Parsimonious Vole

A Systemic Functional Parser for English

Universität Bremen

Eugeniu Costetchi

Supervisor: Prof. John Bateman

Advisor: Dr. Eric Ras

Faculty 10: Linguistics and Literary Studies
University of Bremen

This dissertation is submitted for the degree of
Doctor of Philosophy

October 2018

I would like to dedicate this thesis to my loving parents ...

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Eugeniu Costetchi
October 2018

Acknowledgements

And I would like to acknowledge ...

Abstract

This is where you write your abstract ...

Table of contents

List of figures

List of tables

List of definitions

Chapter 1

Creating the systemic functional constituency structure

The considered input into the parsing pipeline are Stanford dependency parse graphs. These graphs are then rewritten into systemic constituency graphs. The process by which this happens is the focus of the current chapter. Dependency graphs are sometimes erroneous or treat certain linguistic phenomena in an incompatible way with current approach. Therefore a preprocessing stage is needed to correct and canonicalise the dependency graphs which is covered in the first two sections of the chapter. Then in the last section I present in detail the process by which the systemic functional constituency parse graph is created.

1.1 Canonicalisation of dependency graphs

Beside stable errors, there are two other phenomena that are modified in the preprocessing phase: *copula* and *coordination*. They are not errors per se but simply an incompatibility between how Stanford parser represents them and how they need to be represented for processing by the current algorithm and grammar.

In this section I describe a set of transformation operations on the dependency graph before it is transformed into systemic constituency graph. The role of preprocessing phase is bringing in line aspects of dependency parse to a form compatible with systemic constituency graph creation process by (a) correcting known errors in DG, (b) cutting down some DG edges to form a tree (c) changing Stanford parser's standard handling of copulas, coordination and few other phenomena. This is achieved via three transformation types: (a) *relabelling of edge relations*, (b) relabelling node POS, and (c) reattachment of nodes to a different parent.

1.1.1 Loosening conjunction edges

Stanford parser employs an extra edge for each of the conjuncts such that there is one indicating the syntactic relationship to the child or parent nodes (just like for any other nodes) and additionally one that shows the conjunction relationship to its sibling nodes. This process removes the parent or child relations except for the first conjunct and leaves only the sibling relationships.

Some common patterns occurring between noun, verb and adjective conjuncts are depicted below in figures 1.1 - 1.5.

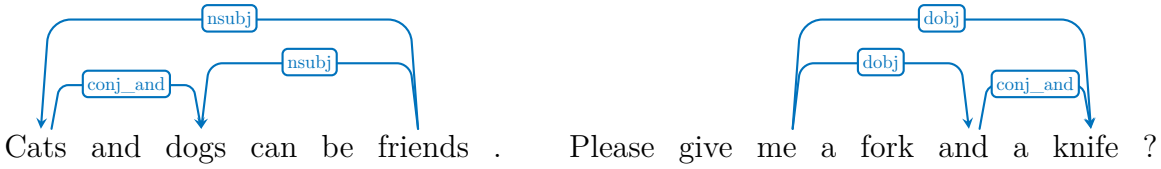

Fig. 1.2 Conjunction of noun objects

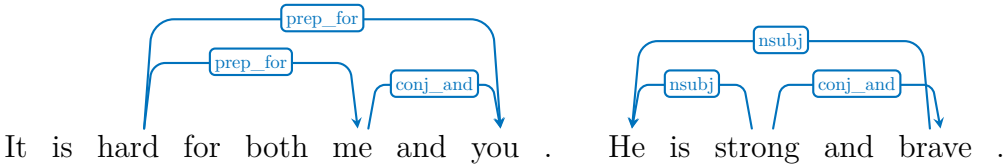

Fig. 1.4 Conjunction of copulatives sharing the subject

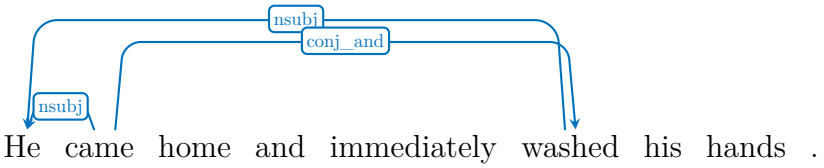

Fig. 1.5 Conjunction of verbs sharing the same subject

The main reason these extra edges need to be removed is to avoid traversal of the same node via different paths. This, in the current algorithm, has as consequence execution multiple times of the same operation such as for example creation of multiple constituents from the same DG node, which of course is undesirable. For example, if multiple subject relations occur in the DG then multiple subject are going to be instantiated in CG which is not covered in the grammar. Rather only one complex unit needs to be created with the subject role composed of two noun phrases (see discussion of this case in the Section **??**).

The way I fix this problem is by removing functional edges to/from each conjunct except the first one. There are two generic patterns in figures 1.6 and 1.8 correspondingly with incoming and outgoing edges that are transformed into the forms depicted in 1.7 and 1.9.

I split the cases into two: patterns with incoming dependency edges and outgoing ones. First, see the pattern of conjuncts with *incoming dependency* relations represented in Figure 1.6 and exemplified in Figures 1.1 - 1.3. In SFG terms it corresponds to cases when the functional element of a parent constituent is filled by a complex unit below.

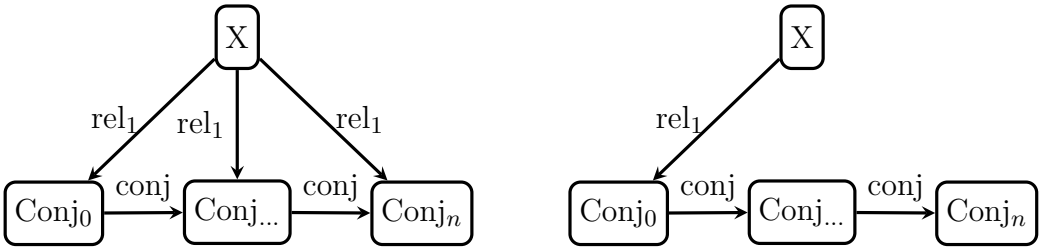

Fig. 1.7 Conjuncted elements with incoming loosely connected dependencies

The second is the pattern of conjuncts with *outgoing dependency relations* depicted in Figure 1.8. In SFG terms it correspond to cases when a unit is sharing an element with another conjunct unit. These are mainly the cases of conjuncted verbs or copulas and are further discussed in the Chapter **??** about null elements. In GBT terms, the second to last conjuncts may miss for example the subject constituent if the conjuncts are verbs or copulas as in Figures 1.4 - 1.5.

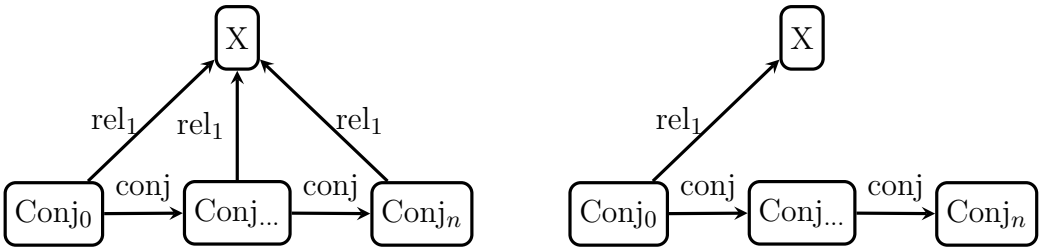

Fig. 1.9 Conjuncted elements with outgoing loosely connected dependencies

1.1.2 Transforming copulas into verb centred clauses

In Stanford dependency grammar *copular verbs* are treated as dependants of their complements (see Figures 1.10 and 1.11) because of the intention to maximize connections between content words. This configuration breaks the rule of the main verb being the head of clause discussed in Sections **??** and **??**.

Moreover, despite that a variety of verbs are recognised as copulative e.g. *act, keep, sound*, etc. Stanford parser provides copula configurations only for the verb *to be* leading to unequal treatment of copular verbs.

This case is sometimes accompanied by two relations that create cycles in th DG. They are the *xsubj*, the relation to a controlling subject and *ref*, the relation to a referent. The two relations are removed and their resolution is transferred to the semantic analysis stage of the algorithm.

Fig. 1.11 Conjunction of copulatives sharing the subject

To make the copulative verb the roots of its clause, following rules are implemented. First, some relations are transferred from the copula complement (adjective JJ or noun NN) to the copulative verb. The transferred relations are listed in Table 1.1 which distinguishes them based on the part of speech which of the *copula complement*.

part of speech	*dominated relation*
NN	dep, poss, possesive, amod, appos, conj, mwe, infmod, nn, num, number, partmod, preconj, predet, quantmod, rcmod,ref, det
JJ	advmod, amod, conj

Table 1.1 Relations dependent on the POS of the dominant node

Second, all the outgoing connections from the copula complement are transferred to the verb except those listed in second column of table 1.1, these relations must stay linked to the NN or JJ nodes. Third the *cop* relation is deleted. Fourth, all the incoming relations to the copula complement are transferred indistinguishably to the verb because these are all clause related and shall be linked to the clause dominant node. Finally the *dobj* link is created from the verb to the complement noun/adjective.

Figure 1.12 represents the generic pattern of copulas in Stanford DGs. The outgoing relations are distinguished between those in the filter as *rel_dep* and the rest simply as *rel* while the incoming relations are not discriminated. Figure 1.13 captures the final state of the transformation where the filtered outgoing relations stay attached to the complement node while the rest incoming and outgoing relations are moved to the verb.

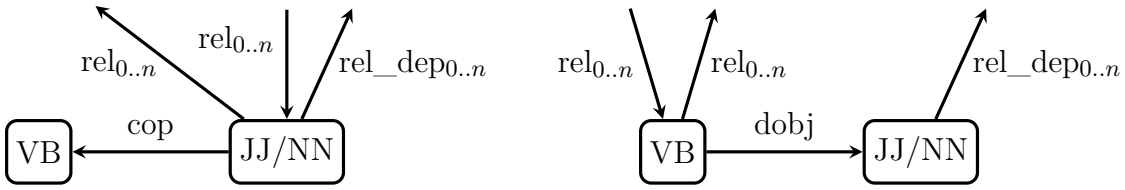

Fig. 1.13 Generic pattern for copulas after the transformation (the same as non-copular verbs).

In case of conjuncted copulas like in the example in Figure 1.11 the approach is slightly complicated by the fact that copula resolution algorithm shall be executed for each copula conjunct, however because of the previous step which is loosening the conjunction and removing graph cycles then only the first copula conjunct is concerned.

1.1.3 Non-finite clausal complements with adjectival predicates (a pseudo-copula pattern)

The Figure 1.14 represents a dependency parse exemplifying a clausal complement with an adjectival predicate. In this analysis there is a main clause governed by the verb *to paint* and the second one by the adjective *white*. In SFL Figure 1.14 receives a different analysis as it is represented in Table 1.2.

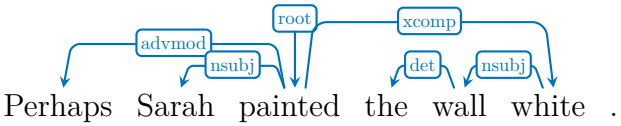

Fig. 1.14 Dependency parse for clausal complement with adjectival predicate

xcomp relation defined in (Marneffe & Manning 2008a) to introduce non-finite clausal complement without a subject. Dependency grammar allows adjectives (JJ) and nouns (NN) to be heads of clauses but only when they are a part of a copulative construction. In figure 1.14 it is not the case, there is no copulative verb *to be* and also *the wall* receives the subject role in the complement clause which should be absent.

Perhaps	*Sarah*	*painted*	*the*	*wall*	*white.*
Adjunct	Subject	Finite/Main Verb	Complement		Complement
	Agent	Material Action	Affected		Attribute

Table 1.2 SFG analysis with attributive adjectival complement

So I treat it as a misuse of *xcomp* relation and the adjective should not be treated as governing a new clause but rather non-clausally complementing the verb *to paint*. Moreover that in SFG adjectival predicates are not allowed.

Certainly, depending on the linguistic school, opinions may diverge on the syntactic analysis comprising one or two clause. But when analysed from a semantic perspective it is hard to deny that there is a Material Process with an Agent and Affected thing which is specifying also the resultant (or goal) Attribute of the Affected thing.

To accommodate such cases the dependency graph is changed from pattern in Figure 1.15 to form in Figure 1.16. The *xcomp* relation is transformed into *dobj* and the subject of the embedded clause (if any) becomes the direct object (*dobj*) in the main clause.

Fig. 1.16 Adjectival clausal complement as secondary direct object

1.2 Correction of errors in dependency graphs

The Stanford Parser applies various machine learning(ML) techniques to parsing. It's accuracy increased over time to ≈ 92% for unlabelled attachments and ≈ 89% for labelled ones (in the version 3.5.1). This section addresses known error classes of wrongly attached nodes or wrongly labelled edges and nodes. These errors have been discovered during the development of Parsimonious Vole parser and are corrected to increase the result accuracy.

As the Stanford parser evolved, some error classes changed from one version to another (v2.0.3 – v3.2.0 – 3.5.1). Also the set of dependency labels for English initially

described in (Marneffe & Manning 2008a,b) changed to a cross-linguistic one (starting from v3.3.0) described in (Marneffe et al. 2014).

As noted by Cer et al. (2010) the most frequent errors are related to structures that are hard to attach i.e. prepositional phrases and relative clauses. During the implementation of current parser there had been discovered a set of errors, most frequent of which are described in this section and how are they treated. These errors are specific to Stanford Parser versions v2.0.3 – 3.2.0. This section may constitute a valuable feedback for SDP error analysis.

1.2.1 *prep* relations from verb to free preposition

As noted before only the collapsed version of the DGs are taken as input. This means that no pure *prep* relations shall occur but their expended version with the specific preposition appended to the relation name i.e. *prep_xxx*.

This is not always the case especially with phrasal verbs, the *prt* relations are mislabelled as *prep*. The correction consist in changing the *prep* (figure 1.17) into *prt* (figure 1.18) if the preposition node has no children e.g. *pobj*.

Fig. 1.18 Corrected relation to free preposition as verbal particle

1.2.2 Non-finite clausal complements with internal subjects

The *xcomp* relation stands for open clausal complements of either a verb(VB) or adjective (JJ/ADJP). The latter is actually transformed as discussed in Section 1.1.3. The open clausal complement defined in Lexical Functional Grammar (Bresnan 2001: p270–275) is always non-finite and does not have its own subject. However sometimes *xcomp* relation appears either (a) with finite verbs or (b) with own local subjects and both cases correspond to definition of *ccomp* relation.

To fix this I transform all the instances of *xcomp* relation to *ccomp* if the dependent verb has a local subject (nsubj) or a finite verb as depicted in Figures 1.19 - 1.20.

Fig. 1.20 Corrected clausal complement

1.2.3 The first auxiliary with non-finite POS

Sometimes the first auxiliary in a clause is mistakenly labelled as a non-finite verb. For some words the exact POS is less important as it has not big impact on the CG graph and features but in the case of first auxiliary verb of a clause it makes a big difference. It has an impact on determining the finiteness of the clause in a latter stage of the algorithm.

The algorithm is thus checking that the POS of the first auxiliary is according to the mapping defined in the Table 1.3.

word	*POS*	*notes*
shall, should, must, may, might, can, could, will, would	MD	modals
do, have, am, are	VBP	present
has, is	VBZ	present 3rd person
did, had, was, were	VBD	past

Table 1.3 Mapping lexical forms of auxiliaries to their POS

1.2.4 Prepositional phrases as false prepositional clauses

prepc is a relation that introduces, via a preposition, a clausal modifier for a verb, adjective or noun. Assuming that the copulas had been changed as described in subsection 1.1.2 then the head and the tail of the relation can only be a verb. However when the relation head is not a verb (only nouns encountered so far) then the relation needs to be corrected from *prepc* to *prep* introducing a prepositional phrase rather than a subordinate clause.

Fig. 1.22 Corrected prepositional phrase

1.2.5 Mislabelled infinitives

In English base form of the verb often coincides with present simple form (non 3^{rd} person). Therefore the POS tagger sometimes mislabels infinitive (VB) as present simple (VBP) the verb is and vice versa.

The algorithm checks the presence of the preposition *to* (linked via *aux* dependency relation) in front of the verb. If the preposition is present then the verb POS is changed to VB and reverse, if the auxiliary preposition is not present the verb POS is changed into VBP.

Fig. 1.24 Correct infinitive

Fig. 1.26 Correct present simple

1.2.6 Attributive verbs mislabelled as adjectives

In English, *attributive verbs* often have the same lexical form as their corresponding adjectives. This is a reason for POS being mislabelled adjective(JJ) instead of verb (VB) leading to situations when an adjective (JJ) has an outgoing subject relation which means that its POS should actually be VB. The algorithm checks for such cases and corrects the JJ POS into VBP (non 3^{rd} person present simple).

Fig. 1.28 Corrected attributive verb

1.2.7 Non-finite verbal modifiers with clausal complements

The early version of Stanford Dependencies (Marneffe & Manning 2008a) proposes two relations for non-finite vernal modifiers *partmod* for participial and *infmod* for infinitival forms exemplified in 1.Latter in (Marneffe et al. 2014) both relations have been merged into the *vmod*.

(1) Tell the boy playing the piano that he is good.

Clauses such as "(that) he is good" following immediately after the qualifier clause ("playing the piano" in the example 1) are problematic with respect to where shall they be attached: to the main clause or to the modifying one. This problem is similar to the prepositional phrase attachment problem.

In this case, of course, attachment would depend on whether the verb accepts a clausal complement or not. In the example 1 the verb *to play* does not take clausal complements then the clause "that he is good" is complementing "tell the boy". Stanford parser does not take into consideration such constraints and sometimes provides an incorrect attachment.

This type of error can be captured as the graph pattern in Figure 1.29 which is transformed by the algorithm into the form represented in Figure 1.30

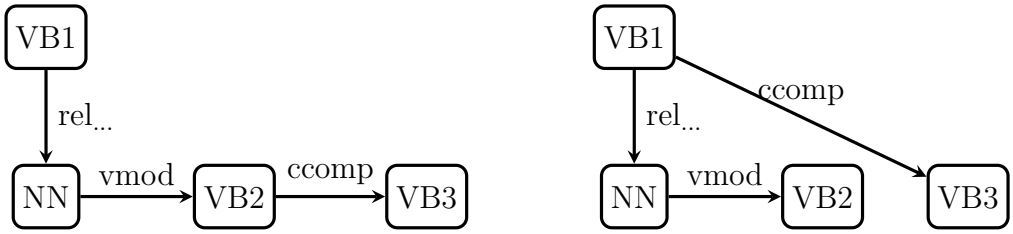

Fig. 1.30 Clausal complement attached to the main clause

Syntactic structure is not enough to capture this error which originates in the semantic influences on the syntax. To grasp the them an extra constraint check is the possible lexico-semantic type of the verb. As only verbal and cognition process types can take clausal complements as phenomena then the verb in the higher clause VB_1 heeds to be capable of accepting clausal complement VB_2 before performing the reattachment. In other words, if VB_1 is capable of accepting two complements (i.e. di-transitive) then most likely the VB_2 is a complement, otherwise it is certainly not.

1.2.8 Demonstratives with a qualifier

Demonstratives (*this, that, these, those*) occurs as both determines and as pronouns. In English, when demonstratives are used as determiners, they function as Deictic element of a nominal group i.e. modifying the head of the nominal group. When used as pronouns, demonstratives never form phrases but occur as single words filling a clause element. Translated into dependency grammar demonstratives may have as parent either a noun (NN) or a verb (VB*).

Examples below show uses of demonstratives in both cases. The word (thing/things)* enclosed between round brackets are not part of the sentence but are elipted.

(2) Bill moved those beyond the counter.

(3) Put that in our plan.

(4) Look at those (things)* beyond the counter.

(5) What is that (thing)* next to the screen?

(6) I thought those (things)* about him as well.

(7) He felt that (thing)* as a part of him.

When demonstratives are followed by a prepositional phrase the question arises whether it shall be attached to the verb and take a clause role or it should be attached to the demonstrative as post-modifier. I shall note that demonstratives cannot take by themselves a post-modifier in either case as determiner or pronoun.

However there are cases when apparently the post-modifier (prepositional phrase) pertains to the demonstrative like in the examples 4 and 5 and cases such as 6 and 7 when the post-modifier pertains to teh clause.

In fact the only acceptable analysis for apparently a demonstrative with a Qualifier (i.e. post-modifier) can be analysed as noun phrases with the Thing missing (elipted) and the Deictic taking the role of the Head. The implied missing head is the generic noun “thing(s)” or any noun anaphorically binding the demonstrative.

The verb argument structure and syntactic constraints on the arguments described in the Transitivity classification of process types enable precise distinctions of such cases. However at this stage the algorithm does not employ this type of information. Therefore as a rule of thumb, the prepositional phrase following the demonstrative shall be attached to the verb in the case of non-projective[1] di-transitive verbs which are *three role actions* and *directional* processes.

[1]Projective verbs express cognitive and verbal processes like saying, thinking or imagining and often they verbs are di-transitive.

In examples 2 and 3, attaching the prepositional phrase to the demonstratives (depicted in Figure 1.31) is incorrect. It should be attached to the verb (like in the figure 1.32) because the prepositional phrase can function as Destination or Location in each case i.e take semantic roles.

The algorithm detects cases of demonstratives that have attached a prepositional phrase. If the parent verb is a three role action or a directional process then the prepositional phrase is reattached to the verb.

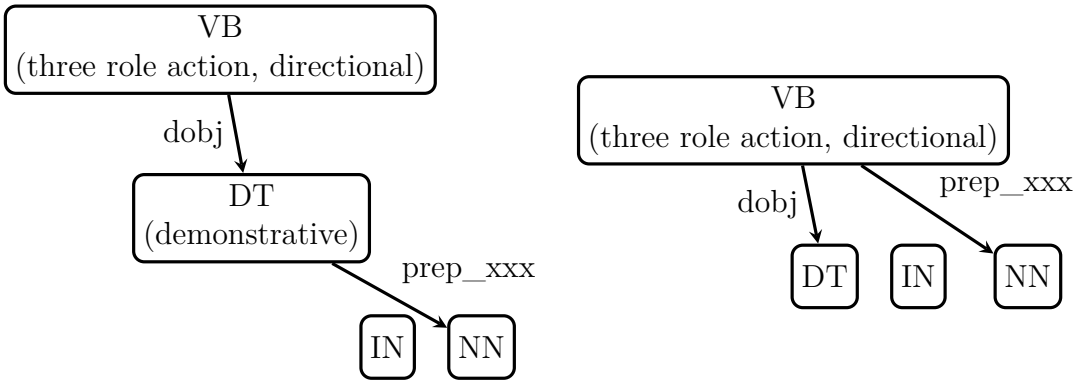

Fig. 1.32 Prepositional Phrase attached to the verb with a demonstrative pronoun in between

Ideally, the algorithm should also change the POS of the demonstrative into pronoun but unfortunately Penn tag-set only contains personal and possesive pronouns. The demonstratives are always labelled as determiners so no POS change is made to the dependency graph but it is properly represented when converted into the constituency graph.

1.2.9 Topicalized complements labelled as second subjects

In generative grammars the topicalization (or thematic fronting) of complements is described in *Trace Theory* as *WH/NP/PP-movement*. Examples 8–13 (from (Quirk et al. 1985: pp. 412-413)) present this phenomena. It is used in informal speech where it is quite common for an element to be fronted with a nuclear stress thus being informationally and thematically stressed. Alternatively this phenomena is used as a rhetorical style to point parallelism between two units and occurs in adjacent clauses like in examples 12–13.

(8) Joe(,) his name is.

(9) Relaxation(,) you call it.

(10) Really good(,) cocktails they make at the hotel.

(11) Any vitamins(,) I could be lacking?

(12) His face(,) I'm not found of but his character I despise.

(13) Rich(,) I may be (but that does not mean I'm happy).

These are difficult cases for Stanford parser (tested with versions up to 3.5.1). None of the above examples are parsed correctly. However, if the comma is present between topicalized complement and the subject, then it produces parses that are closest to the correct one where the topicalized complement is labelled as second subject but still not a complement. So having a comma present helps.

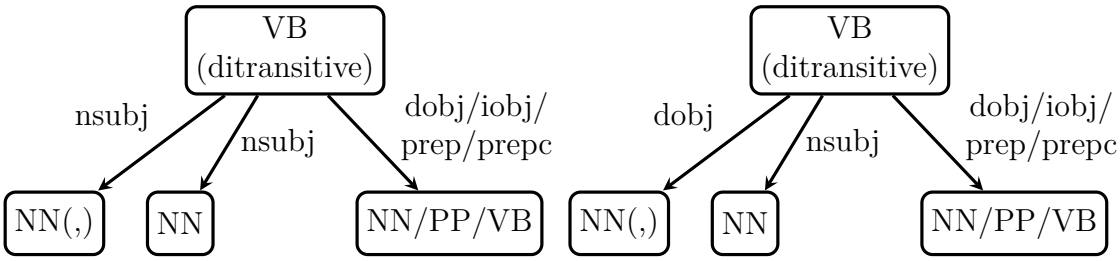

Fig. 1.34 Topicalized Direct Object – moved to pre-subject position

The algorithm is looking for the cases of multiple subjects (represented in figure 1.33) and gives priority to the one that is closest to the verb. The other one is relabelled as a complement (Figure 1.34). The rule is generalized in the algorithm for multiple subjects even if so far only cases of two subjects have been observed.

1.2.10 Misinterpreted clausal complement of the auxiliary verb in interrogative clauses

Sometimes the auxiliary verb in the interrogative clauses (examples 14 and 15) is mistakenly used as a clause main verb. Instead of *aux* relation from the main verb to the auxiliary there is a clausal complement relation from the auxiliary to the main verb.

(14) Do you walk alone?.

(15) Has Jane fed the cat?.

The algorithm searched for the pattern depicted in Figure 1.35 and transforms it into the form Figure 1.36.

Fig. 1.36 Corrected clausal complement

1.3 Creation of systemic constituency graph from dependency graph

This section describes how the systemic constituency structure is generated from the dependency graph. This is known in computer science as graph/tree rewriting.

In the prototype implementation no pre-existing algorithm has been used for graph rewriting but the focus was directed towards understanding the specificities of transforming from dependency into systemic constituency graphs. In the future work the algorithm can be replaced with the state of the art methods in graph rewriting.

The currently implemented process consists of DG traversal and execution of generative operations on a parallel structure, progressively building the CG. The choice of the generative operation is based on a rule table described in Section 1.3.3 The DGs and CGs resemble each other but they are not isomorphic. The CG is created in two phases presented in the Algorithm 1. The first phase, through a top-down breadth-first traversal of a DG, generates, using a rule-set, an incomplete CG that misses the heads and few other elements for each CG unit. Also no unit classes are specified, with except if that unit is a clause. The second phase, through a bottom-up DG traversal, complements the first one ensuring creation of all CG constituents, corresponding to missing unit elements and assigns the unit class.

Algorithm 1: Creation of the constituency graph

input : dg (the dependency graph), rule table
output : cg (the constituency graph)

```
1 begin
2 |   create the partial cg by top-down traversal
3 |   complete the cg by bottom-up traversal
4 end
```

Before presenting the two stages of creation I will first reiterate over the difference in the dependency nature in the constituency and dependency graphs. Then I will also talk about the tight coupling of the two graphs and the rule tables used in traversal.

1.3.1 Dependency nature and implication on head creation

In Section **??** is explained the different dependency relation nature in dependency graphs and in systemic functional constituency graphs. The DG uses a *parent-daughter dependency* while in Constituency Graphs there is a *sibling dependency*. This difference implies that, when mapped into CG, a DG node, stands for both a unit and that unit's head. In other words a DG node corresponds to two functions and unit classes at different rank scales. For example the root verb in DG corresponds to the clause node and the lexical item which fills the Main Verb of the clause.

The two functions at different rank scales is the main reason why the creation algorithm is separated into two phases with a traversal top-down and another bottom up. In current approach, the top-down perspective considers the DG node functioning in the upper rank. The result of the top-down phase is a constituency graph without head (and sometimes few other) elements/nodes.

The bottom-up perspective considers the DG nodes functioning in the lower rank and aims at creating the remaining nodes, mainly heads. The bottom-up phase is performed by traversing the constituency graph and not on the dependency graph. As the dependency nature in CG is among siblings, the traversal task seeks to spot and locally resolve the missing elements. The local resolution is performed based on the syntagmatic unit structure with the aid of the tight coupling between the dependency and constituency graphs that is explained in the section below.

1.3.2 Tight coupling of dependency and constituency graphs

At the creation stage, the CG is tightly coupled with the original DG. It means that each CG node has associated a corresponding DG node in DG together with the set of immediate child nodes. This coupling allows navigating easily from one graph to the other one via stored references. We say that a graph node is *aware* of its ascription in another graph if it carries information to which nodes it is linked within the second graph.

Through a stack of CG nodes, the DG nodes are made aware of which CG nodes and in which order they subsume them (Figure 1.37). On the other hand, the CG nodes are also made aware through a list of DG nodes, over which DG nodes do they

span (Figure 1.38). This way the positioning information is available bidirectionally about CG an DG structures.

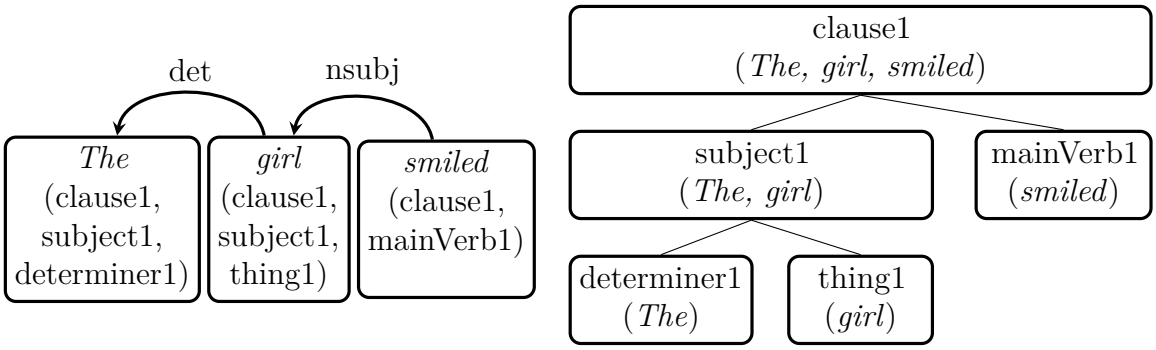

Fig. 1.38 Dependency aware CG nodes

The Figure 1.37 depicts a dependency graph. Each node has a stack of ids corresponding to constituents in the CG (Figure 1.38). Conversely, in Figure 1.38, depicts a constituency graph where the constituent nodes carry a set of tokens corresponding to DG nodes. This way the DG in Figure 1.37 and the CG in 1.38 are aware of each other.

This node awareness has two interesting properties worth exploring. First, the DG nodes receive a vertical constituency strip. Each strip is a direct path from the root to the bottom of the constituency graph where the word of the DG is found. These strips are the very same ones explored in the parsing method explored by Day (2007). Second, the CG nodes receive a horizontal span over DG nodes enabling exploration of elements linear order. These two properties could eventually be explored in future work to inform or verify the correctness of the constituency graph. In the present work the application of the node awareness is limited to the complete construction of the CG.

1.3.3 Rule tables

Constituency Graphs are created through a top-down breadth-first walk of dependency graph. During the top-down breadth-first walk of DG each visited node triggers execution of a *generative operation* on the growing CG on the side. To know what operation to execute a rule table is used where the edge, head and tail nodes are mapped to a generative operation and eventually a parameter (specifying the element type if a constituent is to be created).

A simplified example of the rule table is presented in Table 1.4. It can be regarded as an attribute value matrix (implemented as a Python dictionary) where the left

column, the *key*, contains a unique *dependency graph context* serving as a rule trigger; while the left side column, the *value*, contains the operation to be executed within the given context.

	Key	**Value**	
		Operation	***Parameter***
1	nsubj	new constituent	Subject
2	csubj	new constituent	Subject & clause
3	prepc	new constituent	Complement, Adjunct & clause
4	VB-prep-NN	new constituent	Complement, Adjunct
5	NN-prep-NN	new constituent	Qualifier
6	VB-advmod-WR	new constituent	Complement
7	VB-advmod-RB	new constituent	Adjunct
8	mwe	extend current	
9	nn	extend current	

Table 1.4 Example of rule table mapping specific and generic dependency context to generative operations

Current implementation uses three operation types: (a) *creating a new constituent* under a given one (b) *creating a new sibling* to the given one (c) *extend* a constituent with more dependency nodes.

The parameter is used only for the operations (a) and (b) and specifies which element the new constituent is filling as described in Section **??** and **??**. Most of the time there is only one element provided but in the case of prepositional phrases and clauses it is impossible to specify purely on syntactic basis the exact functional role and thus multiple options are provided (Adjunct or Complement) and then in lather parsing phases, when the verb semantic configurations are verified, these options are reduced to one.

There are two types of keys in the rule table: the *generic* ones where the key consist of the (non-extended) dependency relation and the *specific* ones surrounded by the POSes of head and tail edge nodes taking the form *Tail–relation–Head*. For example *nsubj* relation (row 1) always leads to creation of a Subject nominal constituent regardless if it is headed by a noun, pronoun or adjective. Since all individual cases lead to the same outcome it suffices to map the dependency relation to the creation of a new constituent with Subject role ignoring the POS context of head and tail nodes. The same holds for *prepc* relation (row 3) as it always leads to creation of subordinate clause constituent with Complement or Adjunct roles. So the generic relations can be

viewed as equal to the form *Any–relation–Any* only that the nodes are omitted due to redundancy.

In the case of *prep* relation (rows 4 and 5) the story is different. Its interpretation is highly dependent on its context given by the parent/tail and child/head nodes. If it is connecting a verb and a noun then the constituent prepositional phrase takes the role of either Complement or Adjunct in the clause. But if the prep relation is from a noun to another noun, then it is a prepositional phrase with Qualifier function in the nominal group.

Some dependency relation are not mapped to operation of creating a new but rather extend the existing constituent with all nodes succeeding current one in the DG. These operation is used for two reasons: either (a) the constituent truly consists of more than one word, for example the cases of multi word expressions (e.g. ice-cream) marked via *mwe* relation (row 8) or (b) the relation (with or without it's POS context) is insufficiently informative for instantiating a constituent node and is postponed for the second phase of the CG creation.

Note that the contextualised relations are "slightly" generalised by reducing POS to first two letters which can be up to four letters long. For example nouns generically are marked as NN but they may be further specified as NNP, NNPS and NNS or verbs (VB) may be marked as VBD, VBG, VBN, VBP, and VBZ depending on their form.

Now that I covered the rule table structure I briefly present the algorithm for making rule selections based on simple or contextualised keys.

Algorithm 2: Operation selection in the rule table based on the edge type

```
   input  : rule table, edge
   output : rule
 1 begin
 2     generic key ← the simplified label on the edge
 3     head POS ← the POS of the edge head node
 4     tail POS ← the POS of the edge tail node
 5     specific key ← concatenate (tail POS + generic key + head POS)
 6     if specific key index in rule table:
 7         rule ← value for specific key from rule table
 8     elif generic key index in rule table:
 9         rule ← value for generic key from rule table
10     else:
11         rule ← None
12     return rule
13 end
```

The Algorithm 2 is based on two dictionary lookups: one for specific key (contextualised by the edge relation and its nodes) and another one for generic key based on the edge relation alone. The rule table is conceived as a Python dictionary with string keys and and two-tuple containing the operation and the element type parameter. If the key is found (either specific or generic) in the rule table then the operation and parameter are returned otherwise None is returned.

Next I explain the top-down traversal phase which is the cornerstone of the constituency graph creation.

1.3.4 Top down traversal phase

The goal of this first phase is to bootstrap a partial constituency graph starting from a given dependency graph and a rule table. The CG is created as a parallel graph structure through the process of breadth-first traversal on DG edges as described in Algorithm 3. The rewriting of the DG graph into a partial CG is performed as follows. DG nodes, starting from the root, are traversed top-down breadth first order and for each visited node apply the creation or extension operation as provided in the rule table.

Algorithm 3: Partial constituency graph creation by top down traversal

```
   input  : dg (the dependency graph), rule table
   output : cg (the constituency graph)
 1 begin
 2     create the cg with a root node
 3     make the cg root node aware of the dg root node
 4     for edge in list of dg edges in BFS order:
 5         rule ← find in rule table the rule for current edge context
 6         operation ← get operation from the rule table
 7         element type ← get the parameter from the rule table
 8         constituency stack ← get the constituency stack from the tail node of the
             current dg edge
 9         cg pointer ← get the CG node from the constituency stack
10         children ← all child nodes for the current dg edge
           // constructing or extending the cg with a new node
11         execute the operation on cg given element type, cg pointer and children
12     return cg
13 end
```

First the CG is instantiated and an empty root node is created within. Also, the root node is made aware of the root node in DG as described in Section 1.3.2. Then

the DG is traversed in breadth first order (BFS) starting from the root node and as each DG edge is visited an operation is chosen based on the edge type and POS of the connected nodes (Lines 7 - 10). The Line 5 of the algorithm is responsible for looking up and selecting the operation from the rule table as described in Algorithm 2. Then the creative operation is executed with the established parameters: dependency successors (children), a constituent node parenting the newly created one (cg pointer), element type which is chosen from the rule-table together with the operation.

In Python function are first class objects allowing objects to be called (executed) if they are of callable typed. This duality allows storing the functions directly in the rule table and then, upon lookup they are returned as objects but because they are also callable these objects are executed with the expected set of parameters based on their function aspect. The possible operation have been already explained in Section 1.3.3: *extend current* and *new (sibling) constituent*. Next I present the pseudo-code for each of the operations. Note that these operations do not return anything because their effect is on the input cg and dg.

Extend constituent. Algorithm 4 outlines the functionality for extending current cg pointer. It does two main things. It increases the span of an already existing CG node over more DG nodes and makes them, concomitantly, aware of each other.

Algorithm 4: Extend a constituent with DG nodes

```
   input : cg pointer, children, element type, edge, dg, cg
 1 begin
       // handling special relations prep and conj
 2     if prep∨conj in edge relation:
 3         free nodes ← find in dg the free nodes refereed in the edge relation
 4         create new node with marker function under the cg pointer with free
             nodes as children
 5         children ← children & free nodes
       // making the children and cg pointer aware of each other
 6     for node in children:
 7         constituency stack ← the constituency stack of the node
 8         push the cg pointer to the constituency stack
 9         span ← constituent span of the cg pointer
10         extend the span with current node
11 end
```

If, however, the edge relation is a conjunction or a preposition then the children list is extended with the free nodes that stand for the preposition or conjunction in place and eventually neighbouring punctuation marks (line 3).

This exceptional treatment is due to the fact that *prep* and *conj* relations are always specialised by the preposition or conjunction in place. For details on this aspect of Stanford Dependency Grammar please refer to Section **??**.

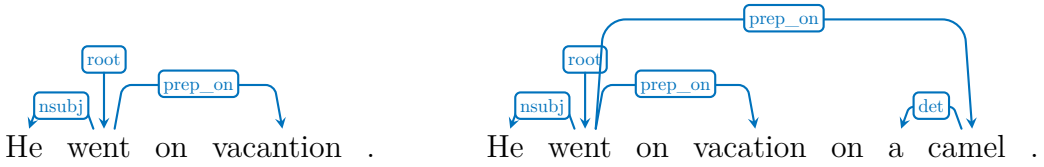

Fig. 1.39 Challenging free nodes

Figure 1.39 exemplifies easy (on the left) and more difficult cases (on the right) of free node occurrence. The challenge in the second figure comes from the fact that there may be two suitable free nodes for the edge *went-prep_on-camel*. Another challenge type in resolving free nodes is when the preposition is a multi word construction.

Once all the free DG nodes are found, a new cg node is created with *Marker* element type spanning over them (line 4). Then the free nodes are included into the list of children and all together are made aware of the current cg pointer and vice versa.

Create new constituent. The Algorithm 5 is a operation that inserts into CG a new node/constituent (line 4). The new constituent is created as a child of a pointed CG node with the element type extracted from the rule table together with the operation. Once the cg node is created, it is extended with the children nodes as described in Algorithm 4 above.

Note that only the functional element is assigned to the freshly created constituent. It's class is added in the second phase of the creation algorithm. This is due to the fact that a function can be filled by units of several classes. The bottom up traversal provides a holistic view on the constituency of each unit giving the possibility to assign a class accordingly. For details see the Chapter **??**.

A variation of the *create new* is *create sibling* outlined in the Algorithm 6. It sets the newly created constituent as a sibling of the current one and not as a child. This will make the new constituent a child of current cg pointer's parent.

Algorithm 5: Creating new child constituent

```
   input  : cg pointer, children, element type, edge, dg, cg
1  begin
2  |  node ← new Constituent
3  |  type(node) ← element type
4  |  add to cg new edge (cg pointer, node)
   |  // invoking the Algorithm 4
5  |  extend cg pointer with children of edge
6  end
```

Algorithm 6: Creating new sibling constituent

```
   input  : cg pointer, children, element type, edge, dg, cg
1  begin
2  |  cg pointer ← get the parent of cg pointer
   |  // invoking the Algorithm 5
3  |  create new constituent to an updated cg pointer
4  end
```

1.3.5 Bottom up traversal phase

Chapter **??** explains that each constituent must specify the unit class and the element it is filling within parent unit. The first phase of the algorithm achieves creating most of the constituents and assigns each unit functional elements derived from the dependency graph. The constituency graph misses, however, the unit classes and the syntactic head nodes. The second phase complements the first one by fulfilling two goals: (a) creation of constituents skipped in the first phase and (b) class assignment to the constituent units.

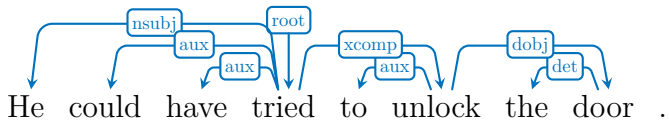

Fig. 1.40 The dependency graph before the first phase

The Figure 1.41 depicts and example CG generated in the 1^{st} phase with dotted lines representing places of the missing constituents.

The missing constituents are the syntactic heads for all units. The clause, besides the Main Verb element which si the syntactic head of the unit, also misses the Finite, Auxiliary elements. Determining these functions strongly depends on the place within a unit and syntagmatic order in which units occur.

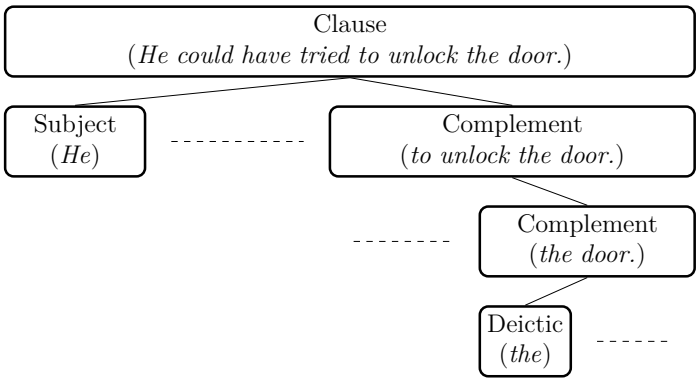

Fig. 1.41 Constituency graph after the top down traversal missing the head nodes

The class membership of constituent units is decided based on three informations available within each constituent: (a) part o speech of the head dependency node (b) element type of the constituent and (c) presence or absence of child constituents and their element types. The corresponding constraints are listed in Table 1.5. The first column carries the unit class (to be assigned), the second and third columns enumerate part of speech and element types that constituents might fill. The Second and third column enumerations are exclusive disjunctive sets (S_{XOR}) because only one may be selected at a time while the list in last column is an open disjunction (S_{OR}) because any of child elements may be present. The last column is an enumeration of what child constituents might the current one have. The *n/a* means that information is unavailable.

Class	*POS of the Head DG Node (XOR)*	*Element Type (XOR)*	*Child Constituents (OR)*
Clause	VB*	Subject, Complement, Qualifier, n/a	Subject, Complement, Adjunct, n/a
Prepositional Group	CD, NN*, PR*,WP*, DT, WD*	Complement, Qualifier, Adjunct	Marker, n/a
Nominal Group	CD, NN*, PR*,WP*, DT, WD*, JJ, JJS	Subject, Complement	Deictic, Numeral, Epithet, Classifier, Qualifier, n/a
Adjectival Group	JJ*	Complement, Epithet, Classifier	Modifier, n/a
Adverbial Group	RB*, WRB	Adjunct	Modifier, n/a

Table 1.5 Constraints for unit class assignment

The Algorithm 7 traverses the CG (not the DG) bottom-up with *post-order depth-first* order (line 2) in order to assign to every visited constituent node a unit class and create the missing child constituents.

Algorithm 7: Creating the head units and assigning classes

```
   input  : cg, dg
 1 begin
 2     for node in list of cg nodes in DFS post-order:
 3         head POS ← get POS of the corresponding dg node
 4         element type ← get assigned function from node
 5         children ← get node children
           // assigning classes
 6         class ← find class based on head POS, element type and children
 7         assign node the class
           // creating the rest of the units
 8         if node is not a leaf:
 9             create the remaining elements under the node
10 end
```

As mentioned above in Section 1.3.2, CG and DG are tight coupled which means that each CG node spans over a set of DG nodes. In this stage, traversal over the CG nodes is equal to traversing groups of DG nodes at each step. This creates focused mini context suitable for resolving the unit class and the missing elements out of the DG chunks.

When assigning unit class the following informations are considered: (a) part of speech of the head DG node that triggered node creation in the first phase (head POS), (b) the assigned element type (element type) and (c) element type of each direct child (children of node). Lines 6 to 7 assign classes according to conditions provided in Table 1.5.

The CG nodes corresponding to non-modal verb DG nodes are assigned clause class. This rule corresponds to the one main verb per clause principle discussed in Section **??**. This approach however does not take into consideration elliptic clauses and they need additional resolution that is considered for the future works and can be overcome by an *ellipsis resolution mechanism*, similar to the one for *null elements* described in Chapter **??**.

The second part of algorithm creates head nodes for every non leaf constituent and in case the node is a clause then it also creates the clause elements such as: Finite, Auxiliary, Main Verb, Negator and Extension.

After the second stage, all the DG nodes must be covered by CG nodes. Moreover the CG nodes build up to a constituency graph that at this stage is always a tree. Provided the class and element type the nodes are ready to be enriched with choices from systemic networks described in the next chapter.

1.4 Discussion

In this Chapter is described in detail a set of transformations on DGs from Stanford parser and a tree rewriting algorithm. The DG transformations are there to correct known errors in Stanford parser version 3.5.1 and to adjust treatment of certain linguistic features such as copulas, conjunction and others. The final section of the chapter describes how the DG is rewritten into a CG using create and extend operations upon graph traversal and tight coupleing between the CG and DG nodes.

There are state of the art algorithms for graph rewriting with proven efficiency. The current work does not intend to be neither generic nor efficient rewriting algorithm but rather explore the process by which a DG can be rewritten into a SFL CG. In the future, to increase the speed performance, the current algorithm would have to be rewritten using for example a graph programming language or a graph rewriting formalism building.

Now that it has been described how to construct the the systemic functional constituency graph, the backbone, we can turn our attention to fleshing out this backbone with systemic features selected from system networks. Or, as we will see, preselected and bundled into graph patterns in order to be attached in batches once the graph pattern has been matched.

Chapter 2

Enrichment of the constituency graph with systemic features

Semantic Role labelling (SRL) is a well established task in the mainstream computational linguistics. Transitivity analysis is the SFL counterpart for SRL and constitutes the focus of this chapter.

The general approach is to first account for covert syntactic constituents and after assign process types and participant roles according to a given resource. In current case the account for empty constituents is implemented as described in Government and Binding Theory (GBT) (Haegeman 1991) and the employed verb database is called Process Type Database (PTDB) (Neale 2002).

Compared to SRL task, where the majority of implementations use probabilistic models trained on an annotated corpus, I employ a static data base based on which I assign a set of configurations that may be the case, or what are the possibilities, rather than the best single guess. For this reason I use the preparatory step of identifying covert constituents (i.e. Null Elements) which reduces the number of possible assignments taking the analysis close to the goal of a single "correct" configuration.

2.1 Enrichment with MOOD features

In this stage the CG nodes are assigned features from system networks. This is achieved by visiting each CG node in a bottom-up order and based on the node class and/or element function (also refereed to as triggers) the relevant system networks are being activated and executed. The relevancy criteria is established by system's precondition set. Each network has one or a set of selector functions associated to it and when the network is activated then the selector function is executed returning the systemic

choices based on the visited node context in the CG graph and features (if any already assigned).

Table 2.1 associates a list of triggers to a list of systemic networks. Table 2.2 associates systems with the choice making functions. Most of the selection criteria have been first implemented as hard-coded Python functions and latter transformed into the Graph patterns with update operations (see Section **??** describing pattern based operations).

Next I describe the enrichment algorithm and then treat some of the selection functions for some of the system networks.

Algorithm 8: Enriching CG with systemic features implemented as custom methods

```
   input  : cg, dg
1  begin
2      for node in list of cg nodes in DFS postorder:
3          for network in activated networks for the node:
4              selector function ← the selector function associated to the network
5              element type ← if any get the selector function parameter
6              systemic choices ← execution result of selector function for node, cg
                 and element type
7              add the systemic choices to the node
8  end
```

The Algorithm 8 shows how to enrich CG nodes with systemic choices using hard-coded selector functions (in the future, these hard coded selector functions will be replaced by verifying realization statements). The outer loop is a bottom-up iteration over the CG nodes in depth first (DFS) post-order. The inner loop iterates over the networks activated by the focus node. Then a lookup in the Table 2.2 returns the function for choosing features from the system network and eventually a parameter (if the function requires it). This technique, involving a mapping table from SN to functions, is similar to the one in CG creation phase (Algorithm 3). The line 5 executes the selection function returning a set of choices. Then the last line assigns the choices to the node in focus.

Currently implemented networks are outlined in Table 2.1 as associations to the node class or function. The left column represents activation triggers and the right column represents an ordered list of systems.

The *dictionary_lookup* function is a type of *naive backwards induction* (Algorithm **??**). It checks whether the word(s) of the *cg_node* are in a dictionary with preselected

Key	*System Activation Order*
clause	POLARITY,VOICE, AGENCY, MOOD TYPE, INDICATIVE TYPE, DEICTICITY, TENSE, MODALITY
nominal	PERSON, ANIMACY, GENDER, NUMBER
deictic	DETERMINATION
predeictic	DETERMINATION
thing	PERSON, ANIMACY, GENDER, NUMBER
possessor	PERSON, ANIMACY, GENDER, NUMBER

Table 2.1 System activation table depending on unit class or element type

System Name	*Python Function*	*Additional Parameter*
POLARITY	check_polarity	
VOICE	check_voice	
AGENCY	check_agency	
MOOD TYPE	mood_type	
DEICTICITY	check_deicticity	
TENSE	check_tense	
MODALITY	check_modality	
DETERMINATION	dictionnary_lookup	determination_dict
PERSON	dictionnary_lookup	person_dict
ANIMACY	dictionnary_lookup	animacy_dict
GENDER	dictionnary_lookup	gender_dict
NUMBER	check_number	
MOOD ADJUNCT TYPE	dictionary_lookup	mood_adjunct_dict

Table 2.2 Mapping systemic networks to selection functions

features. It is the only function that requires a parameter which is the lookup dictionary. An example of dictionary is presented in the Table 2.3

2.2 Creation of the empty elements

Sometimes a semantic participant is not mentioned (is elided) in a clause. It happens for one of two reasons: the participant mentions are contextually implicit and usually located outside the clause borders. So when they are implicit the resolution is contextual and required discourse structure awareness. This task is out of the current scope and constitutes a research field on its own.

Algorithm 9: Dictionary lookup selector function

```
   input  : cg, dg
1  begin
2      for node in list of cg nodes in DFS postorder:
3          for network in activated networks for the node:
4              selector function ← the selector function associated to the network
5              element type ← if any get the selector function parameter
6              systemic choices ← execution result of selector function for node, cg
                and element type
7              add the systemic choices to the node
8  end
```

Lexical item	*Feature*
a	partial-non-selective-singular
all	positive-plural
either	partial-singular
that	non-plural

Table 2.3 Dictionary example for the DEICTICITY system network

However, when the participants are missing because they are syntactically recoverable in the sentence then they are inferred from the structure and reference nodes are created for the missing elements. This phenomena are described in GB theory specifically the *Control and Binding* of empty elements (Haegeman 1991) which has been introduced in Section **??**. The *reference constituents* are important for increasing the completeness of semantic analysis by making the participants and their afferent label explicit.

This stage is particularly important for semantic role labelling because usually the missing elements are participant roles (theta roles) shaping the semantic configuration. The most frequent are the cases of *control* where the understood subject of a clause is in the parent clause like in examples 16 18 where *Subj* is a pronominal subject placeholder.

(16) Piotr is considering whether [*Subj* to abandon the investigation].

(17) Susan promised us [*Subj* to help].

(18) They told you [*Subj* to support the effort].

There are also movement cases when a clause constituent receives no thematic role in higher clause but one in lower clause. The other case is of the non-overt constituents that are subjects in relative clauses and refer to head of the nominal group. This part

of the algorithm is set up to detect cases of *null elements* as described in the Chapter ch:gbt and create placeholder constituents for them which are in the next step enriched with semantic roles.

In Section **??** I discuss how to relate GBT to Stanford Dependency Grammar. This section discusses how to relate GBT to Systemic Functional Grammar. Currently the NP traces and PRO subjects are created with a set of graph patterns while the Wh traces are created with an algorithm.

2.2.1 The PRO and NP-Trance Subjects

The *xcomp* relation in DC can be encoded as an CG pattern graph (Figure 2.1) targeting the constituents that are non-finite clauses functioning as complement that have no subject constituent of their own and no "if" and "for" markers (according to generalization **??**). They shall receive a the PRO subject constituent (governed or not) by the parent clause subject.

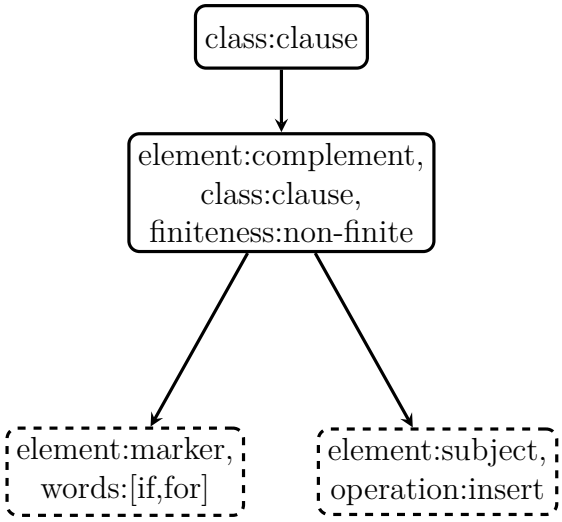

Fig. 2.1 CG pattern for detecting PRO subjects

The generalization **??** reflects criteria for selecting the controller of PRO based on its proximity in the higher clause. The schematic representation of the pattern for obligatory and subject object control is depicted in Figure 2.2 and respectively 2.3. Please note that in case of Figure 2.3 the prepositional complements do not affect subject control in any way since it specifies only the nominal complements this making it complementary to 2.2 with respect to prepositional complements.

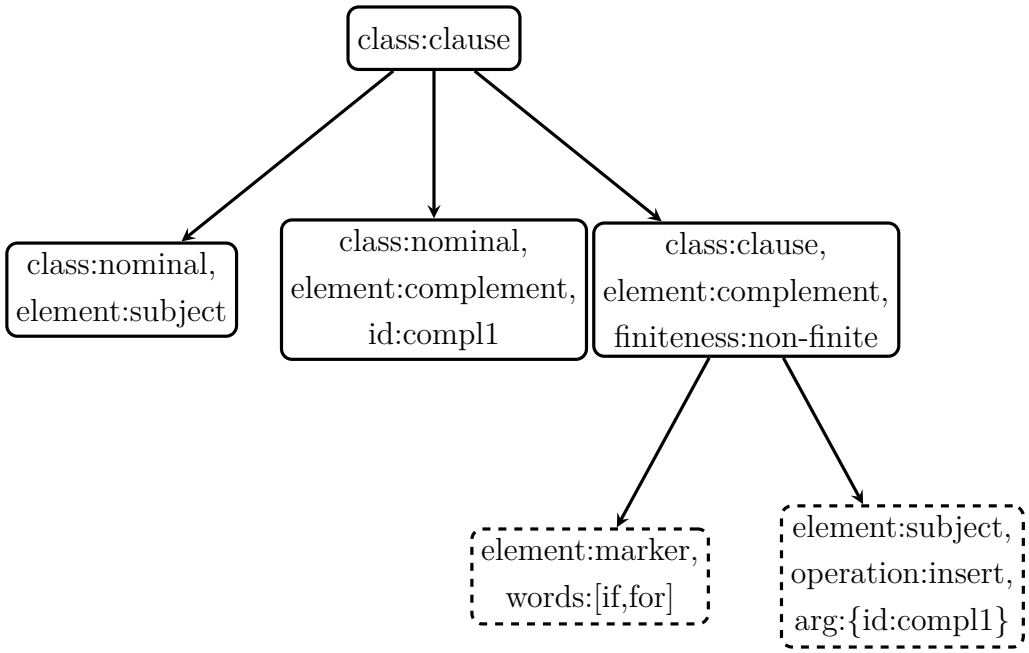

Fig. 2.2 CG pattern for obligatory object control in complement clauses

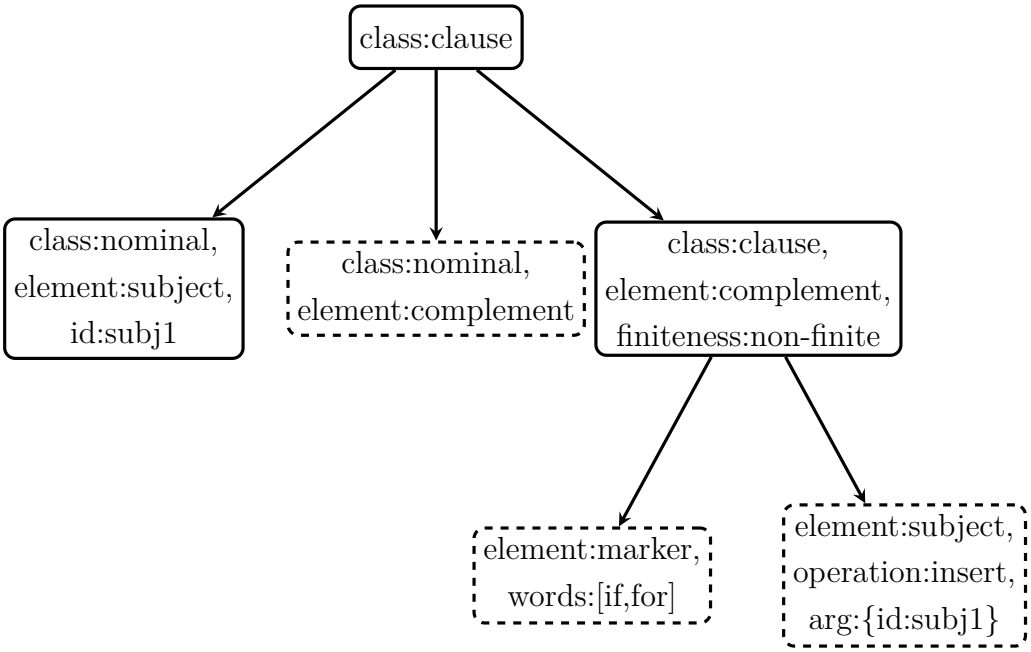

Fig. 2.3 CG pattern for obligatory subject control in complement clauses

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 from Figures 2.2 and 2.3 are applicable.

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) and 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 so this 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. This of course is a dangerous approach as it might lead to unbounded dependencies and so unbounded complexity that needs to be addressed. In the current work, however, the semantic roles are addressed within the clause borders following the principle of one main verb per clause, as will be explained later in the Chapter 2, thus avoiding the above mentioned risk of unbounded complexity.

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 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 labelling) and remove the burdened 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 2.3), (b) identifying the main verbs of higher and lower clauses and corre-

spondingly the set of possible configurations for each clause(by inquiry to process type database(PTDB) described in the transitivity analysis Section 2.3).

If conditions from generalization **??** 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 2.2.1 below.

Generalization 2.2.1. Constituents receiving thematic roles are marked with "thematic-role" label, those that do not receive a thematic role are 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"

According to generalization **??** the PRO is optionally controlled in subject non-finite clauses. Since it is not possible to bind PRO solely on syntactic grounds in the generalization **??** is proposed the arbitrary interpretation.

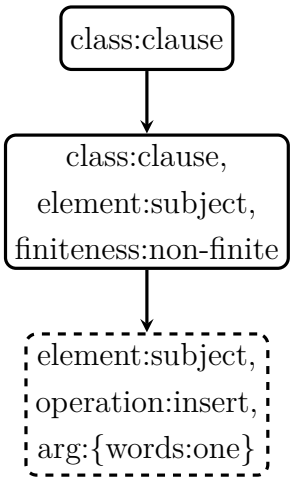

Fig. 2.4 CG pattern for arbitrary control in subject clauses

The pattern for subject control in subject clause is represented in Figure 2.4. This of course is an oversimplification and more rigorous binding rules can be be developed in a future work to cover binding scenarios exemplified in **??**-**??**.

2.2.2 Wh-trances

Creating constituents corresponding to Wh-traces involved a slightly larger number of scenarios and for the pragmatic reasons I have implemented the following algorithm rather than create the set of corresponding graph patterns. Nevertheless in the future

this shall be expressed as graph patterns to be consistent with the general approach. The algorithm pseudo-code below shows how it is done.

Algorithm 10: Creating the Wh-traces

```
   input  : dg, cg
 1 begin
 2     for element in list of Wh-elements in entire cg:
 3         identify the Wh-groups containing the element
 4         identify the syntactic function of the Wh-element within the group
 5         identify the function of Wh-group in the clause
 6         check the number of clauses and which of clauses contains the Wh-group
 7         if more than one clause in cg:
 8             for group in cg: low to high
 9                 if Wh-group is not Subject AND
10                 Wh-group is not in lowest embedded clause:
11                     if Wh-group is Adjunct function:
12                         create Adjunct Wh-trace using dg and cg
13                     else:
14                         create Theta Wh-trance using dg and cg
15 end
```

Algorithm 11: Creating the Adjunct (circumstantial) Wh-traces

```
   input  : wh-group, dg, cg
 1 begin
 2     check the tense and modality for all the clauses
 3     for clause in cg: from the clause of wh-group to lowest
           /* create the adjunct trace in the first clause that has
              non present simple tense                                  */
 4         if clause tense is not present simple:
 5             create Adjunct Wh-trace for Wh-group
 6             return
 7 end
```

Algorithm 12: Creating the Theta (participant) Wh-traces

```
   input  : wh-group, dg, cg
 1 begin
 2     get possible configurations for each clause from the PTDB
       /* check if the higher clause is a projection and and has an
          extra argument                                          */
 3     foreach config in higher clause configurations do
 4         if (config is two role cognition and config takes expletive subject) or
            (config is three role cognition and clause is passive voice):
 5             higher is eligible ← True
 6             break
 7     end
       /* check if the lower clauses might miss an argument       */
 8     foreach clause in lower clauses do
 9         foreach config in clause configurations do
10             if number of clause theta constituents < number of config arguments:
11                 lower is eligible ← True
12                 break
13         end
14     end
15     if higher is eligible and lower is eligible:
16         if higher clause has "that" complementizer:
17             create Object Wh-trace in the lowest clause
18         else:
19             if Wh-group has case:
20                 if Wh-group case is nominative(subjective):
21                     create Subject Wh-trace in lowest clause
22                 else:
23                     create Object Wh-trace in lowest clause
24             else:
25                 create Wh-trace with Subject function and attempt to assign
                    theta roles
26                 if theta roles not successfully assigned in lower clause:
27                     change the Wh-trace to Object function and assign theta roles
28 end
```

Discuss the wh tr
creation algorith

After the null elements are created the constituency graph is ready for the semantic enrichment stage semantic configuration are assigned to each clause. It is described in the next section.

2.3 Enrichment with TRANSITIVITY features

In this section is explained how the parser assigns Transitivity configurations to the constituency graph. Transitivity roughly corresponds to what is known in computational linguistics as semantic analysis of text or *semantic role labelling*. In this task the clause is assigned a semantic frame called configuration in which predicate functions as the process and the participants take frame dependent roles (or functions). The nodes that do not receive any participant role are the adjuncts which act as circumstances.

Because the proposed task goes beyond the syntactic structure it needs to rely on additional external semantic resources. One such resource is the Process Type Database(PTDB) created by Neale (2002). It is a table which listing possible configurations of semantic roles for each verb sense for over five thousands most popular verbs in English. This resource is then integrated into the current parser pipeline to automatically assignment semantic configurations and participant soles. How this is done is covered in the

In the following I will explain the the practical steps of generating Transitivity but not before introducing the PTDB. Then I explain how the automation of the semantic role labelling is performed and why it is important to identify and create references to empty elements.

2.3.1 The Process Type Database

The Process Type Database (PTDB) (Neale 2002) is the key resource in the automatization of Transitivity Analysis. It is the source for creating a set of graph patterns which are then used to enrich the constituency graph as described in the Section 2.1.

PTDB provides information on what possible process types and participants can correspond to a particular verb meaning. The PTDB is a dictionary-like dataset of verbs bound to an exhaustive list of verb senses and the corresponding Process Configuration for each of them.

In her work on PTDB Neale (2002) improved the TRANSITIVITY system of the Cardiff Grammar by systematizing over 5400 senses (and process configurations) for

2750 most popular English verbs. Table 2.4 presents a simplified sample of PTDB content.

verb form	informal meaning	process type	configuration
calculate	work out by mathematics (commission will then,calculate the number of casted votes)	cognition	Ag-Cog + Ph
	plan (newspaper articles were calculated to sway reader's opinions)	two role action	Ag + Cre
catch	run after and seize (a leopard unable to catch its normal prey)	possessive	Ag-Ca + Af-Pos
	fall ill (did you catch a cold?)	possessive	Ag-Ca + Af-Pos
catch (up with)	reach (Simon tried to catch up with others)	two role action	Ag + Ra

Table 2.4 An example of records ins PTDB

2.3.2 Cleaning up the PTDB

The original version of the PTDB available on Neale's personal page is not usable for computational purposes as such. It contains records applying a couple of different notation notations and sometimes informal, human friendly comments which represent noise for the computer programs and cannot be processed as such. In this section I explain now how the original PTDB was transformed in order to be used as a parsing resource.

I focus on three columns which are of interest for the parsing purpose: the *verb form* (1^{st}), the Cardiff grammar *process type* (6^{th}) and the participant role *configuration* (8^{th}) columns. Note that column numbers correspond to the original PTDB structure.

After the transformation the PTDB column descriptions are as described in the Table 2.5.

Next I describe the transformed PTDB and how it be interpreted. For a start, the verb form column contains either base form of the verb (e.g. draw, take), base form plus a preposition (e.g. draw into, draw away, take apart, take away from) or base form plus a phraseologic expression (e.g draw to an end, take on board, take the view that, take a shower). The prepositions are either the verbal particles or the preposition introducing the prepositional phrase complement. Prepositions often influence the process type and the participant configuration. So they are good cues to be considered

Column	Original	Modified
1/A	Form	Form
2/B	*Occurences of form*	*Occurences of form*
3/C	COB class (& figure where possible)	COB class (& figure where possible)
4/D	meaning descrition	meaning descrition
5/E	Occurences in 5 million words	Occurences in 5 million words
6/F	*Cardiff Grammar feature*	*Cardiff Grammar process type (reindexed/renamed)*
7/G	Levin Feature	*Cardiff participant feature*
8/H	*Participant Role Configuration*	*Cardiff participant feature (extra)*
9/I	Notes	Levin feature
10/J		*Participant Role Configuration*
11/K		Notes

Table 2.5 The table structure of PTDB before and after the transformation

during the semantic role assignment. The verb forms that have the same process type and configuration but different prepositions often are grouped together delimited by a slash "/" (e.g. draw into/around, take off/on) or if optional (i.e. coincide with the meaning of the verb base form without any preposition) they are placed in round brackets "()" (e.g. flow (into/out/down)).

The process type column registers one feature in the PROCESS-TYPE systemic network depicted in Figure 2.5.

The participant configuration column contains the sequence of participant type abbreviations joined by plus sign "+" (e.g. Ag + Af, Em + Ph, Ag-Cog + Ph). The order of participants corresponds to the Active voice in Declarative mood also called the *canonical form of a configuration*. Originally the configurations contained the "Pro" abbreviation signifying the place of the main verb/process. As all configurations are in canonical form, the Pro was redundant occurring always in the second position thus had been removed. So the first participant corresponds to the Subject, second to the first complement and third to the second complement. Some participants are optional for the meaning and are marked round brackets "()" (e.g. Ag + Af-Ca (+ Des)) meaning that the participant may or may not be realized.

Not all the records in the original resource fulfill the description above and needed corrections. For example when Neale had doubts during the making of PTDB, she marked uncertainties with question mark "?". Commas "," and and "&" sign are use

with various meaning and inconsistently in all columns. Comments such as “not in Cob” were encountered across several columns.

Some records contain only prepositions listed in the verb form column which actually represent omissions of the main verb which is to be found in the immediately preceding records(s), this have been fixed by pre-pending the verb form to the preposition.

Among the identified verb meanings in PTDB, there are some that do not contain configurations. These records missing a process type and configuration had not been suppressed.

The process type feature column contained originally a second feature which had been removed, representing a compressed version of the participant configuration however it was redundant as the full configuration is registered in the next column.

In PTDB Neale uses a slightly different process type names than the ones adopted in this work. The process type features have been re-indexed and adapted to match exactly the feature labels in the PROCESS-TYPE systemic network (e.g. “one role action” became “one-role-action”). Annexe **??** provides the mapping across the process type versions.

Configuration column is one of the most important ones in PTDB. Checking it’s consistency conform to Fawcett’s Transitivity system revealed the need for some corrections. For example “Af + Af”, “Af-Ca + Pos + Ag”, “Af-Cog + Ph + Ag” are grammatically impossible configurations and were manually corrected to the closest likely configuration “Ag + Af”, “Ag + Af-Ca + Pos”, “Ag + Af-Cog + Ph”.

Other records, judged by the process type, were incomplete. For example instances of two role action registered only one of the role e.g. Af or Ca omitting the Ag participant. These records have been also manually corrected by perpending the Ag, Ag-Af or Cog roles depending on the case.

The “Dir” participant is interpreted as direction is not registered/defined in the Cardiff Grammar. Nevertheless there is a “Des” participant which I believe is the closest match. Therefore all “Dir” occurrences had been changed to “Des”. One may argue that the two have different meanings however grammatically they seem to behave the same (at least in the accounted configurations).

By contrast some process types had been changed from Action into either Locational or Directional because they contained either Loc (location), Des (destination) or So (source) participants which are not found in Action processes unless they function as Adjuncts which are out of the context of the current description.

A note worth making here is that the Locational, Movement specifically but also other spatial verbs are very difficult to assign roles correctly because their participants

are Locations, Directions, Destinations etc. which can also serve as spatial adjuncts. This aspect has not been directly addressed in present work and is reflected by a lower accuracy for these classes of verbs.

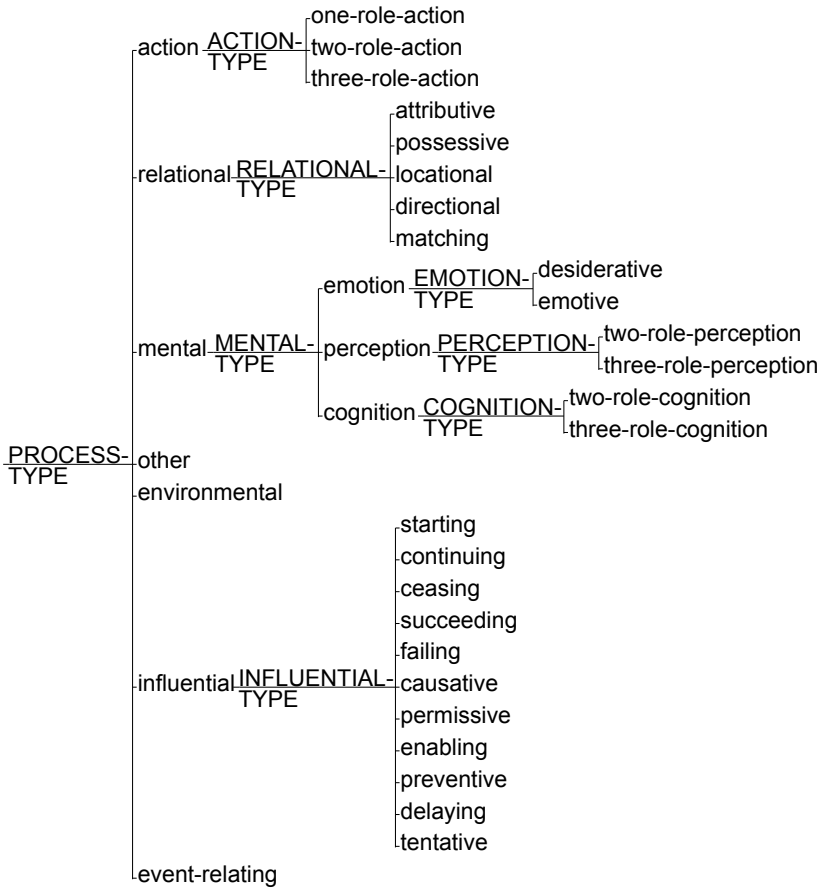

Fig. 2.5 Transitivity System in Cardiff Grammar

The Cardiff features column indicates the process type selected in the TRANSITIVITY system corresponding to one of the top levels depicted in Figure 2.5

2.3.3 Generation of the Configuration Graph Patterns

The configuration pattern graphs are used for CG enrichment executed through graph matching operation described in Section **??**. The PTDB has been cleaned up and normalized to support automatic generation of the *Configuration Graph Patterns* (CGP). These graph patterns represent a constrained syntactic structure that carry semantic features. The latter are to be applied if the graph pattern is identified the sentence CG.

CGPs are generated from the process type and participant configuration columns of the PTDB. Figure 2.6 depicts the prototypical template for generating three role CGP for canonic participant order (indicative mood active voice). This part assigns the Clause constituent Process Type (i.e. configuration type) while Subject and Complement constituents receive participant roles.

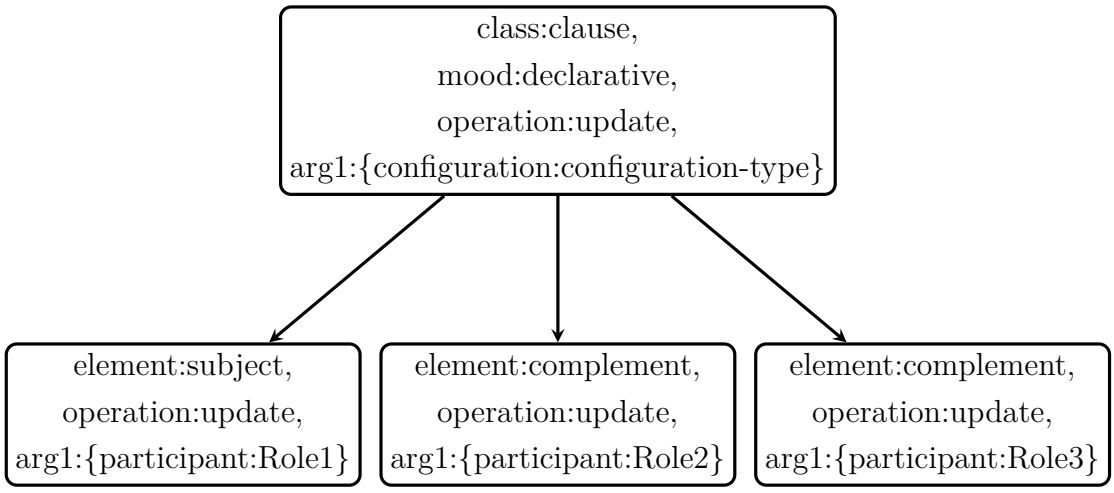

Fig. 2.6 Indicative mood and active voice configuration pattern with three participant roles

Besides the canonic CGP a set of variations are generated for each configurations by transforming the canonic configuration. The variations are function of the process type, participant roles, mood and voice. When each of the variants are supported by the process type and participant configuration the flowing CGP are generated: (1) the declarative active (2) the passive (3) the imperative and (4) Wh-interrogative (Wh-Subj/Wh-obj/Wh-adj especially important for locational and directions processes).

If the configuration accepts passive voice i.e. the first configuration role is not expletive "there" or pleonastic "it" and the last role is not Agent role then both active and passive voice CPG are generated.

The imperative form CGP is generated if the first role of the configuration implies an active animate entity. Roles that accept imperative form are: Agent, Emoter, Cognizant, Perceiver and their compositional derivations (e.g. Agent-Carrier, Agent-Cognizant, Affected-Emoter etc.)

The passive differs from active voice pattern by switching places of the first two roles resulting in the second role matched to the subject function and the first role one the first complement. In the case of imperative the first role as well as the subject constituent are simply omitted.

Algorithm 13 outlines how the CPGs are generated from the PTDB. The CPGs are represented as Python structures and are stored in a Python module. This way the graph patterns are accessible as native structures making it handy to instantiate the graph patterns.

Algorithm 13: Generating the CPGs from the PTDB

```
   input  : PTDB
 1 begin
 2     generate unique set of process type + participant roles
 3     generate unique set os process type
 4     for process type in possible process type:
 5         generate configuration set for given process type
 6         for configuration in configuration set:
 7             generate indicative active pattern graph
 8             if no expletive in configuration:
 9                 if configuration accepts passive:
10                     generate indicative passive pattern graph
11                 if configuration accepts imperative:
12                     generate imperative pattern graph
13                 if Locational process:
14                     generate expletive there pattern graph
15                 if configuration participants may function as Adjuncts:
                       /* the Directional processes varying optional
                          Source, Path and Destination                 */
16                     generate variate role indicative active pattern graphs
17                     generate variate role indicative passive pattern graphs
18                     generate variate role imperative pattern graphs
19                     generate variate role Wh-interrogative pattern graphs
20 end
```

The first two lines of the algorithm synthesize the PTDB by grouping unique configurations for each process type. Then for each configuration of each process type is generated one to three pattern graphs depending on the configuration and process type specifics.

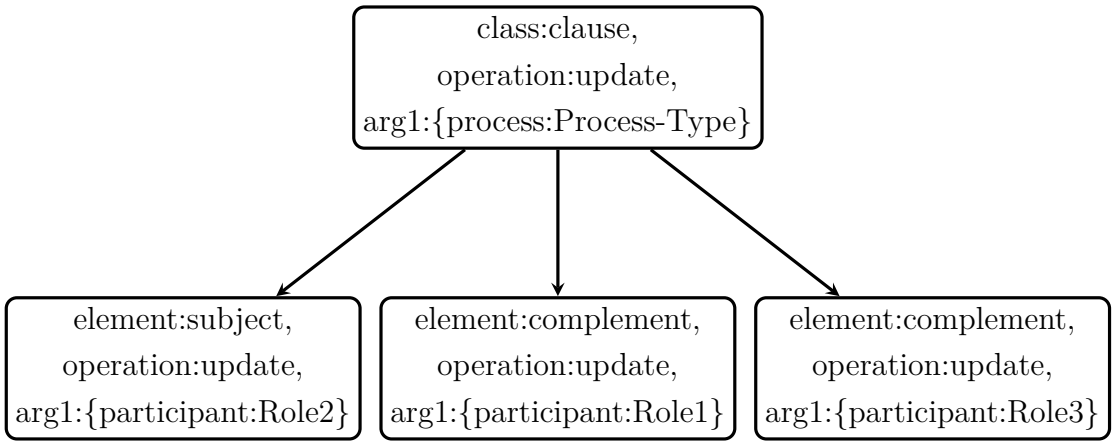

Fig. 2.7 Indicative mood and passive voice configuration pattern with three participant roles

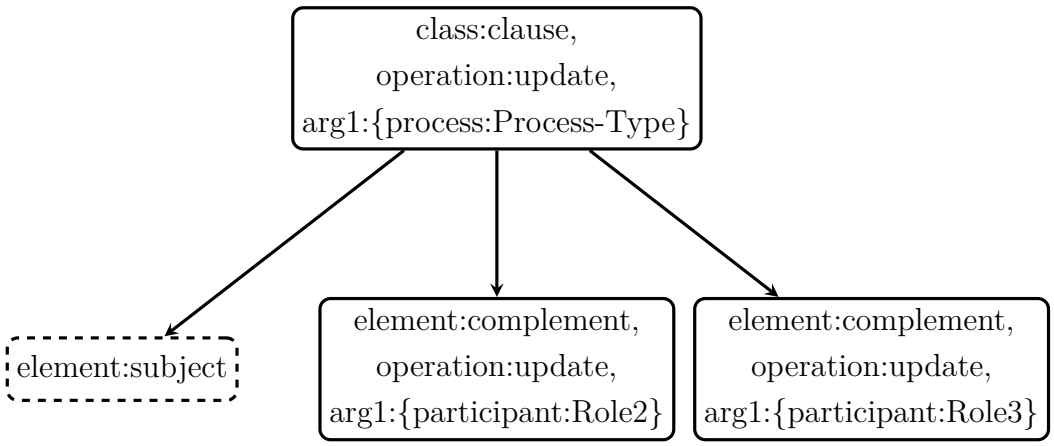

Fig. 2.8 Imperative mood configuration pattern with three participant roles

The indicative mood active voice pattern (depicted in Figure 2.6) corresponding to the canonic form in PTDB is always generated. If the configuration does not contain an expletive and accepts passive voice then the corresponding pattern is generated with first and second roles switched like in Figure 2.7. If the configuration accepts imperatives then also a subjectless pattern graph is generated with first role omitted as depicted in Figure 2.8.

Directional processes are rather a special case. Examples 19 to 21 are equally valid configurations. Example 22 is a generic representation highlighting the optionality of these participants.

(19) The parcel traveled from London[So].

(20) The parcel traveled via Poland[Pa].

(21) The parcel traveled to Moscow[Des].

(22) The parcel traveled (from London[So]) (via Poland[Pa]) (to Moscow[Des]).

So in Directional configurations second, third and fourth participants are optional and may occur in any order but at least one of them should be present. Therefore So, Pa and Des participants patterns should be generated for all combinations as presented in the Table 2.6 below.

So	Pa	Des
+	-	-
-	+	-
-	-	+
+	+	-
+	-	+
-	+	+
+	+	+

Table 2.6 Participant arrangements for Directional processes (order independent)

Finally, the CPGs are grouped by process type. This alleviates the burden of selecting the number of patterns to test for a certain clause. The python structure looks as represented in Listing 2.1

Listing 2.1 Configuration Pattern Set as a Python dictionary structure

```
configuration_pattern_set = {
        process_type1 : {
                config1 : [pattern1, panttern2,...],
                config2 : [pattern3, panttern4,...],
        },
        process_type2 : {
                config3 : [pattern5, panttern6,...],
                config4 : [pattern7, panttern8,...],
        }
        ...
}
```

Role	Prepositions
Des	to,towards,at,on,in,
Ben	to,for,
Attr	as,
Ra	on,in,
So	from,
Pa	through,via,
Loc	in,at,into,behind,in front of, on
Mtch	with,to,
Ag	by,
Ph	about,
Cog	to

Table 2.7 Prepositional constraints on participant roles

2.3.4 Transitivity parsing algorithm

Transitivity enrichment is performed on the constituency graph after it has been enriched with Mood features and the null elements had been identified and appended.

Algorithm 14: Transitivity parsing

```
input : cg, dg
begin
    for clause in clauses in mcg:
        select from PTDB candidate process types and configurations
        filter configurations fitting the clause
        for config in valid possible configurations:
            filter pre-generated pattern graphs of the config fitting the clause
            for pattern in filtered pattern graph set:
                enrich clause using pattern and mcg filtered by role constraints
end
```

Algorithm 14 outlines the semantic parsing process implemented in the current parser which is a cascade of three loops.

The first loop iterates through clauses in the mood constituency graph and for each the candidate process types are identified by considering: (a) the main verb, (b) the prepositions connected to it (either prepositional particles, or prepositions introducing a complement or adjunct prepositional phrase listed in Table 2.7) or (c) phrasal expressions such as "take a shower" which are explained in Section 2.3.1.

The second loop iterates through the candidate configurations for each candidate process type and selects the graph patterns that shall be matched to the current clause.

Then iteratively, each of the retrieved graph patterns (in the third loop) are applied to the clause graph. Line 7 enriches CG nodes with new features of the pattern graph each time they are successfully matched. Before enrichment the CG nodes are checked against an additional set of condition (captured by Algorithm 15) which were omitted in the pattern graph. These conditions may prevent the enrichment even if the pattern has been identified.

Algorithm 15: Participant Role constraint check if a role is not illegal for constituent

```
   input  : node, role, dg
1  begin
       /* Cog, Em and Perc must be animates                          */
2      if role is Cog or Em or Perc:
3          check that the node has animate feature
       /* clauses can only be phenomenas                             */
4      if node is a clause:
5          check that role is Ph only
       /* prepositional phrases can only start with certain
          prepositions for a role                                    */
6      if node is a Prepositional Phrase:
7          get the list of allowed prepositions for the role
8          check if the prepositional phrase starts with any of the allowed
            prepositions
9  end
```

The Transitivity parsing results in multiple process configurations being assigned to clauses introducing uncertainty. This uncertainty, laying within the reasonable limits, is not dealt with by the current parser but shall be further resolved through deeper semantic and pragmatic means.

2.4 Discussion

This chapter describes how the semantic role labelling from Transitivity system network is performed on top of the constituency graph.

First null elements are identified and filled with proxy constituents. This process ensures higher accuracy of semantic role assignments from the configuration patterns in the second step.

The configuration patterns have been generated from the PTDB - a verb database accounting for Transitivity patterns in Cardiff grammar. In order to generate these

patterns the PTDB had to be first cleaned up, unified and aligned to the present Transitivity system. Then for each configuration were generated a set of possible patterns varying based on mood, voice, process type and participants.

Finally the semantic role labelling step employs the same pattern based enrichment mechanism as the one used in Section 2.1

The Algorithms can be improved a great deal by externalization of constraints and conditions. Some of them however are too complex to check to be completely removed but in the next iterations the tendency should definitely be towards higher abstraction and separating the grammatic constraint as realization rules from the code.

Chapter 3

The Empirical Evaluation

3.1 Evaluation settings

There are several well established annotated corpora such as Penn or Prague Tree-Banks for phrase structure, dependency and other grammars that serve as training and evaluation data sets. To date I am not aware of any open corpus annotated with Cardiff or Sydney grammars that would play the role of the gold standard for SFL parsers.

To overcome this problem I decided to create own evaluation corpus comprising two pars. First was created together with Ela Oren covering basic Sydney Mood constituency structure. The Second part is the PhD dataset of Anke Shultz covering Cardiff Transitivity analysis.

The parser is evaluated by comparing manually annotated text to automatically generated analyses. The datasets have not been developed for the purpose of evaluating current parser however they are compatible and can be adapted to evaluate identification of constituent boundaries and how is being assigned a limited set of features. The first dataset represents Transitivity analysis of 31 files spanning over 1466 clauses and 20864 words created by Anke Schultz as part of her PhD project and is mainly used to evaluate semantic parsing. The second dataset represents Constituency and Mood analysis of blog text spanning over 988 clauses and 8605 words. This dataset was created by Ela Oren as a part of her project on analysing texts of people with obsessive compulsive disorder (OCD) and is mainly used to evaluate the syntactic parsing.

Both datasets were created with UAM Corpus Tool authored by O'Donnell (2008a,b). It stores annotations as segments with their corresponding start and end positions in the text together with set of features(selected from a systemic network) attributed to

that segment without any proper constituency relations among segments. Below is a XML representation of an annotation segment for Transitivity.

```
<segment id="4" start="20" end="27" features="configuration;
 relational; attributive" state="active"/>
```

To make manual and automatic analyses comparable constituency graphs (CG) are transformed into a set of segments the evaluation task being reduced to find matches or near-matches between two segment bundles. This task is almost the same as know problem in computer science called Stable Marriage Problem. Because the problem formulation slightly differs from the standard one I implement Gale-Shapley algorithm (Gale & Shapley 1962) one of the most efficient algorithms with the corresponding extra requirements.

The standard stable marriage problem is formulated as such that there is a group of men and a group of women and each individual from each group expresses their preferences for every individual from the opposite group as an ordered list. The assumption is that every individual expressed preferences as an ordering of individuals from the opposite group from the most preferred one to the least preferred one. Thus the preferences must be *complete* and *fully ordered*. None of the last two constraints could be fulfilled in the context of the present evaluation and I explain why by exploring and explaining the identity criteria between two segments.

Leaving aside the features and focusing only on the segment span we can say that two segments are identical when their start and end indexes are identical. However it is not always the case and there are situations when we would like to consider two segments identical even if their spans differ.

Firstly, there is an encoding problem. Some text files start with a BOM or contain non printable characters resulting in a technical problem of mismatch between how UAM Coprus Tool reads files and how the parser evaluation module reads them introducing two types of text-segment mismatch: (a) shifted text and (b) shrieked/enlarged text span. This problem has been overcome by an extra module which indexes CG words and sentences (as word collections) in the raw text. This helps re-indexing the parse segments in such a way that their text "coincides" with the text in the CT annotations and readjusting their indexes.

Secondly, there is a feature comparison problem. Provided that two segments have identical spans and textual content but two different sets of features. To overcome this problem I force segments to carry only one feature. The consequence are multiple segments with the same span (Figure 3.2) for each feature instead of single segment with multiple features (Figure 3.1).

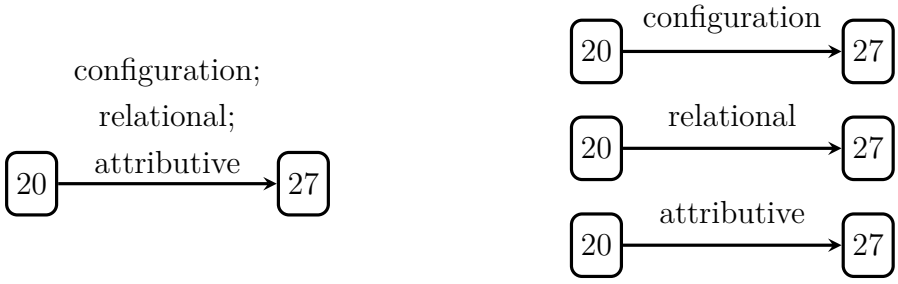

Fig. 3.2 A set of segments with single features

When each segment contains exactly one feature the evaluation can be focused on one or a set of features of interest by selecting segments that contain exactly those.

3.2 Syntactic Evaluation

The syntactic evaluation is based on the corpus produced together with Ela Oren focused on Constituency and clause Mood analysis. The text represents a blog article of a person diagnosed with OCD writing about the challenge of overcoming OCG.

I first present few differences in how corpus has been annotated as compared to parser output, then present segmentation statistics followed by featured statistics.

In the corpus, punctuation marks such as commas, semicolons, three dots and full stops are not included into the constituent segments while the parser includes them usually at the end of each segment. Neither the conjunctions (and, but, so , etc.) are included into the corpus segments because they are considered markers in the clause/group complexes. The parser, on the other hand, includes them into the segments. Moreover the conjunct spans differ as well. Instead of being annotated in parallel segments as represented in Figure 3.3 they are subsumed in a cascade from the former to the latter as depicted in Figure 3.4.

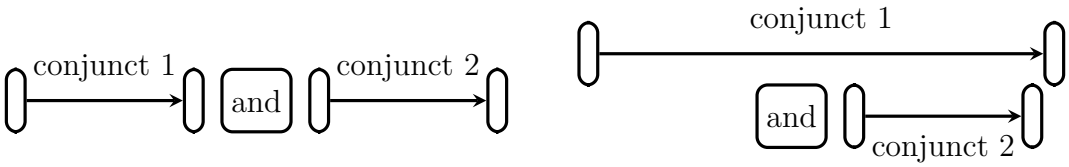

Fig. 3.4 Conjuncts annotated as subsumed segments

Another difference is the way verbs and verb groups are handled. In the parser is implemented the Sydney Grammar's Finite and Predicate constituents while the corpus is annotated according to Cardiff Grammar with Main Verbs, Finite, Auxiliary, Negation Particles etc. as explained in Section **??**.

The above described differences in annotation rules are inevitably reflected in segmentation differences. Let's compare now the corpus and parser segmentation. I use geometric distance ($d = \sqrt{\Delta start^2 + \Delta end^2}$) to measure the difference between the segments matched with Gale-Shapley algorithm. Figure 3.5 represents distribution of matches according to the distance. 71.11% of the segments have identical spans while the rest of 28.8% have minor to major differences of which 19% are distanced 1-5 characters due to differences in (a) punctuation, (b) conjunction and (c)verbal group treatment described above. The rest 9.9% of long and very long distances (6-182 characters) are due to differences in verbal group treatment, subsumed conjuncts (clause and group conjunctions) and erroneous prepositional phrase attachment (treated as Qualifier instead of Complement/Adjunct or vice versa).

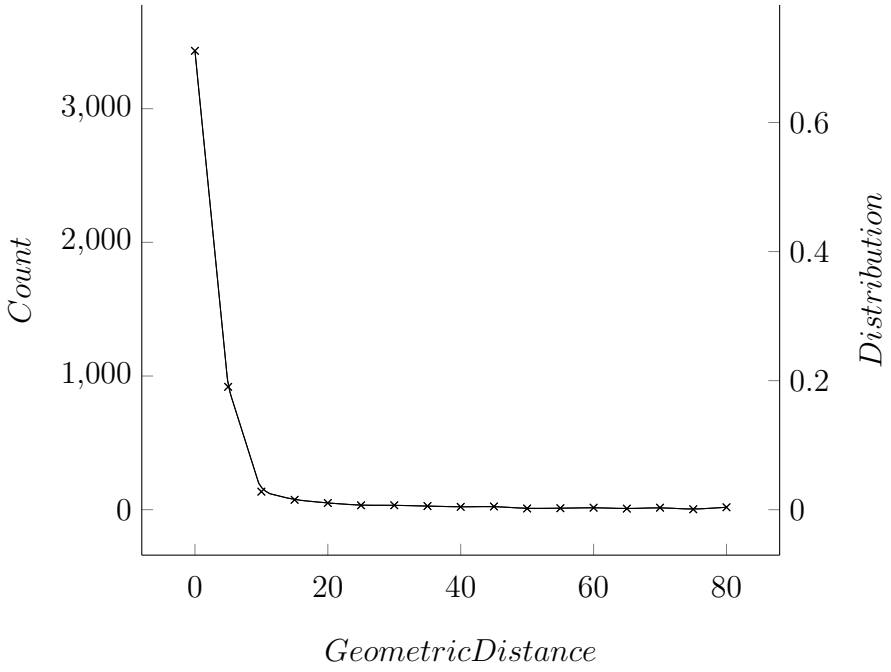

Fig. 3.5 Segment distance distribution

Next I present evaluation data for systemic features and systems overall annotated in the corpus and parser. In the tables below are presented the following statistics for features and systems (in upper caps): number of segments in corpus containing the feature (M/N) percentage of the in corpus (M/%), number of segments generated

by the parser (A/N) and corresponding percentage (A/%), precision, recall and F_1 measures.

Table 3.1 contains evaluation statistics for the rank system. In this evaluation only group and clause level features have been taken into account excluding those at the word rank which are missing in the corpus. Clause and group constituents are identified with 0.88 and 0.9 F_1 scores.

The marker feature is used for functional words such as prepositions, verbal prepositional particles, coordinating and subordinations conjunctions. However the prepositional phrases in the corpus do not contain the marker segment as the parser generates and only conjunctions and verbal particles are marked leading to low precision of 0.38. Overall the rank constituents are identified with 0.82 F_1 score.

Name	*M/N*	*M/%*	*A/N*	*A/%*	*Prec*	*Rec*	F_1
RANK	2455	26.922	4052	36.156	0.716	0.902	0.762
clause	529	21.548	545	13.450	0.875	0.902	0.888
group	1699	69.206	1760	43.435	0.886	0.918	0.902
marker	224	9.124	396	9.773	0.389	0.688	0.497
word	3	0.122	1351	33.342	-	-	-

Table 3.1 Rank system evaluation statistics

The clause elements are systematized in Interpersonal function system whose evaluation statistics are contained in the Table 3.2. Subjects and Predicators have the highest F_1 scores of 0.889 and 0.888. Indirect objects have been perfectly identified but they are very few and do not have a statistical significance. Adjuncts and Complements (direct objects) have a lower precision and higher recall scoring 0.628 and 0.661 F_1. This is in part due to the fact that the parser annotates prepositional phrases that follow a predicate as both Complement and Adjunct being unable to syntactically distinguish between the two and this task being overtaken during the semantic analysis. The Finite elements are distinguished in the corpus only when they stand alone thus are not conflated with the predicator as in Example 23. But when the finite is conflated with the Predicator as in Example 24 it has not been annotated in the corpus but it should be and further corrections of the corpus shall be implemented. For this reason there is a big difference between precision and recall 0.172 and 0.980 because most Finites in corpus have been identified by the parser and most of them are missing from the corpus.

(23) He may (Finite) go (Predicator) home latter.

(24) He already went (Finite/Predicator) home.

Name	*M/N*	*M/%*	*A/N*	*A/%*	*Prec*	*Rec*	F_1
INTERPERSONAL-FUNCTION	1681	18.434	2793	24.922	0.659	0.910	0.726
finite	150	8.923	855	30.612	0.172	0.980	0.293
adjunct	266	15.824	358	12.818	0.547	0.737	0.628
subject	389	23.141	439	15.718	0.838	0.946	0.889
complement-direct	347	20.642	594	21.267	0.524	0.896	0.661
complement-indirect	2	0.119	2	0.072	1	1	1
predicator	527	31.350	545	19.513	0.873	0.903	0.888

Table 3.2 Mood clause elements evaluation statistics

In Table 3.3 are presented evaluation statistics for Group Type system meaning how well are identified grammatical classes at the group level. Nominal and Verbal groups are parsed with 0.9 precision followed by prepositional, adverbial and adjectival groups with 0.72, 0.6 and 0.55 scores.

Name	*M/N*	*M/%*	*A/N*	*A/%*	*Prec*	*Rec*	F_1
GROUP-TYPE	1698	18.620	1728	15.419	0.838	0.853	0.845
nominal-group	648	38.163	622	35.995	0.905	0.869	0.887
verbal-group	678	39.929	705	40.799	0.894	0.929	0.911
prepositional-group	120	7.067	124	7.176	0.726	0.750	0.738
adverbial-group	187	11.013	218	12.616	0.606	0.706	0.652
adjectival-group	65	3.828	59	3.414	0.559	0.508	0.532

Table 3.3 Evaluation statistics for group types

3.3 Semantic Evaluation

For the semantic evaluation has been used a corpus created by Anke Schultz for her PhD project annotated with Transitivity features i.e Configuration, Process and Participant which from constituency point of view correspond to the Clause, Predicator and Participants(predicate arguments) to Subjects and Complements cumulated.

Name	*M/N*	*M/%*	*A/N*	*A/%*	*Prec*	*Rec*	F_1
Configuration	1466	-	1833	-	0.736	0.915	0.814
Process	1423	-	1814	-	0.707	0.902	0.791
Participant	2652	-	3398	-	0.732	0.925	0.816

Table 3.4 Transitivity System evaluation statistics

Transitivity analysis is semantic in nature and poses challenges in meaning selection beyond constituent class or function. Current parser does not select one meaning but rather proposes a set of possible configurations for each clause. If top level Transitivity features correspond fairly to the number of syntactic constituents, the more delicate features are parsed with an average of 2.7 proposed features for each manually annotated one.

Name	*M/N*	*M/%*	*A/N*	*A/%*	*Prec*	*Rec*	F_1
CONFIGURATION-TYPE	1466	23.12	1308	7.57	0.542	0.483	0.508
action	359	24.82	453	33.99	0.315	0.409	0.333
relational	546	37.79	445	34.77	0.645	0.530	0.574
mental	452	29.86	318	24.27	0.744	0.530	0.605
influential	81	6.69	91	7.39	0.556	0.519	0.484
event-relating	28	3.48	1	1.85	1.0	0.5	0.667
environmental	0	0	0	0	0	0	0
other	0	0	0	0	0	0	0

Table 3.5 Configuration type evaluation statistics

The semantic evaluation yields surprisingly positive results for relational and metal processes. At the same time, actions would have been expected to score a high accuracy but as seen in the results it is not the case. Despite a high number of them both in the corpus (24%) and in the parses (33%) they seem to be misaligned and perhaps not matched in the process. Further investigation should be conducted to spot the source of such a low score. Next I present a short critical discussion of the overall evaluation.

3.4 Discussion

Further investigation shall be conducted to determine the error types, shortcomings in the corpus and the parser. But beyond the simple improvement to the corpus such as adding the missing Finite elements it would benefit tremendously from a second annotator in order to evaluate reliability of the annotation itself and how much of a gold standard it can be considered for the current work.

Also the corpus size is very small and many features are missing or severely underrepresented and bear no statistical significance. For example event relating, environmental and other processes are missing from the corpus. Also the number of other features that the parser provides are missing from the manual analysis and it would be interesting to add some of them to study how varies the accuracy distribution as the delicacy of features increases.

Since for both syntactic and semantic annotations there is only a single author annotation available, the results shall be considered indicative and by no means representative for the parser performance. Nevertheless they can already be considered as a good feedback for looking into certain areas of the grammar with considerably low performance in order identify the potential problems.

Chapter 4

Conclusions

The aim of this work is to design a reliable method for English text parsing with Systemic Functional Grammars. To achieve this goal I have designed a pipeline which, starting from a dependency parse of a sentence, generates the SFL-like constituency structure serving as a syntactic backbone and then enriches it with various grammatical features.

In this process a primary milestone the first steps is the creation of constituency structure. Chapter **??** describes the essential theoretical foundations of two SFL schools, namely Sydney and Cardiff schools, and provides a critical analysis of the two to reconcile on the diverging points on rank scale, unit classes, the constituency structure, treatment of coordination, grammatical unit structure, clause boundaries, etc. and state the position adopted in this work.

In order to create the constituency structure from the dependency structure there needs to be a mechanism in place providing a theoretical and a practical mapping between the two. The theoretical account on the dependency grammar and how it is related to SFL is described in Chapter **??**. The practical aspects and concrete algorithms are described in Chapter 1 together with the mapping rules used in the process.

To make clear what are the basic ingredients and how the algorithms are cooked, Chapter **??** introduces all the data structures and operations on them. These structures are defined from a computer scientific point of view emulating the needed SFL concepts. These range from a few graph types, simple attribute-value dictionaries and ordered lists with logical operators. In addition to that, a set of specific graph operations have been defined to perform pattern matching and system network traversals.

Once the constituency structure is created, the second milestone is to enrich it with systemic features. Many features can be associated to or derived from the dependency

and constituency graph fragments. Therefore graph pattern matching is a cornerstone operation used for inserting new or missing units and adding features to existing ones. I describe these operations in detail in the second part of **??**. Then in Chapters 1 and 2 I show how these operations are being used for enrichment of the syntactic backbone with systemic features.

The more precisely graph pattern is defined the less instances it will be matched to and thus decreasing the number of errors and increasing the accuracy. The semantic enrichment is performed via spotting instances of semantic graph patterns. It is often the case that the patterns, in their canonical form, list all the participants of a semantic configuration but in practice, instances of such configurations may miss a participant or two. If applied in their canonical form the patterns will not identify with such the instance. One solution would be to reduce the specificity of the patterns, which leads to a high increase in erroneous applications or populate where possible the covert participants to yield successful matches. It is the second approach that was implemented in this work. To identify and create the covert participants I turned to Government and Binding theory. Two more contributions I bring in this thesis is the theoretical mapping from GBT into dependency structures covered in Chapter **??** and then a concrete implementation described in Chapter 2.

In the last part of the thesis I describe the empirical evaluation I conducted in order to test the parser accuracy on various features. To conduct this evaluation I created together with Ela Oren a corpus using blog articles of OCD patients covering the Mood system and another corpus was provided to my by Anke Schultz covering the Transitivity system. The results show very good performance (0.6 – 0.9 F1) on Mood features slightly decreasing as the delicacy of the features increases. On Transitivity features, the results are expectedly less precise (0.4 – 0.8 F1) and constitute a good baseline for future improvements.

As discussed in the Section 3.4 further investigation shall be conducted to determine the error types, shortcomings in the corpus and the parser. Since for both syntactic and semantic annotations there is only a single author annotation available, the results shall be considered indicative and by no means representative for the parser performance. Nevertheless they can already be considered as a good feedback for looking into certain areas of the grammar with considerably low performance in order identify the potential problems.

4.1 Practical applications

A wide variety of tasks in natural language processing such as document classification, topic detection, sentiment analysis, word sense disambiguation don't need parsing. These are tasks can achieve high performance and accuracy with no linguistic feature or with shallow ones such as as lemmas or part of speech by using powerful statical or machine learning techniques. What these tasks have in common is that they generally train on a large corpus and then operate again on large input text to finally yield a prediction for a single feature that they have been trained for. Consider for example the existing methods for sentiment analysis: they will provide a value between -1 to 1 estimating the sentiment polarity for a text that can be anything from one word to a whole page.

Conversely, there are tasks where extracting from text (usually short) as much knowledge as possible is crucial for the task success. Consider a dialogue system: where deep understanding is essential for a meaningful, engaging and close to natural interaction with a human subject. It is no longer enough to assign a few shallow features to the input text, but a deep understanding for planning a proper response. Or consider the case of information extraction or relationship mining tasks when knowledge is extracted at the sub-sentential level thus the deeper linguistic understanding is possible the better.

Current parser is useful to achieve the latter set of tasks. The rich constituency parses can be an essential ingredient for further goals such as anaphora resolution, clausal taxis analysis, rhetoric relation parsing, speech act detection, discourse model generation, knowledge extraction and other ones.

All these tasks are needed for creating an intelligent interactive agent for various domains such as call centers, ticketing agencies, intelligent cars and houses, personal companions or assistants and many other.

In marketing research, understanding the clients needs is one of the primary tasks. Mining intelligence from the unstructured data sources such as forums, customer reviews, social media posts is particularly difficult task. In such cases the more features are available in the analysis the better. With the help of statistical methods feature correlations, predictive models and interpretations can be conveyed for task at hand such as satisfaction level, requirement or complaint discovery, etc.

4.2 Impact on future research

Pattern graphs and the matching methods developed in this work can be applied for expressing many more grammatic features than the ones presented in this work. They can serve as language for systematizing grammatical realizations especially that the realization statements play a vital role in SG grammars. The graph matching method itself can virtually be applied to any other languages than English. So similar parsers can be implemented for other languages and and respectively grammars.

Linguists study various language properties, to do so they need to hand annotate large amounts of text to come up with conclusive statements or formulate hypothesizes. Provided the parser with a target set of feature coverage, the scale at which text analysis is performed can be uplifted orders of magnitude helping linguists come with statistically significant and grounded claims in much shorter time. Parsimonious Vole could play the role of such a text annotator helping the research on text genre, field and tenor.

4.3 Further work

This section describes improvements of the project that are desirable or at least worth considering along with major improvements that arouse in the process of theoretical development and parser implementation.

4.3.1 Verbal group again: from syntactically towards semantically sound analysis

The *one main verb per clause* principle of the Cardiff school that I adopted in this thesis (briefly discussed in Section **??**) provides a basis for simple and reliable syntactic structures. The alternative is adopting the concept of verbal group, simple or complex, as proposed by the Sydney school in (Halliday & Matthiessen 2013: p.396–418, 567–592), which provides a richer semantically motivated description. However, analysis with verbal group complex is potentially complex one and subject to ambiguities.

Ants	*keep*	*biting*	*me*
Subject	Finite	Predicator	complement
Actor	Process: Material		Goal/Medium
	Verbal group complex expansion, elaborative, time-phase, durative $\alpha \longrightarrow = \beta$		

Table 4.1 Sydney sample analysis of a clause with a *verbal group complex*

Ants	*keep*	-	*biting*	*me*
Subject	Finite/Main Verb	Complement		
Agent	Process: Influential	Phenomena		
		Subject(null)	Main Verb	Complement
		Agent	Process: Action	Affected

Table 4.2 Cardiff sample analysis of a clause *embedded* into another

Check the sample analyses in Table 4.1 and 4.2. The two-clause analysis proposed by Cardiff school can be quite intuitively transformed into a single experiential structure with the top clause expressing a set of aspectual features of the process in the lower (embedded) clause just like in the Sydney analysis in Table 4.1.

The class of *influential* processes proposed in the Cardiff transitivity system was introduced to handle expressions of process aspects through other lexical verbs. I consider it as a class of pseudo-processes with a set of well defined and useful syntactic functions but with poor semantic foundation. The analysis with influential process types reminds me to an unstable chemical substance that, in a chain of reactions, is an intermediary step towards some more stable substance. Similarly, I propose merging the two clauses towards a more meaningful analysis, such as the one suggested by Sydney grammar.

Generalization 4.3.1 (Merging of influential clauses)**.** When the top clause has an influential process and the lower (embedded) one has any of the other processes, then the lower one shall be enriched with aspectual features that can be derived from the top one.

This rule of thumb is described in Generalization 4.3.1. Of course, this raises a set of problems that are worth investigating. Firstly, one should investigate the connections and mappings between the influential process system network described in

Cardiff grammar and the system of verbal group complex described in Sydney grammar (Halliday & Matthiessen 2013: p.589). Secondly, one should investigate how this merger impacts the syntactic structure.

The benefits of such a merger leads to an increased comprehensiveness, not only of the transitivity analysis – demonstrated by the examples in Tables 4.1 and 4.2 – but also of the modal assessment that includes modality, as demonstrated by the Examples 25 and 26.

(25) *I think* I've been pushed forward; *I don't really know*, (Halliday & Matthiessen 2013: p.183)

(26) *I believe* Sheridan once said you would've made an excellent pope. (Halliday & Matthiessen 2013: p.182)

Examples 25 and 26 represent cases when the modal assessment of the lower clause is carried on by the higher one. In both examples, the higher clause can be replaced by the modal verb *maybe* or the adverb *perhaps*.

4.3.2 Nominal, Quality, Quantity and other groups of Cardiff grammar: from syntactically towards semantically sound analysis

Cardiff unit classes are semantically motivated as compared to more syntactic ones in Sydney grammar. This has been presented in Sections **??** and discussed in **??**.

For instance, Nominal class structure proposed in Cardiff grammar (discussed in Section **??**), uses elements that are more semantic in nature (e.g. various types of determiners: representational, quantifying, typic, partitive etc.) than the syntactic one offered in Sydney grammar (e.g. only deictic determiner). To do this shift we need to think of two problems: (a) how to detect the semantic head of the nominal units and (b) how to craft (if none exists) a lexical-semantic resources to help determining potential functions (structural elements) for each lexical item in the nominal group. In my view building lexical-semantic resources asked at point (b) bears actually a solution for point (a) as well.

I need to stress that some existing lexical resources such as WordNet (Miller 1995) and/or FrameNet(Baker et al. 1998) could and most likely are suitable for fulfilling the needs at point (b) but the solution is not straight forward and further adaptations need to be done for the context of SFL.

The same holds for Adverbial and Adjectival groups (discussed in Section **??**) which, in Cardiff grammar, are split into the Quality and Quantity groups. The existent lexical resources such as such as WordNet (Miller 1995) and/or FrameNet(Baker et al. 1998) combined with the delicate classification proposed by Tucker (1997) (and other research must exist on adverbial groups of which I am not aware at the moment) can yield positive results in parsing with Cardiff unit classes.

Just like in the case of verb groups discussed in previous section, moving towards semantically motivated unit classes, as proposed in Cardiff grammar, would greatly benefit applications requiring deeper natural language understanding.

4.3.3 Taxis analysis and potential for discourse relation detection

Currently Parsimonious Vole parser implements a simple taxis analysis technique based on patterns represented as regular expressions.

In the Appendix is listed a database of clause taxis patterns according to systematization in IFG 3 (Halliday & Matthiessen 2004). Each relation type has a set of patterns ascribed to it which represent clause order and presence or absence of explicit lexical markers or clause features.

Then, in taxis analysis process, each pair of adjacent clauses in the sentence is tested for compliance with every pattern in the database. The matches represent potential manifestation of the corresponding relation.

Currently this part of the parser has not been tested and it remains a highly desirable future work. Further improvements and developments can be performed based on incremental testing and corrections of the taxis pattern database.

This work can be extended to handle relations between sentences taking on a discourse level analysis which is perfectly in line with the Rhetorical Structure Theory (RST) (Mann & Thompson 1988; Mann et al. 1992).

To increase the accuracy of taxis analysis, I believe the following additional elements shall be included into the pattern representation: Transitivity configurations including process type and participant roles, co-references resolved between clauses/sentences and Textual metafunction analysis in terms of Theme/Rheme and eventually New/Given.

4.3.4 Towards speech act analysis

As Robin Fawcett explains (Fawcett 2011), Halliday's approach to MOOD analysis differs from that of Transitivity in the way that the former is not "pushed forward

towards semantics" as the latter is. Having a semantically systematised MOOD system would take the interpersonal text analysis into a realm compatible with Speech Act Theory proposed by Austin (1975) or its latter advancements such as the one of Searle (1969) which, in mainstream linguistics, are placed under the umbrella of pragmatics.

Halliday proposes a simple system of speech functions (Halliday & Matthiessen 2013: p.136) which Fawcett develops into a quite delicate system network (Fawcett 2011). It is worth exploring ways to implement Fawcett's latest developments and because the two are not conflicting but complementing each other, one could use Hallidayan MOOD system as a foundation, especially that it has already been implemented and described in the current work.

4.3.5 Process Types and Participant Roles

The PTDB (Neale 2002) is the first lexical-semantic resource for Cardiff grammar Transitivity system. Its usability in the original form doesn't go beyond that of a resource to be consulted by linguists in the process of manual analysis. It was rich in human understandable comments and remarks but not formal enough to be usable by computers. In the scope of current work the PTDB has been cleaned and brought into a machine readable form but this is far from it's potential as a lexical-grammatical resource for semantic parsing.

In the mainstream computational linguistics, there exist several other lexical-semantic resources used for Semantic Role Labelling (SRL) such as FrameNet (Baker et al. 1998), VerbNet (Kipper et al. 2008). Mapping or combining PTDB with these resources into a new one would yield benefits for both sides combining strengths of each and covering their shortcomings.

Combining PTDB with VerbNet for example, would be my first choice for the following reasons. PTDB is well semantically systematised according to Cardiff Transitivity system however it lacks any links to syntactic manifestations. VerbNet, on the other hand contains an excellent mapping to the syntactic patterns in which each verb occur, each with associated semantic representation of participant roles and some first order predicates. However, the systematization of frames and participant roles could benefit from a more robust basis of categorisation. Also the lexical coverage of VerbNet is wider than that of PTDB.

Turning towards resources like FrameNet and WordNet could bring other benefits. For example FrameNet has a set of annotated examples for every frame which, after transformation into Transitivity system, could be used as a training corpus for machine learning algorithms. Another potential benefit would be generating semantic constrains

(for example in terms of WordNet (Miller 1995) synsets or GUM (Bateman et al. 1995, 2010) classes) for every participant role in the system.

PTDB can benefit from mappings with GUM ontology which formalises the experiential model of Sydney school. First by increasing delicacy (at at the moment it covers only three top levels of the system) and second by importing constraints on process types and participant roles from Nigel grammar (Matthiessen 1985). To achieve this, one would have to first map Cardiff and Sydney Transitivity systems and second extract lexical entries from Nigel grammar along with adjacent systemic selections.

4.3.6 Reasoning with systemic networks

Systemic networks are a powerful instrument to represent paradigmatic dimension of language. Besides hierarchies they can include constraints on which selections can actually go together or a more complex set of non hierarchical selection interdependencies. Moreover systemic choices can be also accompanied by the realization rules very useful for generation purpose but they could potentially be used in parsing as well.

In current work system networks are used solely for representation purposes and what would be highly desirable is to enable reasoning capabilities for constraint checking on systemic selections and on syntactic and semantic constituency. For example one could as whether a certain set of features are compatible with each other, or provided a systemic network and several feature selections what would be the whole set of system choices, or being in a particular point in the system network what are the possible next steps towards more delicate systemic choices, or for a particular choice or set of choices what should be present or absent in the constituency structure of the text and so on. All these questions could potentially be resolved by a systemic reasoner.

Martin Kay is the first to attempt formalization of systemics that would become known as Functional Unification Grammar (FUG) (Kay 1985). This formalization caught on popularity in other linguistic domains such as HPSG, Lexical Functional Grammars and Types Feature Structures. One could look at what has been done and adapt the or build a new reasoning system for systemic networks.

With the same goal in mind, one could also look at existing reasoners for different logics and attempt an axiomatization of the systemic networks; and more specifically one could do that in Prolog language or with description logics (DL) as there is a rich set of tools and resources available in the context of Semantic Web.

4.3.7 Creation of richly annotated corpus with all metafunction: interpersonal, experiential and textual

In order to evaluate a parser, a gold standard annotation corpus is essential. The bigger the corpus, covering various the text genres, the more reliable are the evaluation results. A corpus can as well be the source of grammar or distribution probabilities for structure element and potential filling units as is explored by Day (2007), Souter (1996) and other scholars in Cardiff. Moreover such a corpus can also constitute the training data set for a machine learning algorithm for parsing.

A corpus of syntactically annotated texts with Cardiff grammar already exists but, from personal communication with Prof. Robin Fawcett, it is not yet been released to public because it is considered still incomplete. Even so this corpus covers only the constituency structures and what I would additionally find very useful, would be a set of systemic features of the constituting units covering a full SFG analysis in terms of experiential, interpersonal and textual metafunctions; and not only the unit class and the element it fills.

A small richly annotated set of text had been created in the scope of the current work for the purpose of evaluating the parser. However it is by far not enough to offer a reliable evaluation. Therefore it is highly desirable to create one.

To approach this task one could use a systemic functional annotation tool such as UAM Corpus Tool (O'Donnell 2008a,b) developed and still maintained by Mick O'Donnell or any other tool that supports segment annotation with systemic network tag set structure.

To aid this task one could bootstrap this task by converting other existing corpuses such as Penn Treebank. This task had been already explored by Honnibal in 2004; 2007.

4.3.8 The use of Markov Logics for pattern discovery

Markov Logic (Richardson & Domingos 2006; Domingos et al. 2010) is a probabilistic logic which applies ideas of Markov network to first order logic enabling uncertain inference. What is very interesting about this logics is that tools implementing it have learning capabilities not only of formulas weights but also of new logical clauses.

In current approach I am using graph patterns matching technique to generate a rich set of features for the constituent units. However creating those patterns is a tremendous effort.

Since, graph patterns can be expressed via first order functions and individuals, and assuming that there would already exist a richly annotated corpus, the Markov Logic instruments (for example Alchemy[1], Tuffy[2] and others) can be employed to inductively learn such patterns from the corpus.

This approach resembles the Vertical Strips (VS) of O'Donoghue (1991). The similarity is the probabilistic learning of patterns from the corpus. The difference is that VS patterns are syntactic segment chains from the root node down to tree leafs while with ML more complex patterns can be learned independently of their position in the syntactic tree. Moreover such patterns can be bound to specific feature set.

propose a natural
guage task classific
based on the numb
features needed as
and provided as ou

4.4 Overall evaluations

A deduction ma
on the basis of t
main body (i.e.
cluding statemen

4.5 A final word

The writer's perso
opinion on wha
has been discuss

half to one page

[1] http://alchemy.cs.washington.edu/
[2] http://i.stanford.edu/hazy/hazy/tuffy/

References

Abney, S. 1987. *The English noun phrase in its sentential aspects*. MIT Press.

Austin, J L. 1975. *How to do things with words*, vol. 3 (Syntax and Semantics 1). Harvard University Press.

Bach, Emmon. 1966. *An introduction to transformational grammars*. Holt, Rinehart and Winston. Inc.

Baker, Collin F, Charles J Fillmore & John B Lowe. 1998. The Berkeley FrameNet Project. In Christian Boitet & Pete Whitelock (eds.), *Proceedings of the 36th annual meeting on association for computational linguistics*, vol. 1 ACL '98, 86–90. University of Montreal Association for Computational Linguistics. doi:10.3115/980845.980860. <http://portal.acm.org/citation.cfm?doid=980845.980860>.

Bar-Hillel, Yehoshua. 1953. A quasi-arithmetical notation for syntactic description. *Language* 29. 47–58. Reprinted in Y. Bar-Hillel. (1964). *Language and Information: Selected Essays on their Theory and Application*, Addison-Wesley 1964, 61–74.

Bateman, John A. 2008. Systemic-Functional Linguistics and the Notion of Linguistic Structure: Unanswered Questions, New Possibilities. In Jonathan J. Webster (ed.), *Meaning in context: Implementing intelligent applications of language studies*, 24–58. Continuum.

Bateman, John A, Renate Henschel & Fabio Rinaldi. 1995. The Generalized Upper Model . Tech. rep. GMD/IPSI. <http://www.fb10.uni-bremen.de/anglistik/langpro/webspace/jb/gum/gum-2.pdf>.

Bateman, John A, Joana Hois, Robert Ross & Thora Tenbrink. 2010. A linguistic ontology of space for natural language processing. *Artificial Intelligence* 174(14). 1027–1071. doi:10.1016/j.artint.2010.05.008. <http://linkinghub.elsevier.com/retrieve/pii/S0004370210000858>.

Bateman, John A. & Christian M. I. M. Matthiessen. 1988. Using a functional grammar as a tool for developing planning algorithms — an illustration drawn from nominal group planning. Tech. rep. Information Sciences Institute Marina del Rey, California. (Penman Development Note).

Bloomfield, Leonard. 1933. *Language*. New York and London: Henry Holt and Co. and Allen and Unwin Ltd.

Böhmová, Alena, Jan Hajič, Eva Hajičová & Barbora Hladká. 2003. *The prague dependency treebank* 103–127. Dordrecht: Springer Netherlands. doi:10.1007/978-94-010-0201-1_7. <https://doi.org/10.1007/978-94-010-0201-1_7>.

Bohnet, Bernd. 2010. Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd international conference on computational linguistics* COLING '10, 89–97. Stroudsburg, PA, USA: Association for Computational Linguistics. <http://dl.acm.org/citation.cfm?id=1873781.1873792>.

Bondy, John Adrian, Uppaluri Siva Ramachandra Murty et al. 1976. *Graph theory with applications*, vol. 290. Citeseer.

Bresnan, Joan. 1982. Control and complementation. *Linguistic Inquiry* 13(3). 343–434.

Bresnan, Joan. 2001. *Lexical-Functional Syntax*. Blackwell. <http://books.google.lu/books/about/Lexical{_}Functional{_}Syntax.html?id=vMxgevXoq{_}gC{&}redir{_}esc=y>.

Buchholz, Sabine & Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the tenth conference on computational natural language learning* CoNLL-X '06, 149–164. Stroudsburg, PA, USA: Association for Computational Linguistics. <http://dl.acm.org/citation.cfm?id=1596276.1596305>.

Bühler, Karl. 1934. *Sprachtheorie: die Darstellungsfunktion der Sprache*. Jena: Fischer.

Butler, Christopher. 1985. *Systemic linguistics: Theory and applications*. Batsford Academic and Educational.

Butler, Christopher S. 2003a. *Structure and function: A guide to three major structural-functional theories; Part 1: Approaches to the simplex clause*. Amsterdam and Philadelphia: John Benjamins.

Butler, Christopher S. 2003b. *Structure and function: A guide to three major structural-functional theories; Part 2: From clause to discourse and beyond*. Amsterdam and Philadelphia: John Benjamins.

Carreras, Xavier. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (emnlp-conll)*, .

Carreras, Xavier & Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of the ninth conference on computational natural language learning* CONLL '05, 152–164. Stroudsburg, PA, USA: Association for Computational Linguistics. <http://dl.acm.org/citation.cfm?id=1706543.1706571>.

Carroll, John, Ted Briscoe & Antonio Sanfilippo. 1998. Parser evaluation: a survey and a new proposal. In *Proceedings of the international conference on language resources and evaluation*, 447–454.

Carroll, John, Guido Minnen & Ted Briscoe. 1999. Corpus Annotation for Parser Evaluation. In *Proceedings of the eacl workshop on linguistically interpreted corpora (linc)* June, 7. Association for Computational Linguistics. <http://arxiv.org/abs/cs/9907013>.

Cer, Daniel D.M., Marie-Catherine M.C. De Marneffe, Daniel Jurafsky & C.D. Manning. 2010. Parsing to stanford dependencies: Trade-offs between speed and accuracy. In *Lrec 2010*, vol. 0, European Language Resources Association. <http://171.64.67.140/pubs/lrecstanforddeps{_}final{_}final.pdfhttp://www.lrec-conf.org/proceedings/lrec2010/pdf/730{_}Paper.pdf>.

Chen, Danqi & Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (emnlp)*, 740–750.

Chomsky, Noam. 1956. Three models for the description of language. *IRE Transactions on Information Theory* 2. 113–124.

Chomsky, Noam. 1957a. *Syntactic Structures*. Mouton & Co.

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

Chomsky, Noam. 1965. *Aspects of the theory of syntax*. Cambridge, Massachusetts: M.I.T. Press.

Chomsky, Noam. 1981. *Lectures on Government and Binding*. Dordrecht: Foris.

Chomsky, Noam. 1982. *Some concepts and consequences of the theory of government and binding*, vol. 6. MIT press.

Chomsky, Noam. 1986. *Barriers*, vol. 13. MIT press.

Chomsky, Noam. 1993a. A Minimalist Program for Linguistic Theory. In K. Hale & S. J. Keyser (eds.), *The View from Building 20*, Cambridge, Mass: MIT Press.

Chomsky, Noam. 1993b. *Lectures on government and binding: The pisa lectures* 9. Walter de Gruyter.

Clegg, Andrew B & Adrian J Shepherd. 2007. Benchmarking natural-language parsers for biological applications using dependency graphs. *BMC bioinformatics* 8(1). 24.

Day, Michael David. 2007. *A Corpus-Consulting Probabilistic Approach to Parsing : the CCPX Parser and its Complementary Components*: Cardiff University dissertation.

Domingos, Pedro, Stanley Kok, Daniel Lowd & Hoifung Poon. 2010. Markov Logic. *Journal of computational biology a journal of computational molecular cell biology* 17(11). 1491–508. doi:10.1089/cmb.2010.0044. <http://www.ncbi.nlm.nih.gov/pubmed/21685052>.

Fawcett, R. 1988a. Language Generation as Choice in Social Interaction. In Zock, M. & G. Sabah (eds.), *Advances in natural language generation*, vol. 2, 27–49. Pinter Publishers. (Paper presented at the First European Workshop of Natural Language Generation, Royaumont, 1987).

Fawcett, Robin. 2000. *A Theory of Syntax for Systemic Functional Linguistics*. John Benjamins Publishing Company paperback edn.

Fawcett, Robin P. 1988b. What makes a 'good' system network good? In James D. Benson & William S. Greaves (eds.), *Systemic functional approaches to discourse*, 1–28. Norwood, NJ: Ablex.

Fawcett, Robin P. 1990. The COMMUNAL project: two years old and going well. *Network: news, views and reviews in systemic lingustics and related areas* 13/14. 35–39.

Fawcett, Robin P. 1993. The architecture of the COMMUNAL project in NLG (and NLU). In *The Fourth European Workshop on Natural Language Generation*, Pisa.

Fawcett, Robin P. 2008. *Invitation to Systemic Functional Linguistics through the Cardiff Grammar*. Equinox Publishing Ltd.

Fawcett, Robin P. 2009. How to Analyze Process and Participant Roles. In *The functional semantics handbook: Analyzing english at the level of meaning*, Continuum.

Fawcett, Robin P. 2011. A semantic system network for MOOD in English (and some complementary system networks).

Fellbaum, Christiane & George Miller (eds.). 1998. *WordNet: An electronic lexical database*. The MIT Press.

Fillmore, Charles J. 1985. Frames and the semantics of understanding. *Quaderni di Semantica* 6(2). 222–254. <http://scholar.google.it/scholar?q=fillmore{&}hl=it{&}btnG=Cerca{&}lr={#}5>.

Fillmore, Charles J, Christopher R Johnson & Miriam RL Petruck. 2003. Background to framenet. *International journal of lexicography* 16(3). 235–250.

Firth, J.R. 1957. A synopsis of linguistic theory 1930-1955. *Studies in linguistic analysis* 1–32. <http://www.bibsonomy.org/bibtex/25b0a766713221356e0a5b4cc2023b86a/glanebridge>.

Gaifman, Haim. 1965. Dependency systems and phrase-structure systems. *Information and Control* 8(3). 304 – 337. doi:https://doi.org/10.1016/S0019-9958(65)90232-9. <http://www.sciencedirect.com/science/article/pii/S0019995865902329>.

Gale, D. & L. S. Shapley. 1962. College admissions and the stability of marriage. *The American Mathematical Monthly* 69(1). 9–15. <http://www.jstor.org/stable/2312726>.

Garde, Paul. 1977. Ordre lineaire et dependance syntaxique. contribution a une typologie. In *Bulletin de la societe de linguistique de paris*, vol. 1 72, 1–26. Paris.

Gerdes, Kim & Sylvain Kahane. 2013. Defining dependencies (and constituents). *Frontiers in Artificial Intelligence and Applications* 258. 1–25.

Gildea, Daniel & Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational linguistics* 28(3). 245–288.

Haegeman, Liliane. 1991. *Introduction to Government and Binding Theory*, vol. 2. Blackwell.

Hajic, Jan, Eva Hajicová, Petr Pajas, Jarmila Panevová, Petr Sgall & Barbora Vidová-Hladká. 2001. Prague dependency treebank 1.0 (final production label). cdrom cat: Ldc2001t10. Tech. rep. ISBN 1-58563-212-0.

Halliday, M.A.K. 1968a. Language and experience. *Educational Review* 20(2). 95–106.

Halliday, Michael A. K. 1957. Some aspects of systematic description and comparison in grammatical analysis. In *Studies in Linguistic Analysis*, 54–67. Oxford: Blackwell.

Halliday, Michael A. K. 1961a. Categories of the theory of grammar. *Word* 17(3). 241–292.

Halliday, Michael A. K. 1961b. Categories of the theory of grammar. *Word* 17(3). 241–292. Reprinted in abbreviated form in Halliday (1976) edited by Gunther Kress, pp 52-72.

Halliday, Michael A. K. 1967. Notes on transitivity and theme in English — Parts 1 and 2. *Journal of Linguistics* 3. 37–81 and 199–244.

Halliday, Michael A. K. 1968b. Notes on transitivity and theme in English — Part 3. *Journal of Linguistics* 4. 179–215.

Halliday, Michael A. K. 1994. *An Introduction to Functional Grammar*. London: Edward Arnold 2nd edn.

Halliday, Michael A. K. 1996. On grammar and grammatics. In Ruqaiya Hasan, Carmel Cloran & David Butt (eds.), *Functional descriptions – theory in practice* Current Issues in Linguistic Theory, 1–38. Amsterdam: Benjamins.

Halliday, Michael A. K. 1997. Linguistics as metaphor, 3–27. Continuum.

Halliday, Michael A. K. 2003a. Ideas about language. In Michael A. K. Halliday & Jonathan J. Webster (eds.), *On language and linguistics. Volume 3 of collected works of M.A. K. Halliday*, 490. New York: Continuum.

Halliday, Michael A. K. 2003b. Systemic theory. In Michael A. K. Halliday & Jonathan J. Webster (eds.), *On language and linguistics. Volume 3 of collected works of M.A. K. Halliday*, 490. New York: Continuum.

Halliday, Michael A.K. 2002. Categories of the theory of grammar. In Jonathan Webster (ed.), *On grammar (volume 1)*, 442. Continuum.

Halliday, Michael A.K. 2003c. On the "architecture" of human language. In Jonathan Webster (ed.), *On language and linguistics*, vol. 3 Collected Works of M. A. K. Halliday, 1–32. Continuum.

Halliday, Michael A.K. & Christian M.I.M. Matthiessen. 2013. *An Introduction to Functional Grammar (4^{th} Edition)*. Routledge 4th edn.

Halliday, Michael A.K. & M.I.M. Matthiessen, Christian. 2004. *An introduction to functional grammar (3^{rd} Edition)*. Hodder Education.

Harnad, Stevan. 1992. The Turing Test Is Not A Trick: Turing Indistinguishability Is A Scientific Criterion. *SIGART Bulletin* 3(4). 9–10. <http://users.ecs.soton.ac.uk/harnad/Papers/Harnad/harnad92.turing.html>.

Harris, Z.S. 1951. *Methods in structural linguistics* Methods in Structural Linguistics. University of Chicago Press. <https://books.google.lu/books?id=a6nYjgEACAAJ>.

Harrison, Philip, Steven Abney, Ezra Black, Dan Flickinger, Claudia Gdaniec, Ralph Grishman, Donald Hindle, Robert Ingria, Mitch Marcus, Beatrice Santorini & Tomek Strzalkowski. 1991. Evaluating syntax performance of parser/grammars. In *Proceedings of the natural language processing systems evaluation workshop, berekely, ca, june 1991* Rome Laboratory Technical Report, RL-TR-91-362, .

Hasan, Ruqaiya. 2014. The grammarian's dream: lexis as most delicate grammar. In Jonathan Webster (ed.), *Describing language form and function*, vol. 5 Collected Works of Ruqaiya Hasan, chap. 6. Equinox Publishing Ltd.

Hays, David G. 1960. Grouping and dependency theories. *Proceedings of the National Symposium on Machine Translation* 2538. 258–266.

Hays, David G. 1964. Dependency theory: A formalism and some observations. *Language* 40(4). 511–525. <http://www.jstor.org/stable/411934>.

Hjelmslev, Louis. 1953. *Prolegomena to a theory of language*. Bloomington, Indiana: Indiana University Publications in Anthropology and Linguistics. Translated by Francis J. Whitfield.

Hockett, Charles F. 1958. *A course in modern linguistics*. New York: Macmillan.

Honnibal, Matthew. 2004. Converting the Penn Treebank to Systemic Functional Grammar. *Technology* 147–154.

Honnibal, Matthew & Jr James R Curran. 2007. Creating a systemic functional grammar corpus from the Penn treebank. *Proceedings of the Workshop on Deep ...* 89–96. doi:10.3115/1608912.1608927. <http://dl.acm.org/citation.cfm?id=1608927>.

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

Hutchins, W John. 1999. Retrospect and prospect in computer-based translation. In *Proceedings of mt summit vii "mt in the great translation era"* September, 30–44. AAMT.

Johnson, Christopher & Charles J. Fillmore. 2000. The framenet tagset for frame-semantic and syntactic coding of predicate-argument structure. In *Proceedings of the 1st north american chapter of the association for computational linguistics conference* NAACL 2000, 56–62. Stroudsburg, PA, USA: Association for Computational Linguistics. <http://dl.acm.org/citation.cfm?id=974305.974313>.

Kasper, Robert. 1988. An Experimental Parser for Systemic Grammars. In *Proceedings of the 12th International Conference on Computational Linguistics*, .

Kay, Martin. 1985. Parsing In Functional Unification Grammar. In D.Dowty, L. Karttunen & A. Zwicky (eds.), *Natural language parsing*, Cambridge University Press.

King, Tracy Holloway & Richard Crouch. 2003. The PARC 700 Dependency Bank. In *4th international workshop on linguistically interpreted corpora (linc03)*, <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.15.3277>.

Kipper, Karin, Anna Korhonen, Neville Ryant & Martha Palmer. 2008. A large-scale classification of English verbs. *Language Resources And Evaluation* 42(1). 21–40. doi:10.1007/s10579-007-9048-2.

Kübler, S., R. McDonald & J. Nivre. 2009. *Dependency parsing* Online access: IEEE (Institute of Electrical and Electronics Engineers) IEEE Morgan & Claypool Synthesis eBooks Library. Morgan & Claypool. <https://books.google.lu/books?id=k3iiup7HB9UC>.

Kucera, Henry & W. Nelson Francis. 1968. Computational Analysis of Present-Day American English. *American Documentation* 19(4). 419. doi:10.2307/302397. <http://search.ebscohost.com/login.aspx?direct=true{&}db=bth{&}AN=16865479{&}login.asp{&}site=ehost-live>.

Lecerf, Yves. 1961. Une representation algebrique de la structure des phrases dans diverses langues naturelles. In *Compte rendu de l'academie des sciences de paris* 252, 232–234. Academie des Sciences.

Lemke, Jay L. 1993. Discourse, dynamics, and social change. *Cultural Dynamics* 6(1-2). 243–276.

Lin, Dekang & Patrick Pantel. 2001. Discovery of inference rules for question-answering. *Natural Language Engineering* 7(4). 343–360.

Mann, William C. 1983a. An Overview of the PENMAN Text Generation System. Tech. Rep. ISI/RR-83-114 USC/Information Sciences Institute Marina del Rey, CA.

Mann, William C. 1983b. An overview of the PENMAN text generation system. In *Proceedings of the National Conference on Artificial Intelligence*, 261–265. AAAI. Also appears as USC/Information Sciences Institute, RR-83-114.

Mann, William C. & Christian M. I. M. Matthiessen. February 1983. A demonstration of the Nigel text generation computer program. In *Nigel: A Systemic Grammar for Text Generation*, USC/Information Sciences Institute, RR-83-105. This paper also appears in a volume of the *Advances in Discourse Processes Series*, R. Freedle (ed.): *Systemic Perspectives on Discourse: Volume I.* published by Ablex.

Mann, William C., Christian M. I. M. Matthiessen & Sandra A. Thompson. 1992. Rhetorical Structure Theory and Text Analysis. In William C Mann & Sandra A Thompson (eds.), *Discourse description: Diverse linguistic analyses of a fund-raising text*, vol. 16 Pragmatics & Beyond New Series, 39–79. John Benjamins Publishing Company.

Mann, William C & Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text* 8(3). 243–281. doi:10.1515/text.1.1988.8.3.243.

Marcus, Mitchell P, Beatrice Santorini & Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19(2). 313–330. doi:10.1162/coli.2010.36.1.36100. <http://portal.acm.org/citation.cfm?id=972470.972475>.

Marneffe, Marie-Catherine, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre & Christopher D. Manning. 2014. Universal Stanford dependencies: A cross-linguistic typology. In *Proceedings of the ninth international conference on language resources and evaluation (lrec-2014)(vol. 14)*, European Language Resources Association (ELRA). <http://www.lrec-conf.org/proceedings/lrec2014/pdf/1062{_}Paper.pdf>.

Marneffe, Marie-Catherine, Bill MacCartney & Christopher D Manning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In *Lrec 2006*, vol. 6 3, 449–454. Stanford University. <http://nlp.stanford.edu/manning/papers/LREC{_}2.pdf>.

Marneffe, Marie-Catherine & Christopher D. Manning. 2008a. Stanford typed dependencies manual. Tech. Rep. September Stanford University. <http://nlp.stanford.edu/downloads/dependencies{_}manual.pdf>.

Marneffe, Marie-Catherine & Christopher D. Manning. 2008b. The Stanford typed dependencies representation. *Coling 2008 Proceedings of the workshop on CrossFramework and CrossDomain Parser Evaluation CrossParser 08* 1(ii). 1–8. doi:10.3115/1608858.1608859. <http://portal.acm.org/citation.cfm?doid=1608858.1608859>.

Matthiessen, Christian M. I. M. 1995. *Lexicogrammatical cartography: English systems*. Tokyo, Taipei and Dallas: International Language Science Publishers.

Matthiessen, Christian M. I. M. & John A. Bateman. 1991. *Text generation and systemic-functional linguistics: experiences from English and Japanese*. London and New York: Frances Pinter Publishers and St. Martin's Press.

Matthiessen, M.I.M., Christian. 1985. The systemic framework in text generation: Nigel. In James Benson & Willian Greaves (eds.), *Systemic perspective on Discourse, Vol I*, 96–118. Ablex.

McCarthy, John, Marvin L. Minsky, Nathaniel Rochester & Claude E. Shannon. 2006. A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, August 31, 1955. *AI Magazine* 27(4). 12. doi:10.1609/aimag.v27i4.1904. <http://www.aaai.org/ojs/index.php/aimagazine/article/view/1904{%}5Cnhttp://www.mendeley.com/catalog/proposal-dartmouth-summer-research-project-artificial-intelligence-august-31-1955/{%}5Cnhttp://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.htmlhttp://>.

McDonald, David D. 1980. *Natural Language Production as a Process of Decision Making under Constraint*: MIT, Cambridge, Mass dissertation.

McDonald, Ryan, Koby Crammer & Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, 91–98. Association for Computational Linguistics.

McDonald, Ryan, Kevin Lerman & Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the tenth conference on computational natural language learning* CoNLL-X '06, 216–220. Stroudsburg, PA, USA: Association for Computational Linguistics. <http://dl.acm.org/citation.cfm?id=1596276.1596317>.

McDonald, Ryan & Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *11th conference of the european chapter of the association for computational linguistics*, .

Mel'čuk, Igor. 1988. Semantic description of lexical units in an Explanatory Combinatorial Dictionary: basic principles and heuristic criteria. *International Journal of Lexicography* 1. 165–188.

Mel'čuk, Igor A. & Nikolaj V. Pertsov. 1986. *Surface syntax of english*. John Benjamins Publishing. doi:10.1075/llsee.13. <http://www.jbe-platform.com/content/books/9789027279378>.

Miller, George A. 1995. WordNet: a lexical database for English.

Miyao, Yusuke & Jun'ichi Tsujii. 2005. Probabilistic disambiguation models for wide-coverage hpsg parsing. In *Proceedings of the 43rd annual meeting on association for computational linguistics* ACL '05, 83–90. Stroudsburg, PA, USA: Association for Computational Linguistics. doi:10.3115/1219840.1219851. <https://doi.org/10.3115/1219840.1219851>.

Moravcsik, Edith A. 2006. *An Introduction to Syntactic Theory*. Continuum paperback edn.

Neale, Amy C. 2002. More Delicate TRANSITIVITY: Extending the PROCESS TYPE for English to include full semantic classifications. Tech. rep. Cardiff University.

Nivre, Joakim. 2006. *Inductive dependency parsing (text, speech and language technology)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Nivre, Joakim. 2015. Towards a universal grammar for natural language processing. In *Computational Linguistics and Intelligent Text Processing*, 3–16. Springer.

Nivre, Joakim, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel & Deniz Yuret. 2007a. The conll 2007 shared task on dependency parsing. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (emnlp-conll)*, 915–932.

Nivre, Joakim, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov & Erwin Marsi. 2007b. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering* 13(2). 95–135. doi:10.1017/S1351324906004505. <https://doi.org/10.1017/S1351324906004505>.

Nivre, Joakim, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty & Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk & Stelios Piperidis (eds.), *Proceedings of the tenth international conference on language resources and evaluation (lrec 2016)*, Paris, France: European Language Resources Association (ELRA).

O'Donnell, Michael. 1993. Reducing Complexity in Systemic Parser. In *Proceeedings of the third international workshop on parsing technologies*, .

O'Donnell, Michael. 1994. Sentence Analysis and Generation: a systemic perspective. Tech. rep. Department of Linguistics, University of Sydney.

O'Donnell, Michael. 2005. The UAM Systemic Parser. *Proceedings of the 1st Computational Systemic Functional Grammar Conference* <http://www.wagsoft.com/Papers/ODonnellUamParser.pdf>.

O'Donnell, Michael J. & John A. Bateman. 2005. SFL in computational contexts: a contemporary history. In Ruqiaya Hasan, M.I.M. Matthiessen, Christian & Jonathan Webster (eds.), *Continuing discourse on language: A functional perspective*, vol. 1 Booth 1956, 343–382. Equinox Publishing Ltd.

O'Donnell, Mick. 2008a. Demonstration of the UAM CorpusTool for text and image annotation. In *Proceedings of the acl-08:hlt demo session* June, 13–16.

O'Donnell, Mick. 2008b. The UAM CorpusTool: Software for Corpus Annotation and Exploration. In Bretones Callejas & Carmen M. (eds.), *Applied linguistics now: Understanding language and mind*, vol. 00, 1433–1447. Universidad de Almería.

O'Donoghue, Tim. 1991. The Vertical Strip Parser: A lazy approach to parsing. Tech. rep. School of Computer Studies, University of Leeds.

Pei, Wenzhe, Tao Ge & Baobao Chang. 2015. An effective neural network model for graph-based dependency parsing. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*, vol. 1, 313–322.

Penman Project. 1989. PENMAN documentation: the Primer, the User Guide, the Reference Manual, and the Nigel Manual. Tech. rep. USC/Information Sciences Institute Marina del Rey, California.

Pollard, Carl & Ivan Sag. 1987. *Information-Based Syntax and Semantics*. CSLI.

Pollard, Carl J. & Ivan A. Sag. 1994. *Head-driven phrase structure grammar*. University of Chicago Press.

Pollock, Jean-Yves. 1989. Verb movement, universal grammar, and the structure of ip. *Linguistic inquiry* 20(3). 365–424.

Postal, P. M. 1974. *On Raising*. MIT Press.

Quirk, Randolph, Sidney Greenbaum, Geoffrey Leech, Jan Svartvik & David Crystal. 1985. *A comprehensive grammar of the English language*, vol. 1 2. Longman. <http://www.amazon.com/dp/0582517346http://journals.cambridge.org/production/action/cjoGetFulltext?fulltextid=2545152>.

Radford, Andrew. 1997. *Syntax: A Minimalist Introduction*. Cambridge University Press.

Richardson, Matthew & P. Domingos. 2006. Markov logic networks. *Machine learning* 62(1-2). 107–136. doi:10.1007/s10994-006-5833-1.

Saitta, Lorenza & Jean-Daniel Zucker. 2013. *Abstraction in artificial intelligence and complex systems*. Springer-Verlag New York. doi:10.1007/978-1-4614-7052-6. <http://www.springer.com/la/book/9781461470519>.

Santorini, Beatrice. 1990. Part-of-Speech Tagging Guidelines for the Penn Treebank Project (3rd Revision). *University of Pennsylvania 3rd Revision 2nd Printing* 53(MS-CIS-90-47). 33. doi:10.1017/CBO9781107415324.004. <http://www.personal.psu.edu/faculty/x/x/xxl13/teaching/sp07/apling597e/resources/Tagset.pdf>.

Saussure, Ferdinand de. 1959 [1915]. *Course in General Linguistics*. New York / Toronto / London: McGraw-Hill and the Philosophical Library, Inc. Edited by Charles Bally and Albert Sechehaye, in collaboration with Albert Riedlinger; translated by Wade Baskin.

Schank, Roger. 1969. Conceptual dependency as a framework for linguistic analysis. *Linguistics* 7(49). 28–50.

Schuler, Karin Kipper. 2005. Verbnet: A broad-coverage, comprehensive verb lexicon .

Searle, John R. 1969. *Speech Acts: An Essay in the Philosophy of Language*, vol. 0. Cambridge University Press. <http://books.google.com/books?id=t3{_}WhfknvF0C{&}pgis=1>.

Silveira, Natalia, Timothy Dozat, Marie-Catherine De Marneffe, Samuel Bowman, Miriam Connor, John Bauer & Chris Manning. 2014. A Gold Standard Dependency Corpus for English. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk & Stelios Piperidis (eds.), *Proceedings of the ninth international conference on language resources and evaluation (lrec'14)*, European Language Resources Association (ELRA). <http://nlp.stanford.edu/pubs/USD{_}LREC14{_}paper{_}camera{_}ready.pdf>.

Sleator, Daniel Dominic & David Temperley. 1995. Parsing english with a link grammar. *CoRR* abs/cmp-lg/9508004. 93. <http://arxiv.org/abs/cmp-lg/9508004>.

Snow, Rion, Daniel Jurafsky & Andrew Y Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *Advances in neural information processing systems*, 1297–1304.

Souter, David Clive. 1996. *A Corpus-Trained Parser for Systemic-Functional Syntax*: University of Leeds Phd. <http://etheses.whiterose.ac.uk/1268/>.

Sowa, John F. 1976. Conceptual graphs for a data base interface. *IBM Journal of Research and Development* 20(4). 336–357. doi:10.1147/rd.204.0336. <http://dx.doi.org/10.1147/rd.204.0336>.

Stockwell, Robert P., Barbara Hall Partee & Paul Schacter. 1973. *The major syntactic structures of english*. New York: Holt, Rinehart and Winston. <http://hdl.handle.net/2027/mdp.39015002216417>. Bibliography: p. 811-840.

Stowell, T.A. & E. Wehrli. 1992. *Syntax and the lexicon* Syntax and semantics. Academic Press. <https://books.google.lu/books?id=yiEcAQAAIAAJ>.

Taverniers, Miriam. 2011. The syntax-semantics interface in systemic functional grammar: Halliday's interpretation of the Hjelmslevian model of stratification. *Journal of Pragmatics* 43(4). 1100–1126. doi:10.1016/j.pragma.2010.09.003.

Tesniere, Lucien. 1959. *Elements de syntaxe structurale*. Paris: Klincksieck.

Tesniere, Lucien. 2015. *Elements of Structural Syntax*. John Benjamins Publishing Company translation by timothy osborne and sylvain kahane edn.

Tucker, Gordon H. 1997. A functional lexicogrammar of adjectives. *Functions of Language* 4(2). 215–250.

Tucker, Gordon H. 1998. *The Lexicogrammar of Adjectives: A Systemic Functional Approach to Lexis*. Bloomsbury.

Turing, Allan. 1950. Computing machinery and intelligence. *Mind* 59. 433–460.

Žolkovskij, Alexander K. & Igor A. Mel'čuk. 1967. O semantičeskom sinteze. *Problemy kibernetiki* 19(?). 177–238.

Weerasinghe, Ruvan. 1994. *Probabilistic Parsing in Systemic Functional Grammar*: University of Wales College of Cardiff dissertation.

West, Douglas Brent et al. 2001. *Introduction to graph theory*, vol. 2. Prentice hall Upper Saddle River.

Winograd, Terry. 1972. *Understanding natural language*. Orlando, FL, USA: Academic Press, Inc. <http://linkinghub.elsevier.com/retrieve/pii/0010028572900023>.

Zeman, Daniel, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis M. Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinková, Jaroslava Hlavácová, Václava Kettnerová, Zdenka Uresová, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar

Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria de Paiva, Kira Droganova, Héctor Martínez Alonso, Çagri Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali El-Kahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonça, Tatiana Lando, Rattima Nitisaroj & Josie Li. 2017. Conll 2017 shared task: multilingual parsing from raw text to universal dependencies. *Proceedings of the CoNLL Shared Task* 1–19.

Zhang, Niina Ning. 2010. *Coordination in syntax*. Cambridge University Press.

Zhang, Yue & Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies: short papers-volume 2*, 188–193. Association for Computational Linguistics.

Zwicky, Arnold M. 1985. Heads. *Journal of Linguistics* 21. 1–30.