammatical categories). Government is a way of explaining the rich inflections of language (such as Latin) in terms of how particular words govern, that is to say, determine, the inflection of other words. (McDonald 2008: p.66)

Tensiere (2015) explains how government works by giving example of Latin text analysis where the inflections immediately give ~~aloto of~~ information about relations between different words. For example the verb agree in person and number with its subject. From this point of view the verb ~~governs~~ the subject. The verb also governs complements and adjuncts which can be seen as relations of dependency between a governing element or controller and a governed element or dependant. It is important to note that Tensiere analysed syntax at the level of clause where he identified a verb node as the main controller.

~~Contrary~~ to Latin, languages like English or Chinese, where there is little or most of the time none of the inflectional marking ~~to~~ identify dependency relations, are ~~much harder to analyse~~ in terms of such relations. This was a motivation for Tensiere to reinterpret dependency relation in semantic terms rather than inflectional marking. As McDonald (2008) points out, this can be regarded as extending syntagmatic relations of the clause to include what Firth was calling *collocation* relations (i.e. links between lexical items). Tensiere framed his theory in terms of syntagmatic relations as expressing a model of experience. He compares the verbal node of the clause to a complete ~~little~~ drama. Like a drama, it obligatorily consists of an action, most often actors and features of settings. Expressed in terms of syntactic structure, the action, actors and settings become the verb, participants and circumstances (Tensiere 2015).

Further, Tensiere explains *categories of language* as *categories of thought.* The human mind shapes the world in its own measure by organising experience into a systematic framework of ideas and beliefs called categories of though. Likewise, ~~the~~ language shapes thought in its own measure by organising it into a systematic framework of grammatical categories (Tensiere 2015). He stresses though, that the latter ~~ones~~ can vary considerably from language to language and that the analysis of syntactic relations ~~shall~~ be carried ~~on~~ not in terms of grammatical categories but rather in terms of functions. He explain through an example that analysis in terms of nouns and verbs i.e. grammatical categories, tells nothing about the tie that links the words, whereas if we turn to notions such as subject and complement it ~~all of the sudden~~ becomes clear: the connections are established, the lifeless words become a living organism and the sentence take on a meaning (Tensiere 2015).

3.2   Stanford dependency grammar (and parser)

~~Dominant~~ Chomsky (1981) ~~theories~~ define grammatical relations as configurations of *phrase structure* representations, which ~~is~~ nesting of multi word constituents. Other theories such as Lexical-Functional Grammar reject the adequacy of such an approach (Bresnan 2010) and advocate a functional representation for syntax.

When motivating its stance, Marneffe et al. (2006) insists on practical rather than theoretical concerns proposing that structural configurations be defined as grammatical roles (to be read as grammatical functions)(Marneffe et al. 2006). For example she insists that~~,~~ information about functional dependencies between words grants direct access to the predicate-argument relations which are not readily available ~~form~~ the phrase structure parses and can be used off the shelf for real world applications. She

avoids going into theoretical debates and focuses on the suitability of the grammar for parsing within the context of relation extraction, machine translation, question answering and other tasks.

The functional dependency descriptions is precisely the aspect which makes possible the link between the Stanford Dependency Grammar and Systemic Functional Structures targeted in the current thesis.

The design of Stanford dependency set (Marneffe et al. 2006; Marneffe & Manning 2008a; Marneffe et al. 2014; Natalia Silveira et al. 2014) bears a strong intellectual debt to the framework of Lexical Functional Grammars (Bresnan 2010). Marneffe et al. (2006) started designing the relation typology from GR (Carroll et al. 1999) and PARK (King & Crouch 2003) schemes following a LFG style and according to the principles described in Generalization 3.2.1.

Generalization 3.2.1 (Design principles for Stanford dependency set)**.**

1. Everything is represented uniformly and some binary relation between two sentence words.

2. Relations should be semantically contentful and useful to applications.

3. Where possible, relations should use notions of traditional grammar (Quirk et al. 1985) for easier comprehension by users.

4. The representation should be spartan rather than overwhelming with linguistic details.

When proposing the Stanford dependency, Marneffe et al. (2006) inherits many relations from Lexical Functional Grammars (Bresnan 2010) and departs from the sets described by Carroll et al. (1999) and King & Crouch (2003). She arranges the grammatical relations into a hierarchy rooted in a generic relation *dependent*. This is then classified into a more fine-grained set of relations between a head and its dependent.

3.3 Stanford Parser: collapsed-cc output

The Stanford Dependency Parser is capable of generating several types of dependency representations. The most convenient and informative version is Collapsed-CC-processed. This structure is created after the initial parse is ready and constitutes a simplified and more intuitive representation of the dependency parse. The collapsed

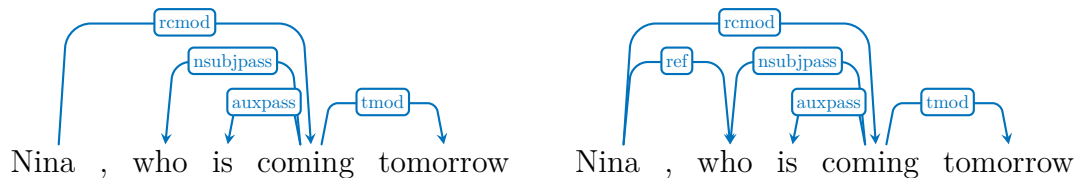Fig. 3.2 Collapsed preposition dependency

Fig. 3.4 Collapsed preposition dependency

form concerns prepositions, conjunctions and relative clause referents. Dependencies involving prepositions and conjunctions are transformed from basic form in Figure 3.1 to the form where preposition are embedded into the edge relation, as in Figure 3.2. Similar transformation is done for conjunctions.

Besides collapsing prepositions and conjunctions the DG is further processed to introduce more relations (i.e. connections) even if they break the tree structure. For example the *ref* relation does not appear in the basic dependency structure in figure 3.3 but it appears in the processed dependency structure forming a cycle with *rcmod* and *subjpass* relations.

Relations like *ref* introduce cycles but add valuable information useful in various stages of further processing. However, ensuring a tree structure in important for the CG creation stage because it is based on a top down traversal. This is taken care of in the preprocessing stage described in the next stage.

3.4 Penn part-of-speech tag set

Stanford dependency parser starts creation of the parse structure process from the list of tokens annotated with Penn part-of-speech tags. Embedded into the dependency graph, these tags are the part of the syntactic context from which SFG constituency graph is built.

The Penn tagset was developed to annotate the Penn Treebank corpora (Marcus et al. 1993). It is a large, richly articulated tagset that provides distinct coding for classes of words that have distinct grammatical behaviour.

It is based on the Brown Corpus tagset (Kucera & Francis 1968) but differs in several ways from it. First, the authors reduced the lexical and syntactic redundancy. In Brown corpus there are many unique pos tags to a lexical item. In Penn tagset the intention is to reduce this phenomena to minimum. Also distinctions that are recoverable from lexical variation of the same word such as verb or adjective forms or distinctions recoverable from syntactic structure were reduced to a single tag.

Secondly the Penn Corpus takes into consideration the syntactic context. Thus the Penn tags, to a degree, encode the syntactic functions when possible. For example *one* is tagged as NN (singular common noun) when it is the head of the noun phrases rather than CH (cardinal number).

Thirdly Penn POS set allows multiple tags per word, meaning that either it cannot be decided or the annotators may be unsure of which one to choose.

There are 36 main POS and 12 other tags in Pen tagset. A detailed description of the schema, the design and principles and annotation guidelines are described in (Santorini 1990).

3.5   Cross theoretical bridge from DG to SFL

The concept of dependency between pairs of words is long acknowledged in linguistic communities. In traditional terms dependencies are treated as *hypotactic expansions* (see Definition 2.3.11) of word classes (or parts of speech) where the expanded word acts as *heads* and expanding ones as *dependent* establishing parent-daughter structural relations illustrated in Figure 3.5a.

In SFL community the concept of dependency is less salient than the foundational role it plays in the Dependency Grammars. Dependencies are regarded as orthogonal relations between sibling elements of a unit (Figure 3.5b) and link the *heads* to their *modifies* in by Hallidayan *logical structure*(Halliday & Matthiessen 2013).

Figure 3.5 illustrates side by side the parent-daughter and sibling dependency relations. In Figure 3.5a dependency are the only relations between the units of structure whereas in Figure 3.5b there are multiple levels (ranks) of units and the dependency relations are relevant only between siblings at the same level within the structure of an unit. SFL regards dependency relations holding only between elements of a unit whereas the relations that connect the units of lower and higher rank are *constituency relations*. Yet when we look at the two structures they resemble in a way each other and next I show how.
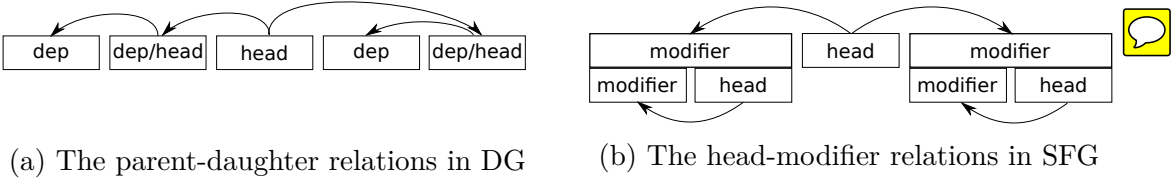
(a) The parent-daughter relations in DG

(b) The head-modifier relations in SFG

Fig. 3.5 ~~The~~ dependency relations in DG and SFG

In a nutshell, the parent-daughter dependency relations in dependency grammar unpack into multiple function in systemic functional grammar ~~and~~ specifically it is the head-modifier indirect relation, unit-element componence relation and the head of unit a "representativeness" function.

This difference implies that, when translated to a constituency unit (described in Section 6.4), the dependency unit, stands for both a unit and that unit's head element. In other words a DG node corresponds to two functional elements at different rank scales. For example the root verb in dependency graph corresponds to the clause node and the lexical item which fills the Main Verb of the clause. By analogy, the head noun of a Nominal Group anchors the entire unit (as a group) and fills the head element of the group

text	some	very	small	wooden	ones
units	**Nominal Group**				
elements	*Quantifying Determiner*	*Modifier*		*Modifier*	*Head*
units		**Quality Group**			
elements		*Degree Temperer*	*Apex*		

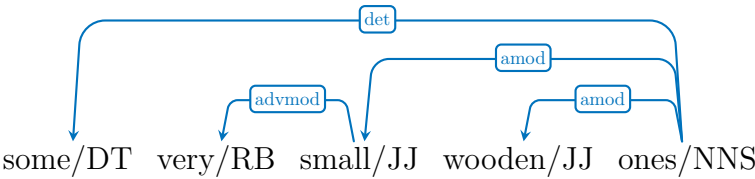Table 3.1 SF analysis of Example 3 (reproduced from Table 2.4)

Fig. 3.6 Dependency analysis for Table 3.1

Figure 3.6 and ~~the~~ Table 3.1 represent the analysis of a nominal group from Example 3 ("some small very small wooden ones") in Cardiff grammar and Stanford dependency grammar ~~exhibiting a~~ contrast ~~of~~ the two structures. Consider the dependency relation "det" a link between the noun "ones" and the determiner "some". When translated into SF variant the dependency relation stands within Nominal Group between the

Head element (filled by word "ones") and the Quantifying Determiner element (filled by the word "some"). By definition all elements in a unit are equal in the structure so the Head and Quantifying Determiner are siblings. So the items (words) filling those elements are sibling. How ~~is~~ then the dependency relation established?
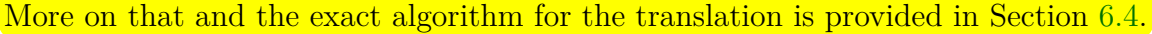
In SFL there is the concept of Head and Modifier. There is no direct relationship between them ~~but it is said that what the Modifier modifies is the Head. The relation is not a direct one,~~ the Modifier and Head stand for two different kinds of meaning and what the Modifier modifies is not the Head per se but the referent denoted by the head (and thus construed by the entire unit). It is precisely this modification of the head that is called a (sibling) dependency relation and is seldom mentioned in SFL literature because it is considered implicit and recoverable from the SF constituency structure.

The Head also is the element that anchors the entire unit and plays a constitutive role. In this sense the word "ones" realizes not only the Head function (sided with Determiner "some") but also anchors the entire unit. The relation between the group and it's elements is one of *componence* (Definition 2.3.4) described in Section 2.3.4. Yet in the role of unit anchor we cannot say that there is a componence relation between "one" and "some" because it is merely a proxy ~~to~~ the referent rather than the entire unit. So in this role "one" can be said to be standing in a parent-daughter dependency relation to "some" incorporating the filling and componence relations.

I ~~just showed~~ how the dependency relation in dependency structure (Figure 3.5a) can be unpacked into two relations in systemic functional structure (Figure 3.5b): sibling dependency considered an indirect relation between Head and Modifier (Logical Metafunction) and parent-daughter dependency between unit anchor and the compounding elements, relation which resembles unit componence but is not.

Lets look at a second example of two relations "advmod" from "small" to "very" and "amod" from "ones" to "small". The interesting case here is the item "small" which is the Head of the Quality Group~~, it~~ anchors the meaning of the whole group and the Quality group fills the Modifier element within the Nominal group. What is not covered in previous example is that the Apex "small" not only is a representative of the entire group but it also is a *representative filler* of the Modifier element within Nominal Group. Using the similar translation mechanism as above, this means that, the incoming dependency needs to be unpacked into three levels: the element within the current group (Modifier), the unit class that element is filling (Quality Group) and finally the head of the filler group (Apex). In fact, to be absolutely correct there is one

more level. The elements of a unit are expounded by lexical items, so fourth relation to unpack is the expounding of the Apex by "small" word.

In this section I laid the theoretical principle for transforming the dependency structure into systemic functional structure. In practice to achieve this level of unpacking the algorithm requires a bottom up and a top down contextualization in terms of elements of structure within a unit and realised sequence of textual units. This implied that the unpacking needs two traversals, a bottom-up and a top one. More on that and the exact algorithm for the translation is provided in Section 6.4.

Next follows the chapter on Governance and Binding Theory, needed to account for the unrealised, covert (Null) Elements in the syntactic structure. It is also an opportunity to perform a similar theoretical translation exercise (as in this section) from one theory of grammar into another.

Chapter 4

~~Governance~~ and binding theory

Government and Binding Theory (GBT) is a theory of syntax based on phrase structure grammar ~~and is~~ a part of transformational grammar developed by Noam Chomsky (1981). It explains how some constituents can *move* from one place to another, where are the places of *non-overt constituents* and what constituents do they refer to i.e. what are their *antecedents*.

Knowing that some constituents are not realised in certain places with specific syntactic functions greatly benefits semantic analysis process. GBT is important for Transitivity parsing which is semantic in nature but still systematised at the level of abstraction that enables capturing the grammatical variation.

GBT approach ~~to~~ explain grammatical phenomena using *phrase structures* (PS) is more distant from SFG than the ~~main~~ approach taken by ~~the~~ dependency grammar. Section 4.2 briefly introduces the theoretical context of GBT and then formulates the principles and generalisations relevant for current work. Then Section 4.3 translates the introduced principles and generalisations into DG rules and patterns. To lay the ground for the two sections, I first place GBT into the context of transformational grammar.
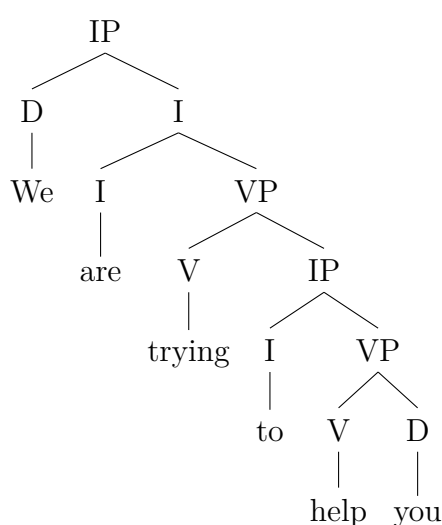
4.1 On Transformational grammar and parse trees

The notion of structure in generative grammar ~~means~~ the way words are combined together to form phrases and sentences. *Merging* is the technical term for the operation of ~~combining~~ two words together into a phrase. In this operation one word will always be more prominent called the *head* of the phrase ~~and~~ the resulting combination is a new constituent ~~which is~~ a *projection* of the head.

These combinations of words and projections can be represented using the *labeled bracketing notation* where the labels denote constituent category. The bracketed notation is a representation equivalent to a hierarchical tree of constituent parts called *parse tree* (aka *syntactic tree*, *phrase structure*, *derivation tree*). The parse tree represents the syntactic structure of a string according to some grammar. The equivalence between a bracketed notation and parse tree is exemplified in the following two representations.

(27) $\left[_{IP}\left[_{D}We\right]\left[_{\bar{I}}\left[_{I}are\right]\left[_{VP}\left[_{V}trying\right]\left[_{IP}\left[_{I}to\right]\left[_{VP}\left[_{V}help\right]\left[_{D}you\right]\right]\right]\right]\right]\right]$

```
                IP
              /    \
            D        I
            |       /  \
           We      I      VP
                   |     /   \
                  are   V      IP
                        |     /   \
                     trying  I      VP
                             |     /   \
                            to    V      D
                                  |      |
                                 help   you
```

Transformational grammar (TG) or transformational-generative grammar (TGG) is, in the study of linguistics, part of the theory of generative grammar, especially of naturally evolved languages, that considers grammar to be a system of rules that generate exactly those combinations of words which form grammatical sentences in a given language. TG involves the use of defined operations called transformations to produce new sentences from existing ones.

Chomsky developed a formal theory of grammar where transformations manipulated not just the surface strings, but the parse tree associated with them, making transformational grammar a system of tree automata (Stockwell et al. 1973).

A transformational-generative (or simply transformational) grammar thus involved two types of productive rules: phrase structure rules, such as "S -> NP VP" (meaning that a sentence may consist of a noun phrase followed by a verb phrase) etc., which could be used to generate grammatical sentences with associated parse trees (phrase markers, or P markers); and transformational rules, such as rules for converting statements to questions or active to passive voice, which acted on the phrase markers

to produce further grammatically correct sentences (Bach 1966: 59-66). This notion of transformation proved adequate for subsequent versions including the "extended", "revised extended" and Government-Binding (GB) versions of generative grammar, but may no longer be sufficient for the current minimalist grammar, in that merge may require a formal definition that goes beyond the tree manipulation. For the purpose of the current work, however, the GBT describes clearly the mechanisms for identification of *null elements* therefore I find it most suitable and preferred to minimalist program.

4.2   On Null Elements (empty categories)

In linguistics, in the study of syntax, an *empty category* is a nominal element that does not have any phonological content and is therefore unpronounced. Empty categories may also be referred to as *covert nouns*, in contrast to overt nouns which are pronounced. The phenomenon was named by Noam Chomsky in his 1981 LGB framework. Some empty categories are governed by the empty category principle. When representing empty categories in trees, linguists use a null symbol to depict the idea that there is a mental category at the level being represented, even if the word(s) are being left out of overt speech. There are four main types of empty categories: NP-trace, Wh-trace, PRO, and pro.

There are four main types of empty categories: *NP-trace*, *Wh-trace*, *PRO*, and *pro*. The types are differentiated by ~~their~~ two binding features: the anaphoric feature [a] and the pronominal feature [p]. The four possible combinations of plus or minus values for these features yield ~~the~~ four types of empty categories.

[a]	[p]	Symbol	Name of empty category	Corresponding overt noun type
-	-	t	Wh-trace	R-expression
-	+	pro	little Pro	pronoun
+	-	t	NP-trace	anaphor
+	+	PRO	big Pro	none

Table 4.1 Four types of empty categories according to their binding features

In ~~the~~ Table 4.1, [+a] refers to the anaphoric feature, meaning that the particular element must be bound within its governing category. [+p] refers to the pronominal feature which shows that the empty category is taking the place of an overt pronoun.

4.2.1   PRO Subjects

PRO stands for the non-overt NP ~~that is the~~ subject in non-finite (complement, adjunct or subject) clause and is accounted by the *control theory* (CT).

Definition 4.2.1 (Control)**.** Control is a term used to refer to a relation of referential dependency between an unexpressed subject (the control element) and an expressed or unexpressed constituent (controller). The referential properties of the controlled element are determined by those of the controller. (Brensan 1982)

Control can be *optional* or *obligatory*. While *Obligatory control* has a single interpretation, that of PRO being bound to its controller, ~~the~~ *optional control* allows for two interpretations: *bound* or *free*. In 28 the PRO is controlled, thus bound, by the subject "John" of the matrix clause (i.e. higher clause) whereas in 29 it is an arbitrary interpretation where PRO ~~is~~ refers to "oneself" or "himself". In 30 and 31 PRO must be controlled by the subject of the higher clause and does not allow for the arbitrary interpretation.

(28)   John asked how [PRO to behave himself/oneself.]

(29)   John and Bill discussed [PRO behaving oneself/themselves in public.]

(30)   John tried [PRO to behave himself/*oneself.]

(31)   John told Mary [PRO to behave herself/*himself/*oneself.]

Sometimes the controller is the subject (~~like~~ in Examples 28, 29, 30) and sometimes it is the object (Example 31) of the higher clause. Haegeman (1991: p. 278) proposes that there are verbs of *subject* and of *object control.* However if there is ~~an~~ Complement in the higher clause, as a rule of thumb, it is likely that it is the controller of the PRO in lower clause. Three-role cognition verbs like in Example 31 take two complements where one of them is the phenomenon and can be a nominal group or a non-finite clause with PRO subject controlled by the object in higher clause.

The following set of generalizations from Haegeman (1991) are instrumental in identifying places where PRO constituent occur~~s~~ and its corresponding binding element.

Generalization 4.2.1. Each clause has a subject. If a clause doesn't have an overt subject then it is covert (non-overt) represented as PRO. (Haegeman 1991: p. 263)

Generalization 4.2.2. A PRO subject can be bound, i.e. it takes a specific referent or can be arbitrary (equivalent to pronoun "one") (Haegeman 1991: p. 263) In case of obligatory control, PRO subject is bound to a NP and must be c-commanded by its controller (Haegeman 1991: p. 278).

Generalization **4.2.3.** PRO must be in ungoverned position. It means that (a) PRO does not occur in object position (b) PRO cannot be subject of a finite clause.

Generalization 4.2.4. PRO does not occur in the non-finite clauses introduced by *if* and *for* complementizers but it can occur in those introduced by *whether*.

Examples 32 and 33 illustrate generalization 4.2.4

(32)   John doesn't know [whether [PRO to leave.]

(33)   * John doesn't know [if PRO to leave.]

Generalization 4.2.5. PRO can be subject of complement, subject and adjunct clauses. (Haegeman 1991: p. 278)

Generalization 4.2.6. When PRO is the subject of a declarative complement clause it must be controlled by an NP i.e. arbitrary interpretation is excluded. (Haegeman 1991: p. 280)

Generalization 4.2.7. The object of active clause becomes subject when it is passivized and also controls the PRO element in complement clause. (Haegeman 1991: p. 281)

Generalization 4.2.8. PRO is obligatorily controlled in adjunct clauses that are not introduced by a marker. (Haegeman 1991: p. 283)

Adjuncts (clauses or phrases) often are introduced via preposition and these are some of rare cases of adjunct clauses free of preposition. Examples 34 and 35 illustrate marker-free adjunct clauses.

(34)   John hired Mary [PRO to fire Bill.]

(35)   John abandoned the investigation [PRO to save money.]

Generalization 4.2.9. PRO in a subject clause is optionally controlled thus by default it takes arbitrary interpretation.

A default assumption is to assign arbitrary "one" interpretation to each PRO subject in subject clauses. However there are cases when it may be bound (resolved) to a pronominal NP in the complement of the higher clause. Binding element can be either the entire complement or a *pronominal* part of it like the qualifier or the possessor. Example 36 illustrates that PRO has only arbitrary interpretation since cannot be bound to the complement "health". Moreover PRO can also be bound to (a) the possessive element of higher clause - example 37, (b) the complement of the higher clause - example 38 and (c) either the possessives in lower or higher clause, example 39.

(36)   PRO$_i$ smoking is bad for the health$_j$.

(37)   PRO$_i$ smoking is bad for your$_i$ health$_j$.

(38)   PRO$_i$ smoking is bad for you$_i$.

(39)   PRO$_i$ lying to your$_i$ friends decreases your$_i$ trustworthiness$_j$.

4.2.2   NP-traces

~~The~~ NP-movement in GB theory is used to explain *passivization, subject movement* (in interrogatives) and *raising*. The raising phenomena (Definition 4.2.2) is the one that is of interest for us here as it is the one involving an empty constituent.

Definition 4.2.2 (NP-raising)**.** NP-raising is the NP-movement of a subject of a lower clause into subject position of a higher clause.

Consider examples 40 to 43. There are two cases of expletives (40 and 42) and their non-expletive counterparts (41 and 43~~-~~) where the subject of the lower clause is moved to the subject position of the matrix clause by replacing the expletive. The movement of NPs ~~leaves~~ traces which here are marked as *t*. This phenomena is called *raising* (Definition 4.2.2) or as postal calls it *subject-to-subject raising* Postal (1974).

(40)   It was believed [Poirot to have destroyed the evidence.]

(41)   Poirot$_i$ was believed [t$_i$ to have destroyed the evidence.]

(42)   It seems [that Poirot has destroyed the evidence.]

(43)   Poirot$_i$ seems [t$_i$ to have destroyed the evidence.]

The subjects "It" and "Poirot" in none of the examples 40-43 receive a semantic role from the main clause. "It" is an expletive and never receives a semantic role while "Poirot" in 41 and 43 takes *Agent* role from "destroy" and is not the *Cognizant* neither of "believe" nor of "seem". So verbs "believe" and "seem" do not theta mark (assign semantic roles) their subjects in none of the examples.

Table 4.2 represents the semantic role distribution for verb senses in examples above. Example 41 is a passive clause with an embedded complement clause. The subjects and complements switch places in passive clauses and so do the semantic roles. That would mean that Cognizant role goes is to be assigned to embedded/complement clause. However Phenomena is the only semantic role that can be filled by a clause, all other roles take nominal, prepositional or adjectival groups.

In example 43 the verb "seem" assigns roles only to complements and as the embedded clauses can take only the phenomenon role, the cognizant is left unassigned

Verb	Process type	canonical distribution of semantic roles
Believe	Cognition	Cognizant + V + Phenomenon
Seem	Cognition	It + V + Cognizant + Phenomenon
Destroy	Action	Agent + V + Affected

Table 4.2 Semantic role distribution for verbs "believe" and "seem"

and so does not assign any thematic role to the subject which then can be filled either by an expletive or a moved NP.

Definition 4.2.3 (Trace, Antecedent and Chain)**.** An empty category which encodes the base position of a moved constituent is referred to as *trace*. The moved constituent is called *antecedent* of a trace. Both the trace(s) and the antecedent form a *chain*.

Raising is very similar to obligatory subject control with a difference in thematic role distribution. In the case of subject control, both the PRO element and it's binder (the subject of higher clause) receive thematic roles in both clauses. However in the case of raising, the NP is moved and it leaves a trace which is theta marked but not to the antecedent. This is expressed in generalization 4.2.10.

Generalization 4.2.10. The landing site for a moved NP is an empty A-position. The chain formed by a NP-movement is assigned only one theta role and it is assigned on the foot of the chain, i.e. the lowest trace. (Haegeman 1991: p. 314)

So, in case of raising, the landing sites for a moved NP are empty subject positions or the ones for expletives. As a result of movement, these positions are filled or the expletives replaced.

4.2.3   Wh-traces

Wh-movement is involved in formation of Wh-interrogatives and in formulation of relative clauses (Haegeman 1991: p. 423). We are interested in both cases as both of them leave traces of empty elements that are relevant for transitivity analysis.

(44)   [What] will Poirot eat?

(45)   [Which detective] will Lord Emsworth invite?

(46)   [Whose pigs] must Wooster feed?

(47)   [When] will detective arrive at the castle?

(48)   [In which folder] does Margaret keep the letter?

(49)   [How] will Jeeves feed the pigs?

(50)   [How big] will the reward be?

Haegeman (1991: p. 375) treats each Wh-element as the head of the Wh-phrase. I do not share this perspective. In some cases they function as heads but there are other cases when they act as determiners, possessors or adjectival modifiers. This issue is extensively discussed by Abney (1987); Quirk et al. (1985); Halliday & Matthiessen (2013).

For clarity purposes I define the Wh-group and Wh-element in 4.2.4. Note that Wh-group is not a new unit class in the grammar but a constituent feature that can span across three unit classes.

Definition 4.2.4 (Wh-group, Wh-element)**.** *Wh-group* is a nominal, prepositional or adverbial group that contains Wh-element either as head or as modifier. The *Wh-element* is a unit element filled by any of the following words or their morphological derivations (by adding suffixes *-ever*, *-soever*): *who, whom, what, why, where, when, how.*

In GBT ~~the~~ *case* occupies an important place in the grammar. Verbs dictate the case of their arguments, a property called case marking. ~~It is established that the~~ Subjects are marked with Nominative case while the Complements of the verb are marked with Accusative case.

In English however case system is very rudimentary as compared to other languages. Hence, *who, whom* and their derivatives *whoever* and *whomever* are the only Wh-elements with overt case differentiation. The other Wh-elements *what, when, where, how* do not change their form based on the case.

~~The~~ example 51 shows that the Accusative (*whom*) is disallowed when its trace is in Subject position ~~requiring~~ Nominative (*who*). In example 52 the reverse holds and the Nominative form is disallowed as the Wh-element moved from Complement position ~~requiring~~ Accusative case.

(51) Who_i/*Whom_i do you think [t_i will arrive first?]

(52) Whom_i/*Who_i do you think [Lord Emsworth will invite t_i?]

Another important distinction to be made among English Wh-elements is the *theta-marking*, i.e. the argument and non-argument distinction. The Functional distribution for Wh-elements and Wh-groups is presented in the table **??** below.

Features	Clause functions of the Wh-group		Group functions of Wh-element
	Subject	**Complement**	
person	who, whoever	whom, whomever, whomsoever	head/thing
person, possessive	whose		possessor
person/non-person	which		determiner
non-person	what, whatever		head/thing
	Adjunct		
various circumstantial features	when, where, why, how (whether, whence, whereby, wherein) (and their *-ever* derivations)		head/modifier

Table 4.3 Functions and features of Wh-elements and groups

There are two places where the Wh-Group can land: (a) either in the subject position of the matrix clause changing its mood to interrogative (example 53) or (b) subject position of the embedded clause creating embedded questions (example 54). However regardless of the landing site the movement principle is subject to *that-trace* filter expressed in generalization 4.2.11) and *Subjacency condition* (generalization 4.2.12)

(53)   Whom$_i$ do [you believe [that Lord Emsworth will invite t$_i$]]?

(54)   I wonder [whom$_i$ you believe [that Lord Emsworth will invite t$_i$.]]

Generalization 4.2.11 (That-trace filter)**.** The sequence of an overt complementizer "that" followed by a trace is ungrammatical (Haegeman 1991: p. 399).

The examples 55 to 58 illustrate how the above generalization applies.

(55)   * Whom do you think that Lord Emsworth will invite?

(56)   Whom do you think Lord Emsworth will invite?

(57)   * Who do you think that will arrive first?

(58)   Who do you think will arrive first?

Generalization 4.2.12 (Subjacency condition)**.** Movement cannot cross more than one bounding node, where bounding nodes are IP[1] and NP (Haegeman 1991: p. 402).

Subjacency condition captures the grammaticality of NP-movement and exposes two property of the movement, namely as being *successive* and *cyclic*. Consider the chain creation resulting from Wh-movement in Examples 59 – 61. The Wh-movement leaves (intermediate) traces successively jumping each bounding node.

(59)   Who$_i$ did [he see t$_i$ last week?]

(60)   Who$_i$ did [Poirt claim [t$_i$ that he saw t$_i$ last week?]]

(61)   Who$_i$ did [Poirot say [t$_i$ that he thinks [t$_i$ he saw t$_i$ last week?]]]

What 4.2.12 states is that a Wh-element cannot move further than subject position and so creating the interrogative form or moving outside the clause into subject position of the higher clause and leaving a Wh-trace.

4.3   Placing null elements into Stanford dependency grammar

This section provides a selective translation of principles, rules and generalizations captured in GB theory into the context of dependency grammar. The selections mainly address the identification of null elements which is later used in the semantic parsing process described in Chapter 7

4.3.1   PRO subject

Coming back to the definition of PRO element, it is strictly framed to the non-finite subordinate clauses. In dependency grammar the non-finite complement clauses are typically linked to their parent via *xcomp* relation defined in Marneffe & Manning (2008a) as introducing an open clausal complement of a VP or ADJP without its own subject, whose reference is determined by an external reference. These complements are always non-finite. Following principles 4.2.1 and 4.2.3 the non-finite complement clause introduced by *xcomp* relation shall receive by default a PRO subject (controlled or arbitrary).

The markers (conjunctions, prepositions or Wh-elements) at the beginning of the clause change the dependency relation to *prepc*, *rcmod*, *partmod* and *infmod* and a slight

---

[1]IP-Inflectional phrase corresponds roughly to the concept of clause (matrix or embedded)

variation in clause features and constituency. The only other relation that introduces a complement clause with PRO element is *prepc* and the only preposition is "whether". The other relations will be discussed latter in this chapter since they correspond to other types of empty elements.

I have explained earlier how to identify the place where PRO element shall be created. Before creating it we need to find out (1) whether PRO is arbitrary (equivalent to pronoun "one") or it is bound to another constituent. And if it is bound then decide (2) whether it is bound to subject or object (case in which we say that PRO is subject or object controlled).

Generalization 4.2.6 introduces a mood test for complement clause. Even is it is non-finite in GB theory it still may be declarative or interrogative. Halliday's MOOD system (Halliday & Matthiessen 2004: p. 107-167) does not allow mood selection for non-finite clauses. The reason declarative mood is mentioned is because GB theory the complement clauses can have declarative or interrogative mood. But as soon as we formulate an interrogative complement clause it can be only a Wh interrogative, thus the test is whether there is a Wh-marker (who, whom, why, when, how) or "whether" preposition. The presence of any marker like in example 62 at the beginning of the complement clause will change the dependency relation and the structure of the dependent clause. The only case when the complement clause remains non-finite and without a subject is the case of "whether" preposition and the clause is introduced via *prepc* relation. However if any such marker is missing then the clause is declarative and thus it must be controlled by a NP, so the arbitrary PRO is excluded. The cases of Wh-marked non-finite clauses will be treated in the further section 4.3.4 on Wh-element movement.

(62)   Albert asked [whether/how/when/ PRO to go.]

Based on the above I propose Generalization 4.3.1 enforcing obligatory control for *xcomp* clauses.

Generalization 4.3.1. If a clause is introduced by *xcomp* relation then it must have a PRO element which is bound to either subject or object of parent clause.

(63)   Albert$_i$ asked [PRO$_i$ to go alone.]

(64)   Albert$_i$ was asked [PRO$_i$ to go alone.]

(65)   Albert$_i$ asked Wendy$_j$ [PRO$_j$ to go alone.]

(66)   Albert$_i$ was asked by Wendy$_j$ [PRO$_i$ to go alone.]

The generalization 4.2.7 appeals to the *voice* feature of the parent clause in order to determine what NP is controlling the PRO element in the complement. Consider 63 and it's passive form 64. In both cases there is only one NP that can command PRO and it is the subject of the parent clause "Albert". So we can generalise that the voice does not play any role in controller selection in one argument clauses (i.e. clauses without a nominal complement). In 65 and 66 the parent clause takes two semantic arguments. Second part of principle 4.2.2 states that in case of obligatory control PRO must be *c-commanded* by an NP. In 65 both NPs("Albert" and 'Wendy") c-command PRO element, however according to Minimality Condition(Haegeman 1991: p. 479) "Albert" is excluded as the commander of PRO because there is a closer NP that c-commands PRO.In case of 66 the only NP that c-commands PRO is the subject "Albert" because "by Mary" is a PP (prepositional phrase) and also only NPs can control a PRO as stated in principle 4.2.6. In the process of passivation the complement becomes subject and the subject becomes a prepositional (PP-by) complement then the latter is automatically excluded from control candidates, thus supporting principle 4.2.7.

Generalization 4.3.2. The controller of PRO element in lower clause is the closest nominal constituent of the higher clause.

The adjunct non-finite clauses (examples 34 and 35) shall be treated exactly as the non-finite complement clauses. Generalization 4.2.8 emphasises obligatory control for them. The only difference between the adjunct and complement clauses is dictated by the verb of the higher clause whether it theta marks or not the lower clause. In dependency grammar the adjunct clauses are also introduced via *xcomp* and *prepc* relations, so syntactically there is not distinction between the two and patterns.

The *prepc* relation in dependency grammar introduces a prepositional clausal modifier for a verb (VN), noun (NN) or adjective (JJ). Adjective and noun modification are cases of copulative clauses. However such configuration are not relevant to the context of this work because the dependency graphs are normalised in the preprocessing phase described in the next chapter Section 6.2. Within the normalization process, among others, the copulas are removed and transformed to verb predicated clauses instead of adjective or noun predicated clauses.

The subject clauses are introduced via *csubj* relation. Subject non-finite clauses are quite different from complement and adjunct clauses because, according to generalization 4.2.9 the PRO is optionally controlled. Since it is not possible to bind PRO solely on syntactic grounds the generalization 4.2.9 proposes arbitrary interpretation.

4.3.2 NP-traces

Syntactically, NP raising can occur only when there is a complement clause by moving the subject of a lower clause into position of higher clause. The subject of higher clause c-commands the subject of lower clause. This is exactly the same syntactic configuration ~~like~~ in the case of PRO subjects. In ~~the~~ dependency grammar the lower clause is introduced via *xcomp* (explained in Section 4.2.1).
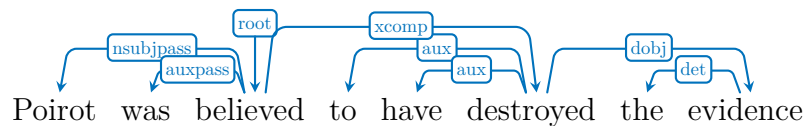

Fig. 4.1 Dependency parse for Example 41

The figures 4.1 and 4.2 represent a dependency parses for examples 41 and 43. To identify the place of the empty category is enough to apply the subject control pattern depicted in 7.3. However, just by doing so it is not possible to distinguish whether the empty subject is a PRO or a NP trace *t*.
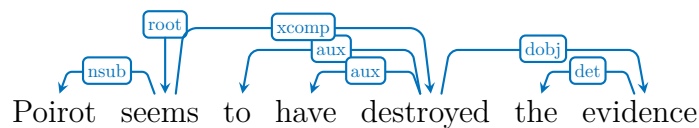

Fig. 4.2 Dependency parse for Example 43

When the empty subject in the embedded clause is detected and instantiated through the *obligatory subject control* pattern it is important to distinguish among the cases.

First of all this distinction is crucial for assignment of thematic roles to the constituents. So the problem is deciding on the type of relationship between the empty constituent and it's antecedent (subject of matrix clause) to which it is bound. In the case of *PRO* constituent, the thematic roles are assigned to both the empty constituent and to its antecedent locally in the clause they are located. So the PRO constituent receives a thematic label dictated by the verb of the embedded clause and the antecedent by the verb of the matrix clause. In the *t*-trace case the thematic role is assigned only to the empty constituent by the verb of the embedded clause and this role is propagated to its antecedent.

Making such distinction directly involves checking role distribution of upper and lower clause verb and deciding the type of relationship between the empty category and

its antecedent. If such a distinction ~~shall~~ be made at this stage of parsing or postponed to Transitivity analysis (i.e. semantic role labeling) is under discussion because each approach introduces different problem.

Before discussing each approach I would like to state a technical detail. When the empty constituent is being created it requires two important details: (a) the antecedent constituent it is bound to and (b) the type of relationship to it's antecedent constituent or if none is available the type of empty element: t-trace or PRO. Now identifying the antecedent is quite easy and can be provided at the creation time but since the empty element type may not always be available then it may have to be marked as partially defined.

The first solution is to create the empty subject constituents based only on syntactic criteria, ignoring for now element type (either PRO or *t*-trace) hence postponing it to semantic parsing phase. The advantage of doing so is a clear separation of syntactic and semantic analysis. The empty subject constituents are created in the places where they should be and ~~it~~ leaves aside the semantic concern of how the thematic roles are distributed. The disadvantage is leaving the created constituents incomplete or under-defined. Moreover the thematic role distribution must be done within the clause limits but because of raising, this process must be broadened to a larger scope beyond clause boundaries. Since transitivity analysis is done based on pattern matching, the patterns rise in complexity as the scope is extended to two or more clauses thus ~~excluding~~ excludes iteration over one clause at a time (which is desirable).

Otherwise, a second approach is to decide the element type before Transitivity analysis (semantic role labeling) and remove the burden~~ed~~ of complex patterns that go beyond ~~the~~ clause borders. Also, all syntactic decisions would be made before semantic analysis and the empty constituents would be created fully defined with the binder and their type but that means delegating semantically related decision to syntactic level (in a way peeking ahead in the process pipeline).

The solution adopted here is mix of the two avoiding two issues: (a) increasing the complexity of patterns for transitivity analysis, (b) leaving undecided which constituents accept thematic role in the clause and which don't.

The process to distinguish the empty constituent type starts by (a) identifying the antecedent and the empty element (through matching the subject control pattern in Figure 7.3), (b) identifying the main verbs of higher and lower clauses and correspondingly the set of possible configurations for each clause (by inquiry to process type database(PTDB) described in the transitivity analysis Section 7.2).

Generalization 4.3.3. To distinguish the *t*-traces check the following conditions:

- the subject control pattern matches the case AND

- the process type of the higher clause is two or three role cognition, perception or emotion process (considering constraints on Cognizant and Phenomenon roles). AND

- among the configurations of higher clause there is one with:

  - an expletive subject OR

  - the Phenomenon role in subject position OR

  - Cognizant in subject position AND the clause has passive voice or interrogative mood (cases of movement).

If conditions from generalization 4.3.3 are met then the empty constituent is a subject controlled *t*-trace. Now we need a set of simple rules to mark which constituents shall receive a thematic role. These rules are presented in the generalization 4.3.4 below.

Generalization 4.3.4. Constituents receiving thematic roles ~~shall be~~ marked with "thematic-role" label, those that do not receive a thematic role ~~shall be~~ marked with "non-thematic-role" and those that ~~might~~ receive thematic role with "unknown-thematic-role". So in each clause:

- the subject constituent is marked with "thematic-role" label unless (a) it is an expletive or (b) it is the antecedent of a *t*-trace then marked "non-thematic-role"

- the complement constituent that is an nominal group (NP) or an embedded complement clause is marked with "thematic-role" label.

- the complement that is a prepositional group (PP) is marked with "unknown-thematic-role".

- the complement that is a prepositional clause is marked with "unknown-thematic-role" label unless they are introduced via "that" and "whether" markers then it is marked with "thematic-role" label.

- the adjunct constituents are marked with "non-thematic-role"

4.3.3   Wh-trances

~~Let's turn now to how~~ Wh-movement and relative clauses are represented and behave in dependency grammar and SF grammars. Figures 4.3, 4.4 present dependency parses for Wh-movement from subject and object positions of lower clause while 4.5 from adjunct position.
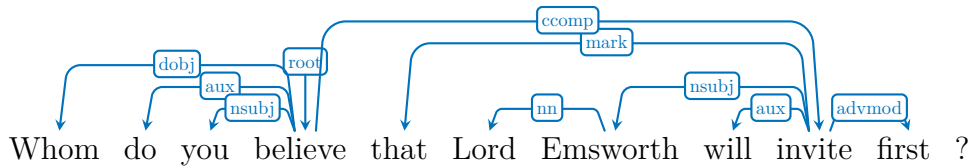
Whom   do   you   believe   that   Lord   Emsworth   will   invite   first   ?

Fig. 4.3 Dependency parse for Example 52
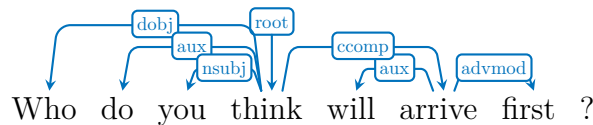
Who   do   you   think   will   arrive   first   ?

Fig. 4.4 Dependency parse for Example 51

Just ~~like~~ in cases of NP-movement, the Wh-groups move only into two and three role cognition, perception and emotion figures. In contrast, if the NP-antecedents land in expletive or passive subject position then the Wh-antecedents land in subject or subject preceding position functioning as subject, complement or adjunct functions depending on the Wh-Element.

The essential features for capturing ~~the~~ Wh-movement in dependency graphs are (a) the finite complement clause identified by *ccomp* relation between the matrix and embedded clause (b) the Wh-element/group plays a complement function in higher clause which is identifiable by *dobj*, *prep* or *advmod* relations to the main verb (c) the function of the Wh-trace in lower clause is either Subject(e.g. 52), Object(e.g. 51) or Adjunct.

ccomp relation is defined in Marneffe & Manning (2008a) to introduce ~~a~~ complement clauses with an internal subject which are usually finite. ~~I must emphasize the fact that the~~ lower clause must be embedded into the higher one and receive a thematic role in higher clause.

Regardless whether the syntactic function of the traces in the lower clause is Subject or Object, in the higher clause the Wh-group takes Object function and is bound to the main verb via *dobj* relation but is positioned before the main verb and the Subject

(a structure corresponding to Wh-interrogatives). Wh-group can take also Subject function in the higher clause, but then it is not a case of Wh-movement and is irrelevant for us at this point because there is no empty element, i.e. Wh-trace. The attribution of clause function to the Wh-trace is based on either the case of the Wh-group or the missing functional constituent in the lower clause.
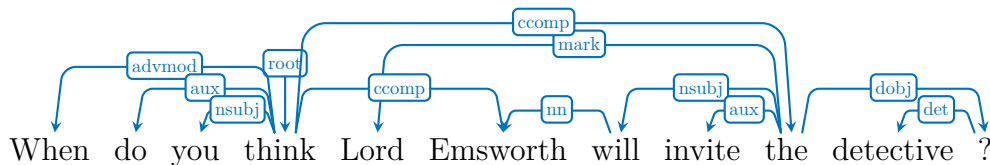

Fig. 4.5 Example dependecy parse with Adjunct Wh-element

In case of Wh-traces with Adjunct function in the lower clause, like in figure 4.5, their antecedents also receive adjunct function in the higher clause. Adjunct Wh-group cannot bind to the clause it resides in if the clause has (generic) present simple tense and thus it has to be antecedent for a trace in lower clause which has other tense or modality than present simple. The reader can experiemnt with changing tense in the example 4.5.

(67)   Who$_i$ believes that Lord Emsworth will invite a detective?

(68)   To whom$_i$ did Poirot say t$_i$ that Lord Emsworth will invite a detective?

Not always the Wh-groups are movements from lower clause. It is possible that the trace of the moved element to reside in the higher clause (complement) or even have a case of no movement when Wh-element takes the subject function in the higher clause. 67 and 68 are good examples of a *short* movement (in clause movement). However the short movement in dependency grammar has no relevance because the dependency grammar is order free and the functions are already assigned accordingly so short movement is not a subject of interest for the current chapter.
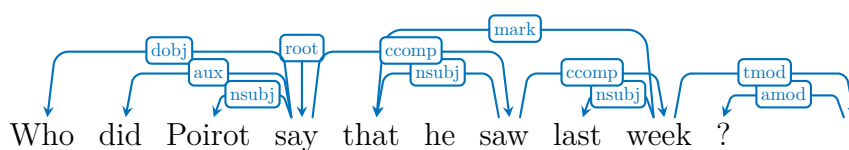
4.3.4   Chaining of Wh-trances

Fig. 4.6 Dependency parse for Example 61

Recall the *cyclic* and *successive* properties of Wh-movement from previous section underlined by example 61 and its dependecy parse in figure 4.6. GBT suggests that the Wh-movement leaves traces in all the intermediary clauses. In dependency grammar these properties shall be treated instrumentally for determining intermediary hops in search for the foot of the chain and none of the intermediary traces shall be created. There simply is no further purpose for them as they do not receive a thematic role in the intermediary clauses.

4.3.5   Wh-trances in relative clauses

In GB theory the Wh-elements that form relative clauses (*who, whom, which, whose*) are considered moved. In dependency grammar such movement is redundant since the Wh-element and its trace are collapsed and take the same place and function. The Wh-elements function either as subject or complement. When the relative clause is introduced by a Wh-group there is no empty element to be detected, rather there is anaphoric indexing relation to a noun it refers to. Focusing now on the relative clauses, there are three more possible constructions that introduce them: (a) a prepositional group that contains a Wh-element, (b) "that" complementizer which behaves like a relative pronoun (c) the Zero Wh-element which is an empty element and which functions the same way as a overt Wh-element. The table 4.4 lists possible elements that introduce a relative clause, their features and the functions they can take.

Next I discuss how to identify, create and bind the traces of Wh-elements to their antecedents.

Relativizing element	Feature	(Clause) Function	Examples
who	person	subject	... the woman who lives next door.
whom	person	complement (non defining clause)	... the doctor whom I have seen today.
which	non-person	subject/ complement	... the apple which is lying on the table. ... the apple which George put on the table.
whose	possessive, person	possessor in subject	... the boy whose mother is ~~in~~ a nurse.
(*any of the above in prepositional group*)	person/ non-person	thing/possessor in subject	... the boy to whom I gave ~~an~~ advice. ... the cause for which we fight.
that	person/ non-person	subject	... the apple that lies on the table.
Zero Wh-element	person/ non-person	subject	... the sword sent by gods.

Table 4.4 The Wh-elements introducing a relative clause.

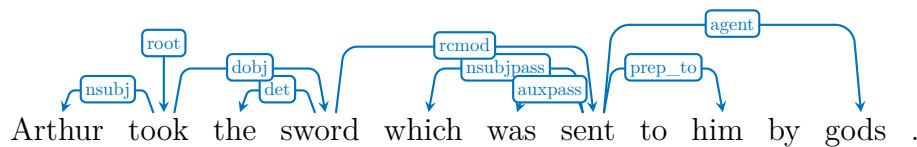Compare the dependency parse with an overt Wh-element in Figure 4.7 and the covert one in 4.8.

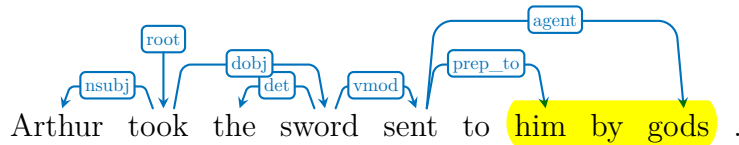Fig. 4.7 Dependency parse for "Arthur took the sword which was sent to him by gods."

Fig. 4.8 Dependency parse for "Arthur took the sword sent to him by gods."

Relative clauses in dependency graphs are introduced by *rcmod, partmod, infmod* and *vmod* relations. The *rcmod* introduces relative clauses containing a Wh-group while the *partmod* and *infmod* introduce finite and non-finite relative clauses with Zero Wh-element. After ~~the~~ Version 3.3 of Stanford parser the *partmod* and *infmod* relations have been merged into *vmod*. So the dependency relation is the main signaller if empty Wh-elements even though they are subject to corrections in preprocessing (Section 6.2) phase to ensure a uniform treatment of each relation type.

The Zero Wh-element behaves exactly like the *PRO element* in the case of non finite complement clauses discussed in Section 4.2.1. It receives thematic roles in both the higher clause and in lower clause and is not a part of a chain like the cases of NP/Wh-movement.

4.4   Discussion

This chapter ~~treats~~ the identification of the *null elements* in syntactic structures. Section 4.2 presents how GBT theory handles null elements and then Section 4.3 shows how the same principles translate into Stanford dependency graphs.

Identification of null elements is important for the semantic role labeling process described in Chapter 7 because usually the missing elements are participant roles (theta roles) shaping the semantic configuration. The semantic configurations (gathered in a database) are matched against the syntactic structure and missing elements lead to failing (false negatives) or erroneous matches (false positives). Therefore to increase the accuracy of semantic role labeling spotting null elements is a prerequisite.

This Chapter also contributes to establishing cross theoretical connections that ~~is among~~ current thesis objectives. Specifically it provides translations of necessary principles and generalizations from GB theory into the context of dependency grammar. These results ~~are directly used~~ in Section 7.1 for generating graph patterns.