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

June 2018

Table of contents

Chapter 1

Introduction

1.1 On Artificial Intelligence (AI) and Computational Linguistics

In 1950 Alan Turing in a seminal paper (Turing 1950) published in Mind was asking if "machines can do what we (as thinking entities) can do?" He questioned what intelligence was and whether it could be manifested in machine actions indistinguishable from human actions.

He proposed the famous *Imitation Game* also known as the *Turing test* in which a machine would have to exhibit intelligent behaviour equivalent or indistinguishable from that of a human. The test was set up by stating the following rules. The machine (player A) and a human (player B) are engaged in a written *natural language* conversation with a human judge (player C) who has to decide whether each conversation partner is human or a machine. The goal of players A and B is to convince the judge (player C) that they are human.

This game underpins the question whether "a computer, communicating over a teleprinter, (can) fool a person into believing it is human?", moreover, whether it can exhibit (or even appear to exhibit) human(-like) cognitive capacities (Stevan Harnad 1992). Essential parts of such cognitive capacities and intelligent behaviour that the machine needs to exhibit are of course the linguistic competences of comprehension (or "understanding") and generation of "appropriate" responses (for a given input from the judge C).

The *Artificial Intelligence* (AI) field was born from dwelling on Turing's questions. The term was coined by McCarthy for the first time in 1955 referring to the "science and engineering of making intelligent machines" (McCarthy et al. 2006).

The general target is to program machines to do with language what humans do. Various fields of research contribute to this goal. Linguistics, amongst others, contributes with theoretical frameworks systematizing and accounting for language in terms of morphology, phonology, syntax, semantics, discourse or grammar in general. In computer science increasingly more efficient algorithms and machine learning techniques are developed. Computational linguistics provides methods of encoding linguistically motivated tasks in terms of formal data structures and computational goals. In addition, specific algorithms and heuristics operating within reasonable amounts of time with satisfiable levels of accuracy are tailored to accomplish those linguistically motivated tasks.

Computational Linguistics (CL) was mentioned in the 1950 in the context of automatic translation (Hutchins 1999) of Russian text into English and developing before the field of Artificial Intelligence proper. Only a few years later CL became a sub-domain of AI as an interdisciplinary field dedicated to developing algorithms and computer software for intelligent processing of text (leaving the very hard questions of intelligence and human cognition aside). Besides *machine translation* CL incorporates a broader range of tasks such as *speech synthesis and recognition, text tagging, syntactic and semantic parsing, text generation, document summarisation, information extraction* and others.

This thesis contributes to the field of CL and more specifically it is an advancement in *Natural Language Parsing* (NLP), one of the central CL tasks informally defined as the process of transforming a sentence into (rich) machine readable syntactic and semantic structure(s). Developing a program to automatically analyse text in terms of such structures by involving computer science and artificial intelligence techniques is a task that has been pursued for several decades and still continues to be a major challenge today. This is especially so when the target is *broad language coverage* and even more when the desired analysis goes beyond simple syntactic structures and towards richer functional and/or semantic descriptions useful in the latter stages of *Natural Language Understanding* (NLU). The current contribution aims at a reliable modular method for parsing unrestricted English text into a feature rich constituency structure using Systemic Functional Grammars (SFGs).

In computational linguistics, broad coverage natural language components now exist for several levels of linguistic abstraction, ranging from tagging and stemming, through syntactic analyses to semantic specifications. In general, the higher the degree of abstraction, the less accurate the coverage becomes and, the richer the linguistic description, the slower the parsing process is performed.

Such working components are already widely used to enable humans to explore and exploit large quantities of textual data for purposes that vary from the most theoretical, such as understanding how language works or the relation between form and meaning, to very pragmatic purposes such as developing systems with natural language interfaces, machine translation, document summarising, information extraction and question answering systems to name just a few.

1.2 Living in technologically ubiquitous world

Developed over thousands of years, the human language has become a versatile highly nuanced form of communication that carries a wealth of meaning which by far transcends the words alone. When it comes to *human-machine* interaction this highly articulated communication form is deemed impractical. So far humans had to learn to interact with computers and do it in formal, strict and rigorous manner via graphical user interfaces, command line terminals and programming languages. Advancements in *Natural Language Processing* (NLP) which is a branch of *Artificial Intelligence* (AI) are a game changer in this domain. NLP starts to unlocks the information treasure locked in the human speech and make it available for processing to computers. NLP becomes an important technology in bridging the gap between natural data and digital structured data.

In a world such ours, where technology is ubiquitous and pervasive in almost all aspects of our life, NLP becomes of great value and importance regardless whether it materializes as a spell-checker, intuitive recommender system, spam filter, (not so) clever machine translator, voice controlled car, intelligent assistants such as Siri, Alexa or Google Now.

Every time you ask Siri or Alexa for directions to the nearest Peruvian restaurant, how to cook Romanian beef stew or what is the dictionary definition for the word "germane", a complex chain of operations is activated that allows 'her' to understand the question, search for the information you are looking for and respond in a human understandable language. Such tasks are possible only in the past few years thanks to advances in NLP. Until now we have been interacting with computers in a language they understand rather than us. Now they are learning our language.

1.3 NLP for business

NLP opens new and quite dramatic horizons for businesses. Navigating with limited resources stormy markets of competitors, customers and regulators and finding an optimal answer/action to a business question is not a trivial task. Markets are influenced by the information exchange and being able to process massive amounts of text and extract meaning can help asses the status of an industry and play an essential role in crafting a strategy or a tactical action. Relevant NLP tasks for gathering market intelligence are *named entity recognition* (NER), *event extraction* and *sentence classification*. With these tasks alone one can build a database about companies, people, governments, places, events together with positive or negative statements about them and run versatile analytics to audit the state of affairs.

Compliance with governmental, European or international regulations is a big issue for large corporations. One question for addressing this problem is whether a product is a liability or not and if yes then in which way. Pharma companies for example, once a drug has been released for clinical trials, need to process the unstructured clinical narratives or patient's reports about their health and gather information on the side effects. The NLP tasks needed for this applications are primarily *NER* to extract names of drugs, patients and pharma companies and *relation detection* used to identify the context in which the side effect is mentioned. NER task help transforming a sentence such as "Valium makes me sleepy" to "(drug) makes me (symptom)" and relation detection will apply patterns such as "I felt (symptom) after taking (drug)" to detect the presence of side effects.

Many customers, before buying a product, check online reviews about the company and the product whether it is pizza or a smartphone. Popular sources for such inquiry are the blogs, forums, reviews, social media, reports, news, company websites, etc. All of them contain a plethora of precious information that stays trapped in unstructured human generated text. This information if unlocked can play a great deal in company's reputation management and decisions for necessary actions to improve it. The NLP tasks sufficient to address this business required are *sentiment analysis* to identify attitude, judgement, emotions and intent of the speaker, and *co-reference resolution* which connects mentions of things to their pronominal reference in the following or preceding text. These tasks alone can extract the positive and negative attitudes from sentence "The pizza was amazing but the waiter was awful!" and connect it to the following sentence "I adore when it is topped with my favourite artichoke" about pizza and not the waiter and discover a topping preference.

NLP is heavily used in customer service in order to figure out what customer means not just what she says. Interaction of companies with their customers contain many hints pointing towards their dissatisfaction and interaction itself is often one of the causes. Companies record, transcribe and analyse large numbers of call recordings for extended insights. They deploy chat bots fo increased responsiveness by providing immediate answers to simple needs and also decrease the load of the help desk staff. NLP tasks that are essential in addressing some of the customer service needs are *speech recognition* that converts speech audio signal into text and *question answering* which is a complex task of recognising the human language question, extract the meaning, searching relevant information in a knowledge base and generate an ineligible answer. Advances in deep learning allow nowadays to skip the need for searching in a knowledge base by learning from large corpora of question-answer pairs complex interrelations.

The above cases underline the increased need in NLP whereas the variation and ever increasing complexity of tasks reveal the need in deeper and richer semantic and pragmatic analysis across a broad range of domains and applications. Any analysis of text beyond the formal aspects such as morphology, lexis and syntax inevitably lead to a functional paradigm of some sort which can be applied not only at the clause level but at the discourse as a whole. This makes the text also an artefact with relation socio-cultural context where it occurs.

1.4 The linguistic framework

Any description or analysis involving language implies some theory of about its essential nature and how it works. A linguistic theory includes also goals of linguistics, assumptions about which methods are appropriate to approach those goals and assumptions about the relation between theory, description and applications (Fawcett 2000: 3).

In his seminal paper "Categories of the theory of grammar" (Halliday 1961a), Halliday lays the foundations of *Systemic Functional Linguistic* (SFL) following the works of his British teacher J. R. Firth, inspired by Louis Hjelmslev (Hjelmslev 1953) from Copehagen School of linguistics and by a group of European linguists from Prague Linguistic Circle. This paper constitutes a response to the need for a *general theory of language* that would be holistic enough to guide empirical research in the broad discipline of linguistic science:

> . . . the need for a *general* theory of description, as opposed to a *universal* scheme of descriptive categories, has long been apparent, if often

> unformulated, in the description of all languages (Halliday 1957: 54; emphasis in original) . . . If we consider general linguistics to be the body of theory, which guides and controls the procedures of the various branches of linguistic science, then any linguistic study, historical or descriptive, particular or comparative, draws on and contributes to the principles of general linguistics (Halliday 1957: 55)

SFL regards language as a social semiotic system where any act of communication is regarded as a conflation of *linguistic choices* available in a particular language. Choices are organised on a paradigmatic rather than structural axis and represented as *system networks*. Moreover, in the SFL perspective language has evolved to serve particular *functions* influencing their the structure and organisation of the language. However, their organisation around the paradigmatic dimension leads to a significantly different functional organisation than those found in several other frameworks which Butler (2003a,b) addresses extensively. Also, making the paradigmatic organization of language a primary focus of linguistic description decreased the importance of the formal structural descriptions which from this perspective appear as realisation of (abstract) features.

Embracing the *oragnon model* formulated by Bühler (1934), Halliday refers to the language functions as metafunctions or lines of meaning that offer a trinocular perspective on language through *ideational*, *interpersonal* and *textual* metafunctions. In SFL, language is first of all an interactive action serving to enact social relations under the umbrella of the *interpersonal metafunction*. Then it is a medium to express the embodied human experience of inner (mental) and outer (perceived material) worlds via *ideational metafunction*. Finally the two weave together into a coherent discourse flow whose mechanisms are characterised through the *textual metafunction*.

Then a linguistic description is provided at various levels of granularity, that in SFL are is called *delicacy*. Just like the resolution of a digital photo defines the clarity and the amount of detail in the picture, the same way delicacy refers to the how fine- or coarse-grained distinctions are made in the description of the language. And if other linguistic traditions speak of a *grammar* in SFL the term *lexico-grammar* is used which, intuitively, is the combination of grammar and lexis explained in Section 3.1 into a unitary body. A deeper description of the SFL theory of language is provided latter in Chapter 3.

Until today, two major Systemic Functional Grammars (SFG) have been developed: the *Sydney Grammar* (Halliday & Matthiessen 2013) and the *Cardiff Grammar* (Fawcett 2008). The latter, as Fawcett himself regards it, is an extension and a simplification of

Sydney Grammar (Fawcett 2008: xviii). Each of the two grammars has advantages and shortcomings (presented in Chapter 3) which I discuss from the perspective of theoretical soundness and suitability to the goals of the current project.

Both Cardiff and Sydney grammars had been used as language models in natural language generation projects within the broader contexts of social interaction. Some researchers (Kasper 1988; O'Donoghue 1991; O'Donnell 1993; Souter 1996; Day 2007) attempted to reuse the grammars for the purpose of syntactic parsing within the borders of NL generation coverage. I come back to these works in more detail in Section 2.1.

To sum up, in this thesis I adopt the Systemic Functional Linguistic (SFL) framework because of its versatility to account for the complexity and phenomenological diversity of human language providing descriptions along *multiple semiotic dimensions* i.e. paradigmatic, syntagmatic, meta-functional, stratification and instantiation dimensions (Halliday 2003c) and at different *delicacy levels* of the *lexico-grammatical cline* (Halliday 2002; Hasan 2014). These notions and other elements of the SFL theory are addressed below in Chapter 3.

1.5 An example of Systemic Functional analysis

To provide a better intuition, this section describes an analysis of sentence using SFL framework. Here a parallel analysis between SFL and a more traditional grammar is provided in order to highlight the richness and high descriptive potential of SFL grammars. A source of the descriptive abundance is achieved through a practice of feature systematisation as mutually exclusive choices which is exemplified as well for three features of traditional grammar below. The feature analysis provided here is partial and restricted to only two constituents as this suffices to provide the reader with an intuition of what to expect from an SFL analysis.

Traditional linguistics teaches us how to carry on a syntactic analysis of a sentence. So let's consider Example 1 in order to perform one. First we would assign a *part of speech* (verb, noun, adjective etc.) to each word, then we would focus on clustering words into constituents guided by the intuitive rule *which words goes together*, and then those would receive syntactic functions (subject, predicate, complement etc.) within the sentence.

(1) He gave the cake away.

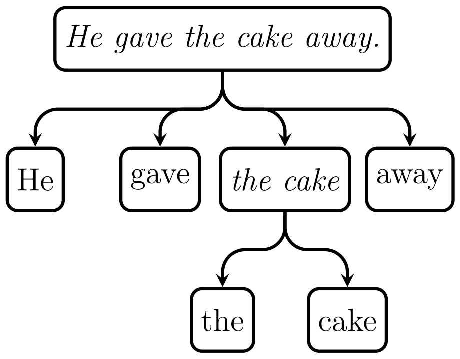


Fig. 1.1 Representation of the Example 1 as constituency tree

Figure 1.1 depicts the constituency division of the clause 1The nodes represent grammatical constituents and the edges stand for the *structure-substructure composition*. Next we can move on to assign constituent class and a grammatical function. Table 1.1 provides a constituency analysis in SFL tradition. Here the sentence is formed of a single clause which has four constituting functional parts: a Subject, a Main Verb (also known as Predicate), a Complement and an Adjunct. Each of these functional parts is filled correspondingly by a pronoun, a verb, a nominal group and an adverb as assigned in the table below.

He	*gave*	*the*	*cake*	*away.*
clause				
Subject	Main Verb	Complement		Adjunct
pronoun	verb	nominal group		adverb
		Deictic	Thing	
		determiner	noun	

Table 1.1 Constituency analysis with unit classes and grammatical functions

Next each constituent can be assigned a set of relevant linguistic features. For example The subject "He" is a pronoun that has features known in traditional grammar: *singular*, *masculine*, and *3rd person*. These features are well differentiated in traditional grammar. For example *singular* means *non-plural*, *masculine* means *non-feminine* and *3rd person* means *non-1st* and *non-2nd*. These are closed classes meaning that there is no *4th person* or that there is no *neutral* grammatical gender in English as other languages have. These features can be systematised (see Figure 1.2) as three systems of mutually exclusive choices that can be assigned to pronominal units. Note that the gender is enabled for 3rd person singular pronouns which can be expressed as is the figure below representing a *system network* (which is properly introduced in Chapter 3).

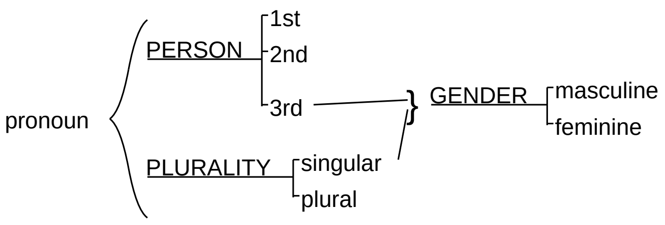


Fig. 1.2 The systematisation of three pronominal features in traditional grammar

In SFG the pronouns are systematised in the system network of Person from *Introduction to Functional Grammar* (Halliday & Matthiessen 2013: 366) that is depicted in Figure 1.3. The red rectangles from the figure represent the selections that are applicable to the Subject constituent "He" in example above.

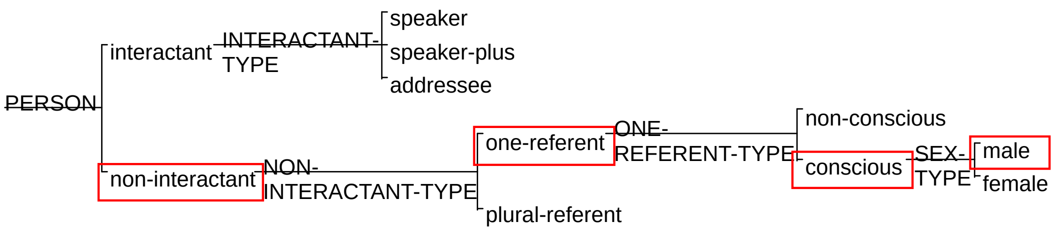


Fig. 1.3 The selections in Person system network from Halliday & Matthiessen (2013: 366)

Lets take now the clause constituent that is the root of the constituency tree and see how SFL features can be applied to it. If in in terms of traditional grammar the clause can be ascribed relatively few features i.e. as having *passive voice*, *positive polarity* and *simple past tense* then in terms of SFL grammar the features are many more i.e. *major, positive, active, effective, receptive, agentive, free, finite, temporal, past, non-progressive, non-perfect, declarative, indicative, mood-non-assessed, comment-non-assessed.* Figure 1.4 depicts the selections applicable to clause constituent in Example 1 from Mood system network that is an adaptation of Mood network proposed in Halliday & Matthiessen (2013: 162).

So far you have seen constituents assigned syntactic functions such as Subject, Complement, Adjunct etc. SFL covers a wider range of functions depending on the kind of meaning it aims at describing. For example what in other grammars is known as *semantic labels*, *thematic* or *θ roles* SFL systematises as Transitivity system network (which will be introduced in Chapter **??** below). Transitivity aims at providing domain independent *semantic frames* called in SFL *process configurations* which describe semantic actions and relationships, along with *semantic roles* ascribed to their *participants*. These semantic frames generally are governed by verbs and more specifically each verb meaning has a dedicated semantic frame.

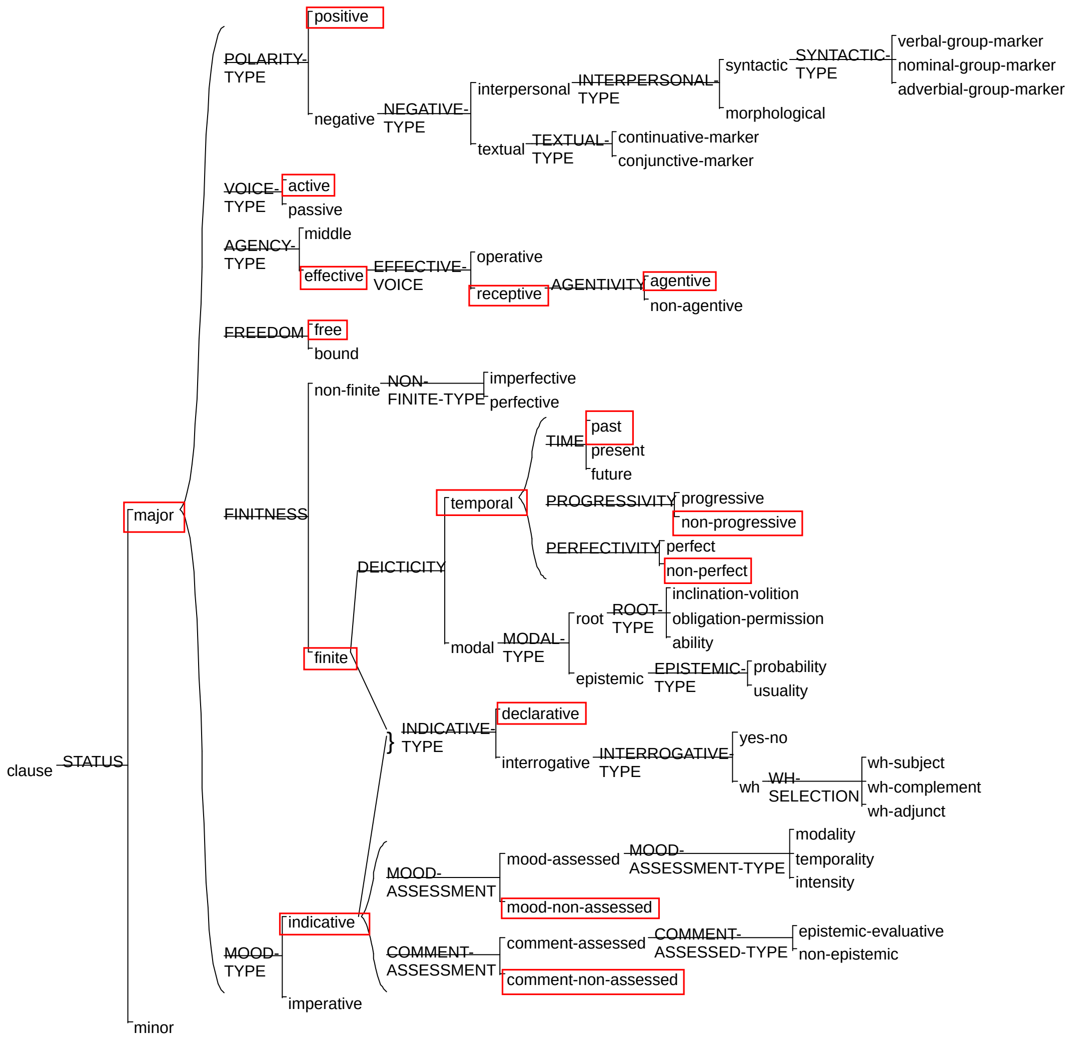


Fig. 1.4 The feature selections in the Mood system network for Example 1

For example 1 corresponds to a Possessive semantic frame where “He” is the Agent and Carrier whereas “the cake” is the Possessed thing as marked in Example 2. These configurations and participant roles correspond to Transitivity system network proposed in (Neale 2002).

(2) $[_{Agent-Carrier}$ He] gave $[_{Possessed}$ the cake] away.

There are more functions and features that can be assigned to the constituents in the Example 1 but I stop here. The analysis provided so far highlights that SFG grammar has a variety of functions serving to express different meanings. The traditional grammar distinguishes them as syntactic and semantic functions but, as we will see in Chapter 3 below, SFL does not make such a distinction. Another aim of the current section was to provide a glance of the feature rich grammar and I hope the example with Mood feature selection in Figure 1.4 fulfils this goal.

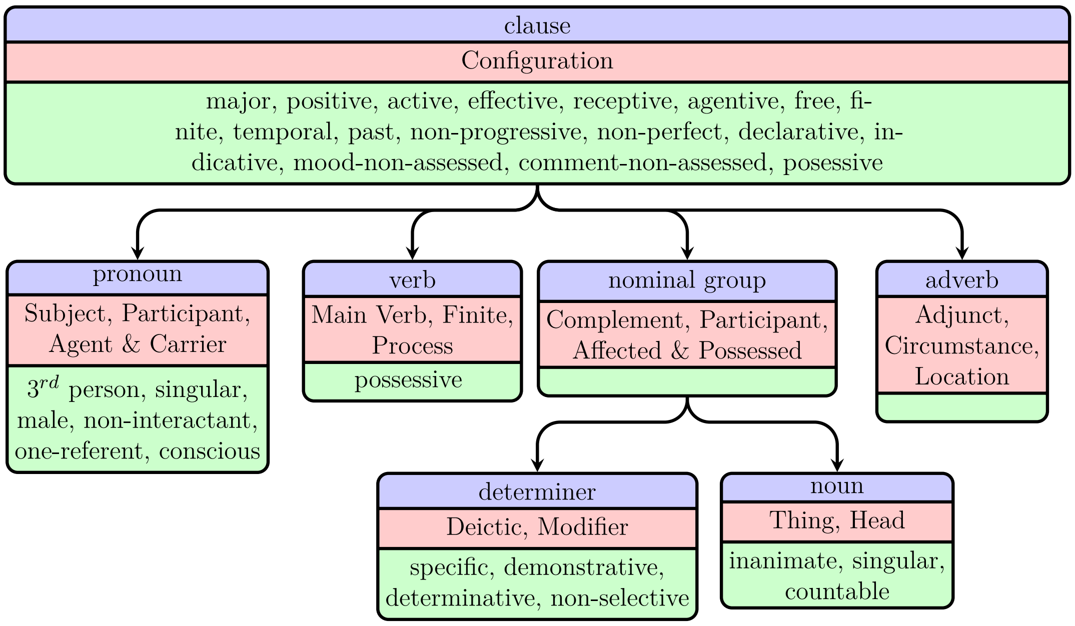

clause
Configuration
major, positive, active, effective, receptive, agentive, free, finite, temporal, past, non-progressive, non-perfect, declarative, indicative, mood-non-assessed, comment-non-assessed, posessive
pronoun
Subject, Participant, Agent & Carrier
3rd person, singular, male, non-interactant, one-referent, conscious
verb
Main Verb, Finite, Process
possessive
nominal group
Complement, Participant, Affected & Possessed
adverb
Adjunct, Circumstance, Location
determiner
Deictic, Modifier
specific, demonstrative, determinative, non-selective
noun
Thing, Head
inanimate, singular, countable

Fig. 1.5 Representation of Example 1 as feature rich constituency graph

Next, Figure 1.5 summarises everything discussed above into a partially filled constituency tree. The constituents that were not discussed are assigned only few high level functions and features. As you can see, every node is richly decorated with syntactic and semantic features. The blue part of each node denotes grammatical class, the red part carries functions some of which are important to establishing a valid constituency structure (that are Mood functions) and the the Transitivity functions; and the green part some grammatical features selected from system networks. In practice, the feature set is much richer than what nodes in Figure 1.5 carry here the restriction aims simply to avoid an over-crowded example.

Next I describe what opportunities and limitations exist in automatically generating rich SFL analyses as until now it has not been possible to use these detailed analysis in computational contexts. This makes them unavailable for corpus work, for training data in machine learning and other end-user application scenarios provided as motivation in the Sections 1.2 and 1.3 above.

1.6 The problem of parsing with SFGs

SFL since it has been established has been primarily concerned with the paradigmatic axis of language. Accounts of the syntagmatic axis of language, for example syntactic structure, have been put in the background. The structure has been placed on the theoretical map and defined in terms of *rank*, *unit*, *class* and *function*, as we will see in detail in Chapter 3 but afterwards it received minimal attention as most of the focus was on the paradigmatic organisation in language (in fact this is is the feature that sets SFL apart from other approaches to study language). This has led to progress in accounting how language works at all strata but little was said about the language constituency. And this can be considered "unsolved" within SFL accounts leaving a "gap in what must be one the central areas of any characterisation of language" (Bateman 2008: 25).

This may be surprising as the syntagmatic dimension is implicit and present everywhere in the SFL literature. For instance all example analyses in the *Introduction to Functional Grammar* (Halliday & Matthiessen 2013) are predominantly syntagmatic. Moreover, Robin Fawcett for decades promotes the motto *no system network without realisation statements* (Fawcett 1988b: 9). Bateman (2008) presents in detail why there is a severe imbalance between syntagmatic and paradigmatic axes in SFL, how it came to be this way and how it is especially damaging to the task of automatic text analysis. Next I describe the main problems and hint at how potential solutions may look like (described Section 1.8).

O'Donnell & Bateman (2005) offer a detailed description to the long history of SFL being applied in computational contexts yielding with productive outcomes on language theorising, description and processing. The transfer between SFL and computation typically involved a delay between the theoretical formulation and the computational instantiation of that formulation (Bateman & Matthiessen 1988: 139) (Matthiessen & Bateman 1991: 19). The theoretically formulated ideas contain hidden pitfalls that are revealed only upon explicit formulations required by in computation (Bateman 2008: 27).

The active exchange between the SFL theory and computation has been almost entirely oriented towards automatic *natural language generation*. Such systems would use abstract semantic specifications and communicative goals to produce grammatically correct and well connected texts achieving those goals. This area has been shown to be successful in part due to decomposition of language along the paradigmatic axis using functionally motivated sets of choices between functionally motivated alternatives (McDonald 1980).

The computational processes driving the natural language generation relied heavily on the notion of *search*. A well defined search problem is defined in terms of a precise description of the search space which then helps navigation process effectively to find solutions. The paradigmatic organization of the *lexicogrammar* as system networks assumed within SFL turns out to organise the search space for possible grammatical units appropriate for communicative goals in almost ideal manner (Bateman 2008: 28).

The *automatic analysis* or *parsing* can be seen as a reverse problem of finding appropriate analysis within a search space of possible solutions. That is identify the exact meaning, systematised in the grammar, of a given natural language sentence. As seen in Section 1.5 above, an account of the sentence meaning would have to provide both, in terms of a formal structure of the sentence revealing the constituents plus their syntactic relations to each other, and in terms of a complete set of features (detailed to the extent that grammar permits) applicable to each constituent of structure. If, in the generation process, the abstract semantic specifications are increasingly materialised through choice making by traversing the system network towards finally generated text, then, in the parsing process, the reverse is the case. The process starts from a given sentence aiming to derive/search the feature choices in the system network afferent to each of the constituents. But if the paradigmatically organised lexicogrammatical resource is effective for generation it turns out, as we will see next, to be by far unsuitable for the analysis task because of the *size problem*. Halliday himself mentions this problem when he asks *how big is a grammar?*.

> Given any system network it should in principle be possible to count the number of alternatives shown to be available. In practice, it is quite difficult to calculate the number of different selection expressions that are generated by a network of any considerable complexity (Halliday 1996: 10).

The issue emerges from the way connections and (cross-)classifications are organised in a system network. In addition to that, the orientation of systemic grammars towards choice means that the grammar full of disjunctions leading to the problem of search complexity. Also the abstract nature of systemic features leads to a structural richness that adds logical complexity to the task (O'Donnell 1993). So estimating the size of the grammar would in fact mean estimating the potential number of feature combinations. For example, a hypothetical network of 40 systems the "size of the grammar it generates lies somewhere between 41 and 2^{40} (which is somewhere around 10^{12})" (Bateman 2008: 28). However it is not easy to predict where would the upper limit of the grammar would fall even when the configuration of relations of a particular system network is known.

For the generation task, that size, is not a problem at all as the number of choice points is actually rather small. Such a paradigmatic organisation is, in fact, an incredibly concise and efficient way to express the linguistic choices where the possible feature selections are relevant only when they are enabled by prior paradigmatic choices and it is only those alternatives that need to be considered (Halliday 1996: 12–13) .

In the analysis task, the paradigmatic context of choice, that helps navigation during the generation process, is no longer available. It is not known any longer which features of a systemic network are relevant and which are not. This leads to a radical asymmetry between the two tasks. That is: in generation, the simple traversal of the network finds only the compatible choices because that is what the network leads to; whereas in analysis it is not evident in advance which path to follow therefore the task is virtually to explore entire search space in order to discover which features apply to the text. This means that any path is potentially relevant and shall be passed and checked leading to evaluation of the system network as a whole; and that there is no way to restricting the search space, as in the case of generation, to a a set of familiar paradigmatic lines (Bateman 2008: 29).

One of the grammars successfully used in generation tasks is Nigel grammar developed within Penman generation project (Mann 1983a). It contains 767 grammatical systems defined over 1381 grammatical features which Bateman evaluates as "a very large computational grammar by current standards, although nowadays by no means the broadest when considered in terms of raw grammatical coverage" (Bateman 2008: 29). To parse with such a grammar would means exploring an incredibly vast search space 3×10^{18} to be more precise. A more detailed break down the complexity by rank or primary class as provided in Table 1.2 below.

rank or primary class	*size*
adverbial-group	18
words	253
quantity-group	356
prepositional-phrase	744
adjectival-group	1045
nominal-group	$>2 \times 10^9$
clause	$>3 \times 10^{18}$

Table 1.2 Size of major components of the Nigel grammar expressed in terms of the number of selection expressions generated (Bateman 2008: 35)

Another difficulty in parsing with SFGs lays in the fact that, as the analysis moves away from directly observable grammatical variations towards more abstract semantic variations, the difficulty of generating an accurate account increases drastically. Transitivity system network for example consists of such semantic features and it is comparable to what is called in computational linguistics (shallow) *semantic parsing* or *Semantic Role Labelling* (SRL) (Carreras & Màrquez 2005).

The main challenge of SRL (well explained in (Gildea & Jurafsky 2002: 245–250)) remain the same since Winograd (1972): moving away from the domain specific, hand-crafted semantic specifications towards domain independent and robust set of semantic specifications. This goal was undertaken in several projects to build large broad-scope lexico-semantic databases such as WordNet (Fellbaum & Miller 1998), FrameNet (Baker et al. 1998; Johnson & Fillmore 2000; Fillmore et al. 2003) and VerbNet (Schuler 2005; Kipper et al. 2008). A similar database exists for Transitivity system network as described in Fawcett (2009) called Process Type Database (Neale 2002).

Such databases describe domain independent *semantic frames* (Fillmore 1985) (know in SFL as *configurations* or *figures*) which describe semantic actions and relationships, along with *semantic roles* ascribed to their *participants*. The semantic frames generally are governed by verbs and more specifically each verb meaning has a dedicated semantic frame. For instance the perception frame contains *Perceiver* and *Phenomenon* roles as can be seen in Example 3.

(3) [$_{Agent-Perceiver}$ Jaqueline] glanced [$_{Phenomenon}$ at her new watch].

The tendency is to identify frames that are generic enough to cover classes of verb meanings (for example Action, Cognition, Perception, Possession frames) and the same applies to participant roles where the tendency is to reuse roles across semantic frames (for example agent role from Action frame is reused in Perception or Possession frames, or Phenomenon is reused in Cognition and Perception frames).

Besides the challenge of pinning them down in text, the problem with semantic features goes one step further. Sometimes the semantic roles correspond to constituents that are displaced or not even realised in the text. This increases the challenge of identifying and assigning them correctly. Consider Example 4. It is a sentence that has three non-auxiliary verbs: seem, worry and arrive. According to Cardiff grammar, which will be introduced in Chapter 3, this corresponds to three clauses *embedded* into each other as represented in Table 1.3.

(4) She seemed to worry about missing the river boat.

The participant role configurations (or the semantic frames) these verbs bring about are provided in the Table 1.4. For the sake of this example lets say that the first role

She	*seemed*	*to*	*worry*	*about*	*missing*	*the*	*river*	*boat.*
clause								
Subject	Main Verb	Complement						
		clause						
		Infinitive Element	Main Verb	Complement				
				clause				
				Binder	Main Verb	Complement		

Table 1.3 SF constituency analysis in Cardiff grammar style

corresponds to the Subject constituent and the second to the Complement constituent. This way the verb meaning seem$_1$ corresponds to an Attributive configuration that distributes Carrier and Attribute roles to the Subject “She” and the Complement “to worry about missing the river boat”. In the case of worry about$_1$ and miss$_1$ the first roles provided by Cardiff grammar are *compound* (i.e. composed of two simple ones) while the second ones are simple. So, in the example above, the verb meaning worry about$_1$ distributes the Phenomenon to the Complement “about missing the river boat” and the Agent & Cognizant role to an empty Subject that is said to be *non-realised*, *covert* or *null element*. A similar situation is for miss$_1$ that assigns an Affected & Carrier role to the empty Subject and the Possessed role to the Complement “the river boat”.

verb meaning	*semantic configuration*	*participant role mandatory distribution*
seem$_1$	Attributive	Carrier + Attribute
worry about$_1$	Two Role Cognition	Agent & Cognizant + Phenomenon
miss$_1$	Possessive	Affected & Carrier + Possessed (thing)

Table 1.4 Semantic role configurations according to Neale (2002); Fawcett (2009)

Those unrealised Subjects in the embedded clauses are recoverable from the immediate syntactic context (no need for discourse) and correspond, in this case, to the Subject in the higher clause. This is easy to see in Examples 5 and 6 therefore we can just mark the places of the null Subjects in the embedded clause in order to be able to assign the semantic labels (otherwise the frame cannot be assigned to the constituents). Notice that I have provided also an index *i* to highlight that the null elements correspond to the higher clause Subject “She”.

(5) *She* worried about missing the river boat.

(6) *She* missed the river boat.

(7) She$_i$ seemed [*null-Subject*$_i$ to worry [about *null-Subject*$_i$ missing the river boat]].

Now that the places of covert constituents are explicitly marked and the recoverable constituent coindexed we can see the distribution of semantic roles realised for this sentence in the Table 1.5 below.

She$_i$	*seemed*	*null$_i$*	*to*	*worry*	*about*	*null$_i$*	*missing*	*the*	*river*	*boat.*
Attributive configuration										
Agent		Attribute								
		Two Role Cognition configuration								
		Agent & Cognizant			Phenomenon					
					Possessive configuration					
						Affected & Carrier		Possessed		

Table 1.5 Transitivity analysis in Cardiff grammar style (Neale 2002; Fawcett 2009)

In language there are many cases where constituents are empty but recoverable from the immediate vicinity relying in most cases on syntactic means and in a few cases additional lexical-semantic resources are required. The mechanisms of detecting and resolving the empty constituents are captured in the Government and Binding Theory (GBT) developed in (Chomsky 1981, 1982, 1986) and based on the phrase structure grammar. GBT explains how some constituents can *move* from one place to another, where are the places of *non-overt constituents* and what constituents do they refer to i.e. what are their *antecedents*. Such accounts of empty elements are missing from any SFG grammar yet they are useful in determining the correct distribution of participant roles to the clause constituents. Translating the mechanisms from GBT into SFG could contribute to decreasing the complexity of the parsing problem mentioned above.

To conclude, I have drown attention to a few issues related to parsing with SFG. First, the parsing task cannot be treated as a reversible generation task because the methods that have been shown to work for generation are not usable for parsing as such due to a high computational complexity. Second, the parsing task, regardless of the grammar, should first and foremost account for the sentence structure on the syntagmatic axis and only afterwards for the (semantic) features selected on the paradigmatic axis. Such syntagmatic account in SFL is insufficient for the parsing task. Third, syntagmatic account alone does not provide enough clues for assignment of semantic features and require a lexical-semantic account within the grammar or as external semantic databases. Moreover it can be aided by identification of places where covert constituents are said to exist. Identifying such the null elements is not the only method of assigning semantic features and some approaches do without them but having access to such information is considered valuable in the present thesis.

Considering the problems above how could the vast search space of grammars such as Nigel be restricted to a reasonable size and how can be compensated the lack of proper syntagmatic description in SFGs? The first part of the question has already been addressed in O'Donnell (1993) but the lack for an answer to the second part and probably for other hidden reasons the results are not usable in real world applications. There have been, in fact, multiple attempts such as Kasper (1988), Kay (1985), O'Donoghue (1991), O'Donnell (1993) and Day (2007), to mention just a few, none of which managed to parse broad coverage English with full SFG without aid of some sort. Each had to accept limitations either in grammar or language size and eventually used simpler syntactic trees as a starting point of the parsing process. A detailed account of the current state of the art in parsing with SFGs is provided in Chapter 2.

Some linguistic frameworks, other than SFL, have been shown to work well in computational contexts solving problems similar to the ones identified above. Instead of attempting to find novel solutions within the current framework an alternative approach, I argue in the next section, would be to establish cross-theoretical and inter-grammatical links to enable integration of the ready solutions.

1.7 On theoretical compatibility and reuse

In the past decades many significant progresses have been made in natural language parsing framed in one or another linguistic theory each adopting a distinct perspective and set of assumptions about language. The theoretical layout and the available resources influence directly what is implemented into the parser and each implementation approach encounters challenges that may or may not be common to other approaches in the same or other theories.

Parsers implementing one theoretical framework may face common or different challenges to those implementing other frameworks. The converse can be said of the solutions. When a solution is achieved using one framework it is potentially reusable in other ones. The successes and achievements in any school of thought can be regarded as valuable cross theoretical results to the degree links and correspondences can be established. Therefore reusing components that have been shown to work and yield "good enough results" is a strong pragmatic motivation in the present work.

This thesis employs three linguistic frameworks namely the *Systemic Functional Linguistics*, *Dependency Grammar* and *Governance & Binding Theory*. SFL has already been motivated as target analysis framework in Section 1.4 which is in detail introduced in Chapter 3. The other two frameworks are employed because some of the accomplishments in those domains I attempt at reusing in this thesis as motivated below.

In the last years *Dependency Grammar* (Tesniere 2015) became quite popular in natural language processing world favoured in many projects and systems. The grammatical lightness and the modern algorithms implemented into dependency parsers such as Stanford Dependency Parser (Marneffe et al. 2006), MaltParser (Nivre 2006), MSTParser (McDonald et al. 2006) and Enju (Miyao & Tsujii 2005) are increasingly efficient and highly accurate. Among the variety of dependency parsing algorithms, a special contribution bring the *machine learning* methods such as those described in McDonald et al. (2005); McDonald & Pereira (2006); Carreras (2007); Zhang & Nivre (2011); Pei et al. (2015) to name just a few.

As the dependency parse structures provide information about functional dependencies between words and grants direct access to the predicate-argument relations and can be used off the shelf for real world applications. This information alone makes the dependency grammar a suitable candidate to supplement the syntagmatic account missing in SFGs and provide some functional hooks for reducing complexity in parsing with SFGs. One of the goals in this work is to investigate to which degree the dependency grammar is structurally and functionally compatible with SFGs to undergo a cross theoretic transformation. This hypothesis is investigated at the theoretical level in Chapter 4 and then indirectly evaluated empirically in Chapter 9 based on Stanford Dependencies parser version 3.5 (Marneffe & Manning 2008b,a; Marneffe et al. 2014).

The problem of accounting for the *null elements*, mentioned above, is not addressed neither in SFL nor in Dependency Grammar. It is, however, in detail addressed in the Government and Binding Theory (GBT) (Chomsky 1981; Haegeman 1991) which is one of Chomsky's Transformational Grammars (Chomsky 1957a). One other goal in this thesis is to investigate to which degree GBT accounts of null elements can be reused as DG or SFG structures to undergo a cross-theoretic transformation enabling those accounts in DG or SFG contexts. In Chapter 5 I introduce GBT and investigate this hypothesis providing some of the cross-theoretic and inter-grammatical links to Dependency and SFL grammars that as we will see in Chapter 8 benefits the Transitivity analysis.

1.8 Thesis Goals and Proposed Solution

This thesis aims at a reliable modular method for parsing unrestricted English text into a feature rich constituency structure using Systemic Functional Grammar (SFG).

As will be described in Chapter 2.1, some parsing approaches use a syntactic backbone which is then flashed out with an SFG description. Others use a reduced set or a single layer of SFG representation; and the third group use an annotated corpus as the source of a probabilistic grammar. Regardless of approach, each limits the SFG in one way or another, balancing the depth of description with language coverage: that is either *deep description but a restricted language* or *shallow description but broad language coverage* is attempted. The current thesis tilts towards the latter: while keeping the language coverage as broad as possible the aim is to provide, in the parse result, as many systemic features as possible.

The process developed in this thesis can be viewed as a pipeline architecture (see Section 1.8.3) comprising of two major phases: the *structure creation* and the *structure enrichment*. The structure creation phase aims to account for the syntagmatic dimension of language.

The structure enrichment phase aims at discovering and assigning systemic features (accounting for the paradigmatic dimension of language) afferent to each of the nodes constituting the structure. In this phase, two kinds of feature enrichments can be distinguished by the kinds of clues used for feature identification. The first kind of clues are syntagmatic (constituency tree, unit class, unit function, linear order, position) and can be detected using, what I call, the *structural patterns* while the second kind of clues are lexical-semantic requires lexical-semantic and potentially more kinds of resources in addition to the structural patterns.

1.8.1 Towards the syntagmatic account

The problem in using SFGs for parsing, as we have seen in Section 1.6 above, manifests when the grammar is instantiated computationally with a primary focus on paradigmatic organisation (prevalent in SFL) at the cost of syntagmatics which leads to the first difficulty that needs to be addressed: discovering from a sequence of words what possible groups are combinable into grammatical groups, phrases or clauses. This is a task of bridging a sequence of words as input and the grammatical description of how they can combine to form a (syntactic) constituency tree structure (known in SFL as *syntagmatic organizations* which will be addressed in Section 3.2.2).

This challenge can be addressed by filling the gap of the syntagmatic account within the SFL grammar directly. It involves, first, providing information about which grammatical functions operate at each rank, second, which grammatical functions can be filled by which classes of units and, third, providing relative and absolute description of the element order for each unit class. This information in the grammar can guide the process of building the constituency structure.

Alternatively the problem of structure construction can be outsourced as parsing with other grammars. This is done in the works of Kasper Kasper (1988) and Honnibal (2004); Honnibal & Curran (2007) and is knows in SFL literature as *parsing with a syntactic backbone*. In this case, the problem changes into creating a transformation mechanism to obtain the SFL constituency structure rather than build it from scratch.

This thesis addresses the problem of building the constituency structure by the latter approach: parsing the text with Stanford Dependencies parser version 3.5 (Marneffe & Manning 2008b,a; Marneffe et al. 2014) and then transforming the parse result into SFG constituency tree. The transformation mechanisms from one grammar into the other one requires also a theoretical discussion in terms of what is being transformed. Such account of linguistic primitives or configurations of primitives in the source grammar corresponds to SFG primitives is provided in Chapter 4.

The SFG constituency structures is generated through a process that involves: traversing the source (dependency) parse tree and, at each traversal step, executing a constructive operation on a parallel tree following a predefined rule set of operations and mapping relations. The detailed description of the structure generation process is provided in the Chapter 7.

1.8.2 Towards the paradigmatic account

Once the constituency structure is in place it can inform the following feature derivation process. The configurations of units of specific classes and carrying grammatical functions can operate as "hooks" on system network to guide the traversal in the same way the paradigmatic context available in the generation process. Such configurations resemble the SFG *realization rules* which, in the generation process, instantiate the (abstract) features into text.

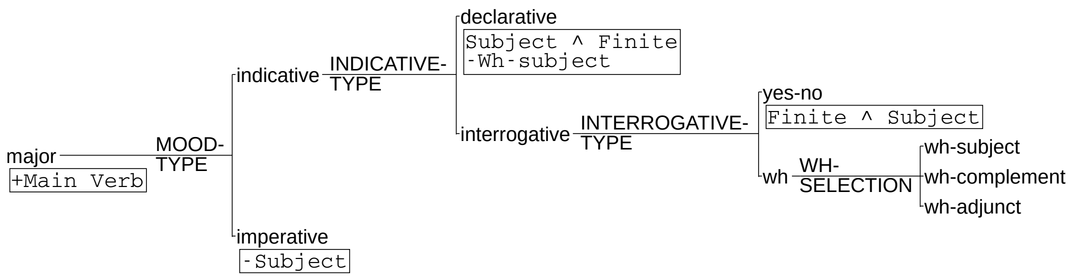


Fig. 1.6 A fragment of mood system from Halliday & Matthiessen (2013: 366)

The system network fragment in Figure 1.6 contains the realisation rules positioned in rectangles below a few features. These realisation rules indicate what shall be reflected in the structure when a feature is selected (discussed already in Section 1.6). The converse is also true: if structure contains a certain pattern then it is a (potential) manifestation of a given feature.

For example the structure of a *major* clause needs to have a predicate or Main Verb element realised. In the parsing process, testing whether there is a unit functioning as Main Verb below in the clause node suffices to assign the *major* feature to that clause. Next, if the clause has no unit functioning as Subject then it shall be assigned *imperative* feature otherwise the *indicative* one. Further the INDICATIVE-TYPE system is enabled. Here the test is whether a Subject node is positioned in front of the Finite node and whether the Subject contain the preposition "who". This sort of queries on the structure can be formulated as *structure patterns* (see Section 6.2) and associated to features in the system network in the same manner the realisation rules are.

The pattern recognition plays an essential role in current parsing method for fleshing out the constituent backbone with systemic selections. In the parsing process the structural patterns are tested whether they *match* (see Section 6.6) anywhere in the constituency structure and if so then the matched nodes are enriched with the features proved in the pattern (described in Section 6.7).

The structural patterns in this work are expressed as *graph patterns* (described in Section 6.2). Note that I employ graph and not tree patterns because the tree patterns are too restrictive for the purpose of the current work. While most of the time they are hierarchically structured as a tree there are few patterns that involve sibling connections or nodes with more than one parent. In both cases the tree structure is broken. The graph approach allows a wide range of structures which also includes the trees.

The enrichment stage of the parsing process comprises of a series of graph pattern matching operations that in case of success leads to the enrichment of the constituency structure with the features given in the graph pattern. This mechanism is described in detail in the Section 7.5.

In this work most of the graph patterns have been manually created. Because this is laborious exercise only a few system networks have been covered in the implementation of the parser. Nonetheless they suffice for deriving some conclusions regarding the parsing approach. The future work may investigate how can graph patterns be generated automatically from the realisation rules of large grammars such as Nigel grammar.

The two main system networks targeted in this are MOOD and TRANSITIVITY (both briefly described in Chapter 3). The MOOD network is composed of features which can be identified through graph patterns involving only the unit classes and functions provided in the constituency structure (described in Chapter 7). The TRANSITIVITY network requires a lexical-semantic database in order to derive graph patterns. This work employs the Process Type Database (PTDB) (Neale 2002) to aid enrichment with TRANSITIVITY features described in Chapter 8.

1.8.3 The Implementation Architecture

The current thesis is accompanied by a software implementation called Parsimonious Vole parser. It is written using Python programming language and is available as open source distribution (https://bitbucket.org/lps/parsimonious-vole).

The parser follows a pipeline architecture depicted in Figure 1.7 where, starting from an input text, a rich systemic functional constituency structure is progressively built. This section provides an overview to the construction process.

The figure provides three types of boxes. The rounded rectangles represent the parsing steps. The parsing steps linearly flow from one to the next via green trapezoid boxes on the left-hand side, which represent input-output data data intermediating the processes. On the right-hand side are positioned double edged orange trapezoids representing fixed resources used as additional input for some steps. For example, the *constituency graph creation* step takes a normalised dependency graph for input and produces a constituency graph as output.

On the right-hand side a series of green vertical arrows are provided naming phases in parsing process (spanning one ore more process steps) where the first thee i.e. Bootstrapping, Pre-processing and Graph Building accomplish construction of the constituency backbone (corresponding to the syntagmatic account described in Section 1.8.1 above) and the last phase *Graph Enrichment* (spanning three process steps) flashes out the backbone with features (described in Section 1.8.2).

The parsing process starts with an input English text and ends with production of a Rich Constituency Graph. Input Text is first parsed with the Stanford dependency parser version 3.5 resulting in a Dependency Graph.

The dependency graphs often contain errors, however. Some of these errors are predictable, and so easy to identify and correct. Also, in addition, some linguistic phenomena are treated in a slightly different manner than that proposed in the current thesis. Therefore the dependency graph produced by the Stanford parser is Corrected

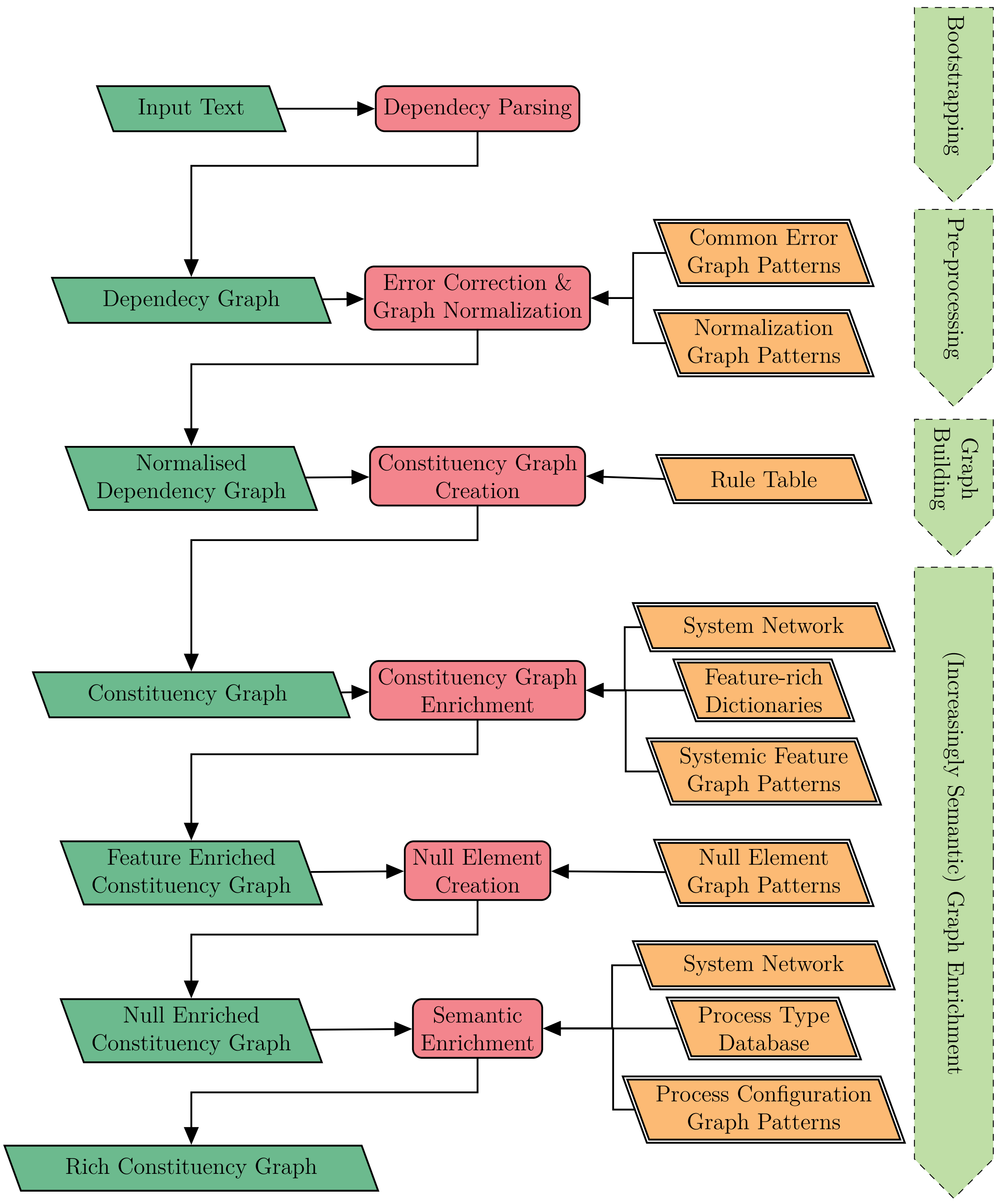

Input Text
Dependecy Parsing
Dependecy Graph
Error Correction & Graph Normalization
Common Error Graph Patterns
Normalization Graph Patterns
Normalised Dependecy Graph
Constituency Graph Creation
Rule Table
Constituency Graph
Constituency Graph Enrichment
System Network
Feature-rich Dictionaries
Systemic Feature Graph Patterns
Feature Enriched Constituency Graph
Null Element Creation
Null Element Graph Patterns
Null Enriched Constituency Graph
Semantic Enrichment
System Network
Process Type Database
Process Configuration Graph Patterns
Rich Constituency Graph
Bootstrapping
Pre-processing
Graph Building
(Increasingly Semantic) Graph Enrichment

Fig. 1.7 The parsing process pipeline

and Normalised using pattern matching against collections of known errors and a set of normalization rules.

Once normalised the dependency graph is ready to guide the *building process* of the systemic functional constituency graph. Through a traversal of the dependency graph the constituency graph is constructed using a mapping rule set. The mappings indicate what operation to perform on the newly emerging graph given the visited dependency node and the incoming and outgoing dependency relations. So the constituency graph is, in a way, a transformation of the dependency graph, and serves later as a syntactic backbone on which the subsequent enrichment phases are performed.

Next follow two phases where the syntactic backbone is *enriched* with features, some of which are of a *syntactic* and others of a *semantic* nature. In between these enrichment phases there is a construction process which produces structural changes to the backbone adding some *empty constituents* that play a role in semantic enrichment. The enrichment phases use additional resources such as *system networks*, *feature rich lexicons*, *graph patterns* and *semantic databases*. The *null element creation* process also needs a collection of graph patterns for identifying where and what kind of null elements occur as motivated in Section 1.6 and explained in detail in Chapter 5. The final result of the process is a *Rich Constituency Graph* of the original text comprising a substantial set of systemic feature selections associated with constituting units of structure. The next section provides an

1.8.4 Research questions and contributions

This thesis addresses the following questions:

- To what extent techniques from other areas of computational linguistics can be reused for the SFL parsing?
- To what degree the syntactic structures of the Dependency Grammar and Systemic Functional Grammar are compatible to undergo a transformation from one into the other?
- Is Stanford dependency grammar suitable as a syntactic backbone for Systemic Functional Grammar parsing?
- Can the Process Type Database be used as a resource for SFG Transitivity parsing?
- How can Government and Binding Theory be used for detecting places of null elements in the context of SFL constituency structure?

Also it brings the following theoretical and practical contributions:

- A set of theoretical principles and generalizations establishing cross-theory links between Dependency Grammar and SFL.
- A set of mapping rules between Stanford Dependency v3.5 to SFG constituency structure.
- A parallel graph construction process for creating SF syntactic backbone.
- A flexible and expressive method to represent systemic features as graph patterns and a strategy for choice propagation in the systemic networks.
- A set of pattern graphs covering Mood, Transitivity and a few other small system networks.
- A clean machine readable version of the PTDB.
- A method to transform PTDB into Transitivity graph patterns.
- Several principles for detecting null elements in the sentence which were translated from the Government and Binding Theory (GBT) into Dependency and SFL terms.
- Implementation of the translated GBT principles as graph patterns corresponding usable to identify the null elements in a sentence.
- A small test corpus to evaluate the Parsimonious Vole parser.

1.9 Provisional Thesis Structure

1: introduction
2: SFG parsing problem (simple explanation) and SOTA
3: Architecture presentation, introducing challenges of every step and make future references, also the structure of the thesis.
4: SFL grammar
5: Dep Grammar
5: GBT (has to stay here somehow)
7: the structure creation (introducing basic computer scientific definitions of data structures and other needs) (introducing the exact SFG grammar constituents)
8: the syntactic feature enrichment (introducing basic computer scientific definitions of data structures and other needs) (introducing exact Mood features)
9: the semantic feature enrichment (introducing basic computer scientific definitions of data structures and other needs) (introducing exact Cardiff Transitivity features)
10: Empirical Evaluation
11: Conclusions (what has been achieved and outlook)

Chapter 2

State of the art review

2.1 Previous works on parsing with Systemic Functional Grammars

There have been various attempts to parsing with SFGs. This section covers the most significant attempts to parse with a Systemic Functional Grammar. The firs attempt was made by Winograd (Winograd 1972) which was more than a parser, it was an interactive a natural language understanding system for manipulating geometric objects in a virtual world.

Starting from early 1980s onwards, Kay, Kasper, O'Donell and Bateman tried to parse with Nigel Grammar (Matthiessen 1985), a large and complex natural language generation (NLG) grammar for English used in Penman generation project. Other attempts by O'Donoghue (1991), Weerasinghe (1994), Souter (1996), Day (2007) aim for corpus based probability driven parsing within the framework of COMMUNAL project starting from late 1980s.

In a very different style, Honnibal (2004); Honnibal & Curran (2007) constructed a system to convert Penn Treebank into a corresponding SFGBank. This managed to provide a good conversion from phrase structure trees into systemic functional representation covering sentence mood and Thematic constituency (a kind of analysis in SFL which is not considered in current work). Transitivity has not been covered there because of its inherently semantic nature but it is in the current work.

2.1.1 Winograd's SHRDLU

SHRDLU is an interactive program for understanding (if limited) natural language written by Terry Winograd at MIT between 1968-1970. It carried a simple dialogue about a world of geometric objects in a virtual world. The human could as the system to manipulate objects of different colours and shapes and the ask questions about what has been done or the new state of the world.

It is recognised as a landmark in natural language understanding demonstrating that a connection with artificial intelligence is possible if not solved. However, his success was not due to the use of SFG syntax but rather due to small sizes of every system component to achieve a fully functional dialogue system. Not only it was parsing the input but it was developing an interpretation of it, reason about it and generate appropriate natural language response.

Winograd combined the parsing and interpretation processes such that the semantic interpreter was actually guiding the parsing process. The knowledge of syntax was encoded in the procedures of interpretation program. He also implemented an ingenious backtracking mechanism where the the program does not simply go back, like other parsers, to try the next possible combination choice but actually takes a decision on what shall be tried next.

Having data embedded into the program procedures, as Winograd did, makes it non-scalable for example in accommodation of larger grammars and knowledge bodies and unmaintainable on the long term as it becomes increasingly difficult to make changes (Weerasinghe 1994).

2.1.2 Kasper

Bob Kasper in 1985 being involved in Penman generation project embarked on the mission of testing if the Nigel grammar, then the largest available generation grammar, was suitable for natural language parsing. Being familiar with Functional Unification Grammar (FUG), a formalism developed by Kay and tested in parsing (Kay 1985) which caught on popularity in computational linguistics regardless of Kay's dissatisfaction with results, Kasper decided to re-represent Nigel grammar into FUG.

Faced with tremendous computational complexity, Kasper (1988) decided to manually create the phrase-structure of the sentences with hand-written rules which were mapped onto a parallel systemic tree structure. Kasper in 1988 was the first one to parse with a context-free backbone. He first parsed each sentence with a Phrase Structure Grammar (PSG), typical to Chomsky's Generative Transformational Linguistics

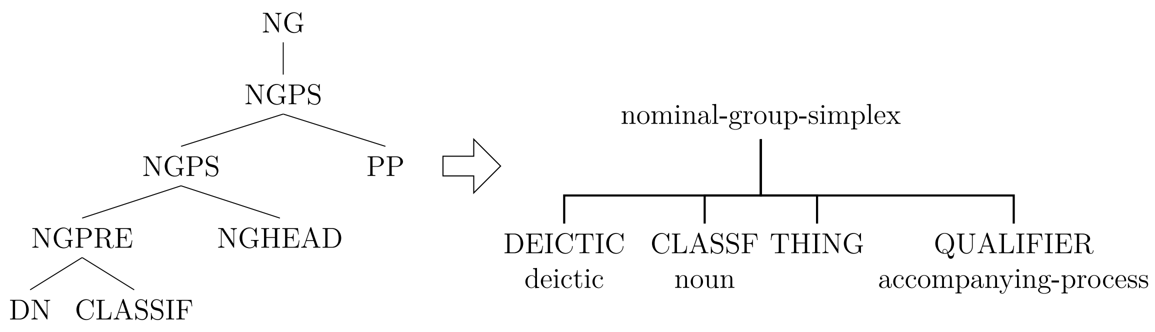


Fig. 2.1 Transformation from phrase structure into systemic constituency structure. Rule example from O'Donnell & Bateman (2005).

Chomsky (1957a). He created a set of rules for mapping the phrase structure (PS) into a parallel systemic tree like the one depicted in Figure 2.1. When all possible systemic tree were created they were further enriched using information from Nigel Grammar (Matthiessen 1985).

Once the context-free phrase-structure was created using bottom-up chart parser it was further enriched from the FUG representation of Nigel grammar. This approach to parsing is called *parsing with a context-free backbone* as phrase-structure is conveyed as simplistic skeletal analysis, fleshed out by the detail rich systemic functional grammar.

Even though Kasper's system is represents the first attempt to parse with full Hallidayan grammar, it's importance is lowered, as O'Donnell & Bateman (2005) point out, by the reliance on phrase structure grammar.

2.1.3 O'Donnell

Since 1990, Mick O'Donnell experimented with several parsers for small Systemic grammars, but found difficulty when scaling up to larger grammars. While working in EAD project, funded by Fujitsu, he recompiled a subset of Nigel grammar into two resources: the set of possible function bundles allowed by the grammar (along with the bundles preselections) and a resource detailing which functions can follow a particular function (O'Donnell 1993, 1994).

This parser was operating without a syntactic backbone directly from a reasonable scale SFG. However when scaled to the whole Nigel grammar the system became very slow because of the sheer size of the grammar and its inherent complexity introduced by multiple parallel classifications and functional combinations - a problem well described by Bateman (2008). Then O'Donnell wrote his own grammar of Mood that was more suitable for the parsing process and less complex than the recompiled Nigel.

In 2001, while working in a Belgian company O'Donnell came to conclusion that dependency grammars are very efficient for parsing. Together with two colleagues, he developed a simplified systemic grammar where elements were connected through a single function hence avoiding (functional) conflation. Also the ordering of elements was specified relative to the head rather than relative to each other.

More recently, O'Donnell in UAM Corpus Tool embedded a systemic chart parser (O'Donnell 2005) with a reduced systemic formalism. He classifies his parser as a left to right and bottom up with a custom lexicon where verbs are attributed features similar to Hallidayan process types and nouns a unique semantic category like thing-noun, event-noun, location-noun etc.

Because of previously reported complexity problems (O'Donnell 1993) with systemic grammars, the grammatical formalism is reduced to a singular functional layer of Mood-based syntactic structure (Subject, Predicate, Object etc.) ignoring the Transitivity (Actor/Goal, Sensor/Phenomenon etc.) and Textual (Theme/Rheme) analyses. O'Donnell deals away with the conflation except for the verbal group system network. He also employs a slot based ordering where elements do not relate to each other but rather to the group head only simplifying the number of rules and calculation complexity.

In his paper (O'Donnell 2005) does not provide a parser evaluation so its accuracy is still unknown today. The lexicon that was created is claimed to deal with word semantic classes but it is strongly syntactically based assigning a single sense to nouns and verbs ignoring the peculiar aspect of language polysemy. Moreover it is not very clear the framework within which the semantic classes have been generated.

2.1.4 O'Donoghue

O'Donoghue proposes a corpus based approach to parsing using *Vertical Strips* (O'Donoghue 1991). They are defined as a vertical path of nodes in a parse tree starting from the root down to the lexical items but not including those. He extracted the set of vertical strips from a corpus called Prototype Grammar Corpus together with their frequencies and probability of occurrence. This approach differ from the traditional one with respect to the kind of generalization it is concerned and specifically, the traditional approached are oriented towards horizontal order while the vertical strip approach is concerned with vertical order in the parse tree.

To solve the order problem O'Donoghue uses a set of probabilistic collocation rules extracted from the same corpus indicating which strips can follow a particular strip. He also created a lexical resource indicating for each word which elements can expanded it.

The parsing procedure is a simple lookup of words in the lexical resource selecting all possible elements it can expound and then selecting possible strips starting with the elements expounded by the word. Advancing from left to right for each sentence word more strips compatible with the previously selected ones are selected within the collocation network constrains. The parser finds all possible combinations of strips composing parse trees representing possible output parses.

The corpus from which the vertical strips were extracted is 100,000 sentences large and was generated with Fawcett's natural language generation system and was tested on the same corpus leaving unclear how would the parser behave on a real corpus. In 98% of cases the parser returns a set of trees (between 0 and 56) that included the correct one with an average of 6.6 trees per parse.

Actually, using a larger corpus could potentially lead to a combinatorial explosion in the step that looks for vertical strips. It would decrease the accuracy of the parse because of the higher number of possible trees per parse.

2.1.5 Honnibal

Honnibal (2004; 2007) describes how Penn Treebank can be converted into a SFG Treebank. Before assigning to parse tree nodes synthetic features such as mood, tense, voice and negation he first transforms the parse trees into a form that facilitates the feature extraction.

The scope of SFG corpus was limited to a few Mood and Textual systems leaving aside Transitivity because of its inherently lexico-semantic nature. He briefly describes how he structurally deals with verb groups, complexes and ellipses as functional structures are much flatter than those exhibited in the original Treebank. Then he describes how are identified metafunctional features of unit class, mood function, clause status, mood type, polarity, tense, voice and textual functions.

The drawback of his approach is that the Python script performing the transformation does not derive any grammar but rather implements directly these transformations as functions falling into the same class of problems like Winograd's SHRDLU. By doing so the program is non-scalable for example in accommodation of larger grammars and knowledge bodies and unmaintainable on the long term as it becomes increasingly difficult to make changes.

Chapter 3

The systemic functional theory of grammar

Any description of language requires a theory that provides the frame, scope and the necessary concepts. Having a solid theory of grammar contributes to explaining what language is and how it works. It also frames how language is ought yo be analysed by either human or machines.

In his seminal paper Halliday (1961b) addresses the ardent need of the time for a general theory of language and partially answers the proposal for a universal theory of language. He sets out what was known at the time as Scale and Category Grammar. In such a model *units* are set up to account for for pieces of language which carry grammatical patterns. They are seen as arranged on a hierarchical *rank* scale of words, groups and clauses. These and other foundational concepts are covered in the first part of this Chapter.

There are two variants of Systemic Functional Grammars: the *Sydney Grammar* started in 1961 by Halliday (2002) and the *Cardiff Grammar* proposed by Fawcett (2008) which is a simplification and an extension of the Sydney Grammar. To understand the underlying common motives and how they are different we shall start looking at their theories of grammar. They also have quite different historical developments.

Sydney and Cardiff grammars have been formalised to the point where they could be computationally applied to natural language generation. They have been implemented in PENMAN (Mann 1983b; Penman Project 1989) and respectively COMMUNAL projects (Fawcett 1990). Both versions of SF grammars have been used predominantly for English and implementations of for other languages are also available. The major component of PENMAN is a computer model of Halliday's SF grammar described by Mann & Matthiessen (February 1983), Matthiessen & Bateman (1991), (Matthiessen

1995) and others. COMMUNAL is the computer implementation of Cardiff grammar described by Fawcett (1988a), Fawcett (1993) and others.

This chapter first sets out the basic organisational dimensions for each of the theories and then discusses comparatively Halliday's (Halliday 2002) and Fawcett's (Fawcett 2000) versions of SFL.

3.1 A word on wording

Before going into deeper discussion I first make terminological clarifications on the terms: grammar, grammatics, syntax, semantics and lexicogrammar. I start with a definitions adopted in "mainstream" generative linguistics and then present how the same terms are discussed in systemic functional linguistics.

Radford, a generative linguist, in the "Minimalist Introduction to Syntax" (1997), starts with a description of grammar as a field of study, which, in his words, is traditionally subdivided into two inter-related areas of study: syntax and morphology.

Definition 3.1.1 (Morphology (Radford))**.** Morphology is the study of how words are formed out of smaller units (traditionally called morphemes) (Radford 1997: 1).

Definition 3.1.2 (Syntax (Radford))**.** Syntax is the study of how words can be combined together to form phrases and sentences. (Radford 1997: 1)

Halliday, in the context of *rank* scale discussion (see Definition 3.2.1 and 3.2.2), refers to the traditional meaning of syntax as the *grammar above the word* and to morphology as *grammar below the word* (Halliday 2002: 51). Such a distinction, he states, has no theoretical status and is deemed as unnecessary distinction. Halliday adopts this position to motivate the architecture of grammar he was developing and is inherited from his precursor, Firth, as he puts it:

> . . . the distinction between morphology and syntax is no longer useful or convenient in descriptive linguistics. (Firth 1957: 14)

Radford adds that, traditionally, grammar is not only concerned with the principles governing formation of words, phrases and sentences but also with principles governing their interpretation. Therefore *structural aspects of meaning* are said to be also a part of grammar.

Definition 3.1.3 (Grammar (Radford))**.** [Grammar is] the study of the principles which govern the formation and interpretation of words, phrases and sentences. (Radford 1997: 1)

Interestingly enough, the Definition 3.1.3 makes no mention at all to the lexicon. This is because the formal grammars focus primarily on unit classes and how they are accommodated in various structures and so in formal linguistics the lexicon is often disconnected from the grammar. The systemic grammar, on the other hand, along with formal descriptions of grammatical categories and structures, includes the lexicon as part of grammar to form a *lexicogrammar*. At this point I have to mention that systemic functional grammar is not the only lexicalised one and there are others taking the same approach such as Lexical Functional Grammar (LFG), Head Phrase Structure Grammar (HPSG), Combinatory Categorial Grammar (CCG) and others.

Another important aspect to notice is that the grammar is defined as a field of study rather than a set of rules. De divergence in perspective on the subject led Halliday, since his early papers, to become conscious the difference between a study of a phenomenon with the phenomenon itself. By analogy to language as phenomenon and linguistics as the study of the phenomenon, discussed in (Halliday 1997), Halliday adopts the same wording for *grammar* as phenomenon and *grammatics* as the study of grammar; the same distinction holds for *syntax* and *syntactics*.

Definition 3.1.4 (Grammatics (Halliday))**.** Grammatics is a theory for explaining grammar (Halliday 2002: 369)

Moravcsik, another generative linguist, stresses the same distinction, in her "An introduction to syntax" (Moravcsik 2006), and presents two ways in which the word *syntax* is used in the literature: (a) in reference to a particular aspect of grammatical structure and (b) in reference to a sub-field of descriptive linguistics that describes this aspect of grammar. In her words:

> ...syntax describes the selection and order of words that make well-formed sentences and it does so in as general a manner as possible so as to bring out similarities among different sentences of the same language and different languages and render them explainable. ...syntax rules also need to account for the relationship between string of word meanings and the entire sentence meaning, on one hand, and relationship between strings of word forms and the entire sentential phonetic form, on the other hand. (Moravcsik 2006: 25)

In her definition of grammar she includes the lexicon and semantics which is a somewhat more explicit statement than Radford's *interpretation*. She is also getting, in Definition 3.1.5, somewhat closer to what grammar stands for in SFL - Definition 3.1.6.

Definition 3.1.5 (Grammar (Moravcsik))**.** ... maximally general analytic descriptions, provided by descriptive linguistics, [are] called grammars. A grammar has five components: phonology (or, depending on the medium, its correspondent e.g. morphology), lexicon, syntax and semantics(Moravcsik 2006: 24–25).

Definition 3.1.6 (Grammar (Halliday))**.** To Halliday, lexico-grammar, or for short,simply grammar is a part of language and it means the wording system - the "lexical-grammatical stratum of natural language as traditionally understood, comprising its syntax, vocabulary together with any morphology the language may display [...]" (Halliday 2002: 369).

The last point I want to mention is the approach to semantics. Formal grammars aim to account for the realisation variations, that is formation of words, phrases and sentences along with their arrangements and mention of semantics is often restricted to what may be termed the *formal aspect of meaning*.

By contrast, a systemic grammar is a functional grammar, which means (among other things) that it is semantically motivated, i.e. "natural". So the fundamental distinctions between formal and functional grammars is the semantic basis for explanations of structure.

Also, in SFL, the meaning is being approached from a semiotic perspective, placing the linguistic semantics in perspective with the linguistic expression and the real world situation. In this respect, Lemke (1993) offers a well formulated theoretical foundation that "human communities are eco-social systems that persist in time through ongoing exchange with their environment; and the same holds true for any of their sub-subsystems [...]" including language. The social practices constituting such systems are both material and semiotic, with a constant dynamic interplay between the two. (Halliday 2002: 387)

To Halliday, the term *semiotic* accounts for an orientation towards meaning rather than sign. In other words, the interaction is between *the practice of doing and the practice of meaning*. As the two sets of practices are strongly coupled, Lemke points out that there is a high degree of redundancy between the *material-semiotic interplay*. And it perfectly resonates with Firth's idea of *mutual expectancy* between the text and the situation. This idea of interplay is incorporated in SFL as *language stratification* and is graphically represented in Figure 3.1.

Having that said, the stratification axis is a useful dimension to relate the formal and the systemic functional grammars. This is also an instrument employed by Hjelmslev (Taverniers 2011).

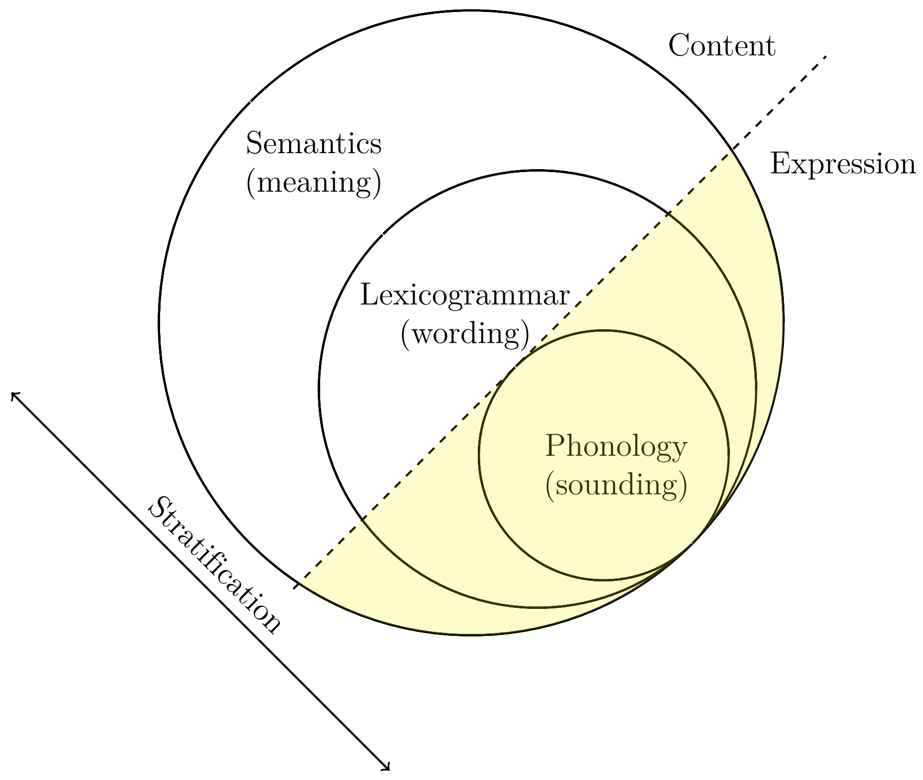


Fig. 3.1 The levels of abstraction along the realisation axis

The SFL model defines language as a resource organised into three strata: phonology (sounding), lexicogrammar (wording) and semantics (meaning). Each is defined according to its level of abstraction on the realisation axis. The realisation axis is divided into two planes: the expression and the content planes. Although debate about the precise division continues, for current purpose it is sufficient to see the first stratum (i.e. phonology/morphology) belongs to the *expression plane* and the last two (lexicogrammar and semantics) belong to the *content plane.* In this context, the formal grammar could be localised entirely within the expression plane, including the phonology/morphology, syntax, lexicon while formal semantics, stripped of any explanations in terms of the meaning potential, belongs in the content plane.

3.2 Sydney theory of grammar

I start introducing the terms of SFL theory with the Sydney grammar as this is in accordance with the historical development originating with Halliday (2002) defining the categories of the theory fo grammar. He proposes four fundamental categories:

unit, *structure*, *class* and *system*. Each of these categories is logically derivable from and related to the other ones in a way that they mutually define each other. These categories relate to each other on three scales of abstraction: *rank*, *exponence*, *delicacy*. Halliday also uses three scale types: *hierarchy*, *taxonomy* and *cline*.

Definition 3.2.1 (Hierarchy)**.** Hierarchy [is] a system of terms related along a single dimension which involves some sort of logical precedence. (Halliday 2002: 42).

Definition 3.2.2 (Taxonomy)**.** Taxonomy [is] a type of hierarchy with two characteristics:

1. the relation between terms and the immediately following and preceding one is constant
2. the degree is significant and is defined by the place in the order of a term relative to following and preceding terms. (Halliday 2002: 42)

Definition 3.2.3 (Cline)**.** Cline [is] a hierarchy that instead of being made of a number of discrete terms, is a continuum carrying potentially infinite gradations. (Halliday 2002: 42).

The concept of cline may not necessarily originate in SFL but it is used quite extensively in the domain literature. Next I define and introduce each category of *grammatics* and the related concepts that constitute the theoretical foundation for the Sydney Theory of grammar.

3.2.1 Unit

Language is a patterned activity of meaningful organization. The patterned organization of substance (*graphic* or *phonic*) along a linear progression is called *syntagmatic order* (or simply *order*).

Definition 3.2.4 (Unit)**.** The unit is a grammatical category that accounts for the stretches that carry grammatical patterns (Halliday 2002: 42). The units carry a fundamental *class* distinction and should be fully identifiable in description (Halliday 2002: 45).

Generalization 3.2.1 (Constituency principles)**.** The five principles of constituency in lexicogrammar are:

1. There is a scale or rank in the grammar of every language. That of English (typical of many) can be represented as: clause, group/phrase, word, morpheme.

2. Each unit consists of *one or more* units of rank next below.
3. Units of every rank may form complexes.
4. There is potential for rank shift, whereby a unit of one rank my be down-ranked to function in a structure of a unit of its own rank or of a rank below.
5. Under certain circumstances it is possible for one unit to be enclosed within another, not as a constituent but simply in such a way as to split the other into two discrete parts (Halliday & Matthiessen 2013: 9–10).

For example, the down-ranking (Point 4) can be observed in nominal groups that incorporate a relative clause functioning as qualifier. In example 8 *that I got for Christmas* is a relative clause specifying which books are being referred. The unit split (Point 5) can be encountered in the instances of Wh-interrogative clauses containing a preposition at the end which in fact belongs to the Wh-group. In example 9 the prepositional phrase *Who . . . about* is gapped and has an inverted order of constituents.

(8) I haven't read any books *that I got for Christmas*.

(9) *Who* are you talking *about*?

(10) I am talking about George.

The relation between units is that of consistency for which we say that a unit *consists of* other units. The scale on which the units are ranged is the *rank scale*. The rank scale is a levelling system of units supporting unit composition regulating how units are organised at different granularity levels from clause, to groups/phrases to words and the units of a higher rank scale consist of units of the rank next below. Table 3.1 presents a schematic representation of the rank scale and its derived complexes.

Rank scale ↓	**Complexing**
	Clause complex
Clause	
	Group(/phrase) complex
Group(/phrase)	
	Word complex
Word	
	(Morpheme complex)
(Morpheme)	

Table 3.1 Rank scale of the (English) lexicogrammatical constituency

Generalization 3.2.2 (Rank scale constraints)**.** The rank relations are constrained as follows:

1. in general elements of clauses are filled by groups,the elements of groups by words and the elements of words by morphemes.
2. downward *rankshift* is allowed i.e. the transfer of a given unit to a lower rank.
3. upward rankshift is not allowed.
4. only whole units can enter into higher units (Halliday 2002: 44).

The Generalization 3.2.2 taken as a whole means that a unit can include, in what it consists of, a unit of rank higher than or equal to itself but not a unit of rank more than one degree lower than itself; and not in any case a part of any unit (Halliday 2002: 42).

Following the rank scale constraints above the concept of embedding can be defined as follows.

Definition 3.2.5 (Embedding)**.** Embedding is the mechanism whereby a clause or phrase comes to function as a constituent within the structure of a group, which is itself a constituent of a clause. (Halliday & Matthiessen 2013: 242)

Halliday states that embedding is a phenomena that occurs only when a phrase/-group or clause function within the structure of a group which is itself a constituent of a clause (Halliday 1994: 242). The above definition of embedding permits the only for a clause and groups that function as elements of groups which means that a clause cannot fill the elements of another clause (Fawcett 2000: 237).

3.2.2 Structure

Definition 3.2.6 (Structure)**.** The structure (of a given unit) is the arrangement of *elements* that take places distinguished by the order relationship (Halliday 2002: 46).

Definition 3.2.7 (Element)**.** Element is defined by the place stated as absolute or relative position in sequence and with reference to the unit next below (Halliday 2002: 47).

We say that a unit is composed of elements located in places and that its internal structure is accounted for elements in terms of functions and places taken by the lower (constituting) units or lexical items. The graphic representation of the unit structure

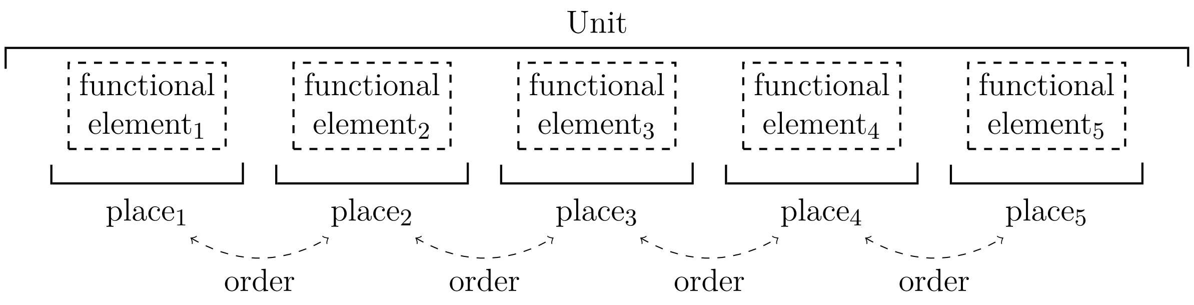


Fig. 3.2 The graphic representation of (unit) structure

is depicted in Figure 3.2. The unit structure is referred in linguistic terminology as *constituency* (whose principles are enumerated in Generalization 3.2.1). In the unit structure, the elements resemble an array of empty slots that are *filled* by other units or lexical items.

For example to account for the English clause structure four elements are needed: *subject*, *predicator*, *complement* and *adjunct*. They yield the distinct symbols, so that S, P, C, A is the inventory of elements. They then can be arranged in various orders falling in particular places, say SPC, SAPA, ASPCC etc. The places of elements are important with respect to the structure of the whole unit but also with respect to the relative ordering between these elements. For example S always fronts P, C is fronted by P unless the clause realises a Wh-interrogative whereas A is quite free and can occur anywhere in the unit structure.

3.2.3 Class

To one place in the structure corresponds one occurrence of the unit next below. This means that there will be a certain grouping of members identified by the functional element they take in the structure. Patterning such groupings leads to emergence of *classes* of units.

In the clause structure example, elements in the unit are occupied by units of lower rank and of a particular class. The relation between the element and the class is mutually determined. In each of these elements is placed a lower rank unit and of an expected class. For instance in the S position can be placed a *noun*, *nominal group*, *pronoun* or another *clause* (that will be a down-ranking situation defined above).

Definition 3.2.8 (Class)**.** The class is that grouping of members of a given unit which is defined by the operation (i.e. functional element) in the structure of the unit next above (Halliday 2002: 49).

Halliday defines class (Definition 3.2.8) as likeness of the same rank *phenomena* to occur together in the structure. He adopts a top-down approach stating that the class of a unit is determined by the *function* (Definition 3.2.13) it plays in the unit above and not by its internal structure of elements. In SFG the structure of each class is well accounted in terms of syntactic variation recognizing six unit classes: *clause*, *nominal*, *verbal*, *adverbial* and *conjunction* groups and *prepositional phrase*. The Sydney unit structure model is briefly summarised in the Appendix **??**.

Halliday identifies the concept of *grammatical metaphor* defined in 3.2.9 and it plays an important role in the SFG as a whole for accounting for the versatility of natural language. It typically found in adult language where one type of process are expressed in the grammar of another.

Definition 3.2.9 (Grammatical metaphor)**.** Grammatical metaphor involves the substitution of one grammatical class or structure for another, often resulting in a more compressed expression.

(11) The fifth day saw them at the summit.

(12) On the fifth day they arrived at the summit.

(13) Guarantee limited to refund of purchase price of goods.

(14) we guarantee only to refund the price for which the goods were purchased.

For example 11 and 13 are instances of grammatical metaphor whereas the 12 and 14 are their non metaphorical counterparts. In Examples 11 and 12 the temporal circumstance of an action expressed through a prepositional phrase becomes the nominal actor of a perception process. Children speech is largely free of such kind of metaphors, in fact this is the main distinctions between the two.

3.2.4 System

As described above, structure is a syntagmatic ordering in language capturing regularities and patterns which can be paraphrased as *what goes together with what*. However in SFG most of the descriptive work is carried not syntagmatically but paradigmatically via *system networks* (Definition 3.2.10) describing *what could go instead of what* (Halliday & Matthiessen 2013: 22). Note that the paradigmatic-syntagmatic axes date back to the works of Saussure (1959 [1915]). Both are important for completing a linguistic description. Here lies one of the main differences between SFL and other approaches which is is taking the paradigmatic path whereas many others take the syntagmatic path to language representing it as an inventory of structures. The structure of course is a part of language description but it is only a syntagmatic manifestation of the systemic choices and one needs to account for both (Halliday & Matthiessen 2013: 23).

Definition 3.2.10 (System)**.** A system is a set of mutually exclusive set of terms referring to meaning potentials in language and are mutually defining. The system is considered self-contained, closed and complete with the following characteristics:

1. the number of terms is finite,
2. each term is exclusive of all others,
3. if a new term is added to the system it changes the meaning of all the other terms Halliday (2002: 41).

The concept of a system as presented in Definition 3.2.10 has its roots in the works of Saussure (1959 [1915]) and Hjelmslev (1953) and Halliday only cements it in SFL architecture of grammar.

Going back to the notion of class previously defined as a grouping of items identified by functions in the structure, it needs stressed here that class is not a list of formal items but an abstraction from them. By increase in *delicacy* a class is broken into secondary classes.

Definition 3.2.11 (Delicacy)**.** Delicacy is the scale of differentiation or depth of detail whose limit at one end is the primary degree of categories of structure and class and on the other end, theoretically, is the point beyond which no further grammatical relations obtain. Halliday (2002: 58)

We say that a category is refined into more subtle distinctions of subcategories which form a system as define above. Subsequently those distinctions fo subcategories can be further refined in other systems. This relationship between these two systems is one of delicacy where the second one is more delicate than the first one and together they form a *system network*.

The graphical notations introduced by Halliday & Matthiessen (2013) are useful in reading and writing system networks in this thesis. Below is a system network with a simple *entry condition* (Figure 3.3), a *system network grouping* that share the same entry condition (Figure 3.4), a system network with a *disjunctive* and *conjunctive* entry conditions (Figure 3.5 and 3.6).

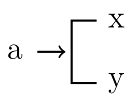


Fig. 3.3 A system with a single entry condition: if *a* then either *x* or *y*

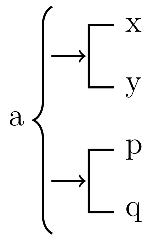


Fig. 3.4 Two systems grouped under the same entry condition: if *a* then both either *x* or *y* and, independently, either *p* or *q*

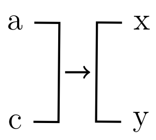


Fig. 3.5 A system network with a disjunctive entry condition: if either *a* or *c* (or both), then either *x* or *y*

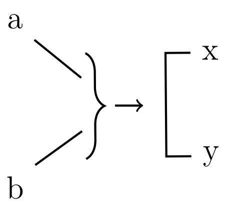


Fig. 3.6 A system with a conjunctive entry condition: if both *a* and *b* then, either *x* or *y*

It is worth noting that when a piece of language is analysed, it can be approached at various levels of delicacy. We say that delicacy is variable in description, and one may choose to provide coarse grained analysis without going beyond primary grammatical categories or it can dive into fine grained categorial distinctions, still being comprehensive with regards to the rank, *exponence* and grammatical categories.

Definition 3.2.12 (Exponence)**.** Exponence is the scale which relates the categories of theory which are with high degree of abstraction to formal items on its low end. Each exponent can be linked directly to the formal item or by taking successive steps on the exponence scale and changing rank where necessary. Halliday (2002: 57)

And in relation to the previous section, the class stand in the relation of exponence to an element of primary structure of the unit next above. This breakdown gives a system of classes that constitute choices implied by the nature of the class (Halliday 2002: 41).

3.2.5 Functions and metafunction

Above, when talking about structure, I described a unit as being composed of elements accounted in terms of *functions* and places taken by the lower (constituting) units or lexical items.

Definition 3.2.13 (Function)**.** The functional categories or functions provide an interpretation of grammatical structure in terms of the overall meaning potential of the language. (Halliday & Matthiessen 2013: 76).

Most constituents of clause structure, however, have more than one function, which is called a *conflation of elements*. For example in the sentence “Bill gave Dolly a rose”, “Bill” is the Actor doing the act of giving but also the Subject of the sentence. So we say that Actor and Subject functions are conflated in the constituent “Bill”. This is the concept of *metafunction* or *strand of meaning* comes into the picture. The Subject function is said to belong to the *interpersonal metafunction* while the Actor function belongs to the *experiential metafunction*.

Halliday identifies three fundamental dimensions of structure in the clause, each meaning: *experiential*, *interpersonal* and *textual*. He refers to them as *metafunctions* and they account for the functions that language units take on in communication. Table 3.2 presents the metafunctions and their reflexes in grammar as proposed by Halliday & Matthiessen (2013: 85).

Metafunction	**Definition(kind of meaning)**	**Corresponding status in clause**	**Favored type of structure**
experiential	construing a model of experience	clause as representation	segmental (based on constituency)
interpresonal	enacting social relationship	clause as exchange	prosodic
textual	creating relevance to context	clause as message	culminative
logical	constructing logical relations	complexes (taxis & logico-semantic type)	iterative

Table 3.2 Metafunctions and their reflexes in the grammar

Across the rank scale, with respect to structure and metafunctions, Halliday formulates the general principle of *exhaustiveness* (Generalization 3.2.3) saying that clause constituents have at least one and may have multiple functions in different strands of meaning; however this does not mean that it must have a function in each of them. For example interpersonal Adjuncts such as “perhaps” ot textual Adjuncts such as “however” play no role in the clause as representation.

Generalization 3.2.3 (Exhaustiveness principle). Everything in the wording has some function at every rank but not everything has a function in every dimension of structure (Halliday 2002; Halliday & Matthiessen 2013).

This principle implicitly relates to the property of language meaning that there is nothing meaningless and thus every piece of language must be explained and accounted for in the lexicogrammar. Also this principle implies that each metafunction has its own structure or that text is analysed through a multi structural approach.

At the very top of the rank scale, clauses form complex structures. Halliday employs systematically the concepts of *taxis* and *logico-semantic relations* to account for inter-clausal relations.

Definition 3.2.14 (Taxis). *Taxis* represents the degree of interdependency between units systematically arranged in a linear sequence where *parataxis* means equal and *hypotaxis* means unequal status of units forming a *nexus* or a *unit complex* together.

The concept of taxis which is very useful at describing unit relations not only at the group and clause ranks but all the way down to smallest linguistic unit such as morphemes and phonemes. I will also refer to it when describing the Cardiff theory of grammar and also briefly in the discussion of dependency relations in Section 4.6.

The elements of logical paratactic structure are notated left to right with numbers (1 2) while those of hypotactic structure with Greek letters ($\beta \quad \alpha$) right to left. The tactic relations can be of two types: that of expansion which relates phenomena of the same order of experience and that of projection which relates phenomena of one order of experience (usually saying or thinking) to an order of experience higher (what is said or thought). Projection can be of two types: *idea* (' single quote) and *locution* (" double quotes).

Expansion is further divided into three: *elaborating* (= equals), *extending* (+ is added to) and *enhancing* (× is multiplied by). Elaboration is a way to re-state the same thing, exemplify, comment or specify in detail. Extending is the way to add new element give an exception or offer an alternative. And finally enhancing is the way to qualify something with some circumstantial feature of time, place, cause, intensity or condition.

3.2.6 Lexis and lexicogrammar

In SFL the terms *word* and *lexical item* are not really synonymous. They are related but they refer to different things. The term *word* is reserved (in early Halliday) for the grammatical unit of the lowest rank whose *exponents* are lexical items.

Definition 3.2.15 (Lexical Item)**.** In English, a lexical item may be a *morpheme*, *word* (in traditional sense) or *group (of words)* and it is assigned to no rank (Halliday 2002: 60).

Examples of lexical items are the following: "'s" (the possessive morpheme), "house", "walk", "on" (words in traditional sense) and "in front of", "according to", "ask around", "add up to", "break down" (multi word prepositions and phrasal verbs).

If some theories treat grammar and lexis as discrete phenomena, Halliday brings them together as opposite poles of the same cline. He refers to this merge as *lexicogrammar* where they are paradigmatically related through delicacy relation. Hasan (2014), explores the feasibility of what would it mean to turn the "whole linguistic form into grammar". This then implies a assumption that lexis is not form and that its relation to semantics is unique which in turn is challenging the problems of polysemy.

3.3 The Cardiff theory of grammar

As presented in the introduction and explained by Bateman (2008), the accounts along the syntagmatic axis had gone missing in the Sydney grammar leaving unresolved how to best represent the structure of language at the level of form. This section present the theory of systemic functional grammar as conceived by Robin Fawcett at the University of Cardiff. His book "A theory of syntax for Systemic Functional Linguistics" (Fawcett 2000) presented a proposal for a *unified syntactic model* for SFL that contrasts several aspects of Hallidayan grammar but share the same set of fundamental assumptions about the language; it is an extension and a simplification in a way.

Fawcett questions the status of multiple structures in the theory and whether they can finally be integrated into a simpler sole representation. A big difference to Hallidayan theory is renouncing the concept of rank scale which has an impact on the whole theory. Another is the bottom-up approach to unit definition as opposed to top-down one advocated by Halliday. These two and a few other differences have important implications for the overall theory of grammar and consequently for the grammar itself. As a consequence, to accommodate the lack of rank-scale, Fawcett adapts the definitions of the fundamental concepts and changes his choice of words (for example "class" and "unit" turn into "class of unit" treated as one concept rather than two distinct ones).

Fawcett (2000) proposes three fundamental categories in the theory of grammar: *class of unit*, *element of structure* and *item*. Constituency is a relation accounting for the prominent compositional dimension of language. However a unit does not function directly as a constituent of another unit but via a specialised relation which Fawcett breaks down into three sub-relations: *componence*, *filling* and *exponence*. Informally it is said that a unit is composed of elements which are either filled by another unit or expounded by an item. He also proposes three secondary relations of *coordination*, *embedding* and *reiteration* to account for a more complete range of syntactic phenomena.

3.3.1 Class of units

Fawcett's theory of language assumes a model with two levels of *meaning* and *form* corresponding to *semantic units* and *syntactic units* which are mutually determined (which is the case for any sign in a Saussurean approach to language).

Definition 3.3.1 (Class of Unit)**.** The class of unit [...] expresses a specific array of meanings that are associated with each one of the major classes of entity in semantics

[...and] are to be identified by the elements of their internal structure (Fawcett 2000: 195).

For English Fawcett proposes four main kinds of semantic entities: situations, things, qualities (of both situations and things) and quantities. Each of these semantic units corresponds to five major classes of syntactic units: *clause*, *nominal group*, *prepositional group*, *quality group* and *quantity group*. In addition he recognises two more minor classes i.e. the *genitive cluster* and the *proper name cluster* (Fawcett 2000: 193–194).

Fawcett's classification is based on the idea that the syntactic and semantic units are mutually determined and supported by grammatical patterns. However those patterns lie beyond the syntactic variations of the grammar and so blend into lexical semantics.

In Sydney theory the class is determined by the function it plays in the unit above. By contrast, in Cardiff theory, the class of unit is determined based on its internal structure i.e. by its *elements of structure* (and not by the function it plays in the parent unit).

3.3.2 Element of structure

The terms *element* and *structure* have roughly the same meaning as defined in Sydney theory of grammar (defined in Section 3.2) but with two additional stipulations presented below.

Definition 3.3.2 (Element of Structure)**.** Elements of structure are immediate components of classes of units and are defined in terms of their *function* in expressing meaning and not in terms of their absolute or relative position in the unit. (Fawcett 2000: 213–214).

The definition above leads as a consequences to two important properties of elements formulated as follows.

Generalization 3.3.1 (Element functional uniqueness)**.** Every element in a given class of unit serves a function in that unit different from the function of the sibling elements (Fawcett 2000: 214).

Even if for example, different types of *modifiers* in English nominal group seem to have very slight differences in functions, they are still there.

Generalization 3.3.2 (Element descriptive uniqueness)**.** Every element in every class of unit will be different from every element in every other class of unit (Fawcett 2000: 214).

Thus the terms of modifier and head shall not be used for more than one class of unit. In English grammar the head and modifier are used for nominal group only. And in other groups the elements of structure may seem similar to modifier and head, they still receive different names such as *apex* and *temperer* in the quality group.

The elements (of structure) are functional slots which define the internal structure of a unit but still they are *located* in *places*. One more category that intervenes between element and unit is the concept of *place* which become essential for the generative versions of grammar.

There are two ways to approach place definition. The first is to treat places as positions of elements relative to each other (usually previous). This leads to the need for an *anchor* or a *pivotal element* which may not always be present/realised.

The second is to treat places as a linear sequence of locations at which elements may be located, identified by numbers "place 1", "place 2" etc. This place assignment approach is absolute within the unit structure and makes elements independent of each other. This approach has been used in COMMUNAL (Fawcett 1990) and PENMAN (Mann 1983b) projects.

3.3.3 Item

Definition 3.3.3 (Item)**.** The item is a lexical manifestation of meaning outside syntax corresponding to both words (in the traditional sense), morphemes and either intonation or punctuation (depending whether the text is spoken or written). (Fawcett 2000: 226–232).

Items correspond to the leaves of syntactic trees and constitute the raw *phonetic* or *graphic* manifestation of language. The collection of items of a language is generally referred to as *lexis*.

Since items and units are of different natures, the relationship between an element and a (lexical) item must be different from that to a unit. We say that items *expound* elements and not that they *fill* elements as units do.

Definition 3.3.4 (Exponence (restricted))**.** Exponence is the relation by which an element of structure is realised by a (lexical) item (Fawcett 2000: 254).

If in Sydney model exponence (Definition 3.2.12) is a relation that links abstract grammatical categorise to the data In Cardiff model it has a restricted meaning referring to relation between items and elements only.

3.3.4 Componence and obscured dependency

Definition 3.3.5 (Componence)**.** Componence is the part-whole relationship between a unit and the elements it is composed of (Fawcett 2000: 244).

Note that componence is not a relationship between a unit and its places; the latter, as discussed in Section 3.3.2, simply locationally relate elements of a unit to each other.

Componence intuitively implies a part-whole constituency relationship between the unit and its elements. But this is not the only view. Another perspective is the concept of *dependency* (which I will address in Chapter 4) or strictly speaking the *sister* or *sibling dependency* (not parent-daughter). It is suitable for describing relations between elements of structure within a unit.

(15) the man with a stick

For example the componence of nominal group in Example 15 is (*dd h q*) which are symbols for (*determiner head qualifier*). The same can be expressed in terms of sibling dependency relations depicted in Figure 3.7. The relations from *stick* to *with a* are not depicted because they belong in description of prepositional group *with a stick*.

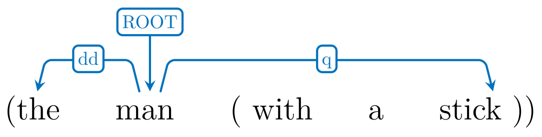


Fig. 3.7 Sibling dependency representation for “the man with a stick”

In both SFL theories, the sister dependency relations is considered a by-product or second order concept that can be deduced from the constituency structure thus unnecessary in the grammar model. I will come back to this point because current work relies on this dual view on elements of structure and relation to the whole unit.

The (supposed) dependency relation between a modifier and the head, in the framework of SFG is, not a direct one. Simply assume that what modifier modifies is the head. Here, however, the general function of the modifiers is to contribute to the meaning of the whole unit which is anchored by the head.

In the nominal group from Example 15, the *determiner* and *qualifier* are modifiers that contributes to the description of the referent stated by the *head*. So the head realises one type of meaning that relates the *referent* while modifier realises another one. Both of them describe the referent via different kinds of meaning, therefore, according to Fawcett, they are related indirectly to each other because the modifier does not modify the head but the referent denoted by the head From this point of view, whether the element is dependent on a sibling element such as the head or on the parent unit is beside the point because in syntax we can observe its realization in system networks (Fawcett 2000: 216–217). Next I move towards last concept in Cardiff model, *filling*, which is a relation between the elements of structure and the units below.

3.3.5 Filling and the role of probabilities

Definition 3.3.6 (Filling)**.** Filling is the probabilistic relationship between a element and the unit lower in the tree that operates at that element (Fawcett 2000: 238, 251).

Fawcett replaces the rank scale with the concept of *filling probabilities*. The probabilistic predictions are made in terms of filling relationship between a unit and an element of structure in a higher unit in the tree rather than being a relationship between units of different ranks. This moves focus away from the fact that a unit is for example a group, and towards what group class it is.

In this line of thought, some elements of a clause are frequently filled by groups, but some other elements are rather *expounded* by items. The frequency varies greatly and is an important factor for predicting or recognizing either the unit class or the element type in the filling relationship.

Filling may add a *single unit* to the element of structure or it can introduce *multiple coordinated units*. Coordination (Example 16) is usually marked by an overt *Linker* such as *and*, *or*, *but*, etc. and sometimes is enforced by another linker that introduces the first unit such as *both*.

Definition 3.3.7 (Coordination)**.** Coordination is the relation between units that fill the same element of structure (Fawcett 2000: 263).

(16) she is (friendly, nice and polite)

(17) she is (very very) nice!

Coordination is through by Fawcett as being not between syntactic units but between mental referents. It always introduces more than one unit which are syntactically and semantically in similar (somehow) resulting in a *syntactic parallelism* which often leads to *ellipsis*.

Definition 3.3.8 (Reiteration)**.** Reiteration is the relation between successive occurrences of the same item expounding the same element of structure (Fawcett 2000: 271).

Reiteration (Example 17) often is used to create the effect of emphasis. Like coordination, reiteration is a relation between entities that fill the same element of the unit structure which is problematic in my opinion and I further discuss it in Section 3.4.6.

Filling also makes possible the embedding relation which Fawcett treats as a general principle in contrast to more specific Definition 3.2.5 from Sydney model.

Definition 3.3.9 (Embedding (generic))**.** Embedding is the relation that occurs when a unit fills (directly or indirectly) an element of the same class of units; that is when a unit of the same class occurs (immediately) above it in the tree structure (Fawcett 2000: 264).

(18) (To become an opera singer) takes years of training.

(19) The girl (whom he is talking to) is an opera singer.

In Example 18 we can see an occurrence of direct embedding where an infinite clause acts as the subject of another clause. In Example 19 the embedding is indirect as the relative clause is part of the nominal group which functions as the subject in the parent clause. In both cases we say that a lower clause is embedded (directly or indirectly) in higher or parent clause. I will further discuss this in the context of rank-scale concept in Section 3.4.1.

A situation converse to reiteration and coordination where a element is filled by more than one unit, is known as *conflation* where a unit can take more than one function within another.

Definition 3.3.10 (Conflation)**.** Conflation is the relationship between two elements that are filled by the same unit having the meaning of "immediately after and fused with" and function as one element (Fawcett 2000: 249–250).

Conflation is useful in expressing multi-faceted nature of language when for example syntactic and semantic elements/functions are realised by the same unit. For example the Subject "the girl whom he is talking to" is also a *Carrier* while the Complement "an opera singer" is also an *Attribute*. Also conflation relations frequently occur between syntactic elements as well such as for example the *Main Verb* and *Operator* or *Operator* and *Auxiliary Verb*.

Cardiff Grammar in case of both coordination and embedding relations deals without *hypotaxis* and *parataxis* relations described in Sydney Grammar.

Note also that filling and componence are two complementary relations that occur in the syntactic tree down to the level when the analysis moves out of abstract syntactic categories to more concrete category of items via the relationship of exponence.

3.4 Critical discussion on both theories: consequences and decisions for parsing

The two sections above cover the definitions and fundamental concepts from each of the two systemic functional theories of grammar. The work in this thesis uses a mix of concepts from both theories and this section discusses in detail what is being adopted and why attempting a rather pragmatic reconciliation for the purposes of achieving a parsing system than a theoretical debate. Next I draw parallels and highlight correspondences between the Sydney and Cardiff theories of grammar and where alter and present my position on the matter.

3.4.1 Relaxing the rank scale

The *rank scale* proposed by Halliday (2002) became over time a highly controversial concept in linguistics. The discussion whether it is suitable for grammatical description or not still continues. The historic development of this debate is documented in some detail (Fawcett 2000: 309–338).

In this section I present a few cases that highlighting when the rank scale as defined by Sydney is too rigid. As a consequence for the purpose of thesis I drop the *rank scale constraints* as enunciated in Generalization 3.2.2. Also the *rankshift* operation, exceptionally employed to accommodate special cases, is overridden by a broad definition of *embedding* operation (Definition 3.3.9) treated as naturally occurring phenomena in language at all ranks. I do not entirely dismiss the concept of rank scale as proposed in Cardiff school as I still find it useful in classification of units.

(20) some very small wooden ones

Consider the nominal group 20. Here the modifying element, the Epithet "very small", is not a single word but a group (Halliday & Matthiessen 2013: 390–396). As the rank scale constraints mentioned above state that the group elements need to be filled by words. To account for this phenomena, Halliday, introduces a *substructure* of modifiers and heads leading to a logical structure analysis as the one in Table 3.3. In such a structure the modifier is further broken down into a Sub-Head and Sub-Modifiers.

some	*very*	*small*	*wooden*	*ones*
Modifier				Head
δ	γ		β	α
	Sub-Modifier	Sub-Head		
	$\gamma\beta$	$\gamma\alpha$		

Table 3.3 Sydney logical structure analysis of Example 20

The corresponding experiential structure analysis is provided in the Table 3.4 Halliday & Matthiessen (2013: 391). Accordingly, the Epithet "very small" is composed of a quality adjective "small" and an enhancer modifier "very".

some	*very*	*small*	*wooden*	ones
Deictic	Epithet		Classifier	Thing
	Sub-Modifier	Sub-Head		

Table 3.4 Sydney experiential analysis of Example 20

As you can see the elements are further broken down into sub-elements composing in a way a structure of their own. This is possible because of the poly-structural and multi functional approach to text analysis which in this case leads to a complex structure of a nominal group. This kind of intricate cases can be simplified through the permission that elements of a group to be filled by other groups or expounded by words. This way, instead of having a sub-modifier construction simply consider that the Epithet is filled by an adjectival or nominal group which in turn has its own structure. Please note that I mention adjectival or nominal group because in Sydney grammar the adjectival group is considered as a nominal group with covert Thing where the Epithet acts as Head; this however is a discussion beyond the point I make here.

The same example analysed with Cardiff grammar would look like in Table 3.5. It follows precisely the above suggestion of filling the Epithet with another unit, in this case a Quality Group which in turn has its own internal structure.

some	*very*	*small*	*wooden*	*ones*
Quantifying Determiner	Modifier		Modifier	Head
	Quality Group			
	Degree Tamperer	Apex		

Table 3.5 Cardiff analysis of Example 20

(21) Indians had originally planned to present the document to President Fernando Henrique Cardoso.

Indians	*had*	*originally*	*planned to present*	*the document*	*to President Fernando Henrique Cardoso*
Mood		Residue			
Subject	Finite	Adjunct	Predicator	Complement	Adjunct
nominal group		adverbial group		nominal group	prepositional phrase
	verbal group				

Table 3.6 Sydney grammar Mood analysis of Example 21

Another case that deems the rank scale constraints too strict for the present work is in the case of Finite element in the Clause. Consider example 21 where the Finite and Predicator elements are filled by a single unit which is the verbal group which is against the constituency principles which restricts the composition relation to engage only with whole units.

Alternatively, if the unit filling the Finite element is considered separate from the verbal group filling the Predicator then it is always a single word, a modal verb, and never a verbal group. This again is a breach in the rank scale constraints which postulates that a unit may be composed of units of equal rank or a rank higher and cannot be composed of units that are more than one rank lower thus it is not permitted to have clause elements expounded by words directly.

The two cases above I use to demonstrate how the ranks scale construct as defined by Sydney grammar is too rigid and thus unsuitable for the current work. I drop the constituency constraints hence allowing the flexibility for elements to be filled by other units or, in other words, allow unit *embedding*. This approach removes the need of sub-structures in the unit elements reducing thus the structural complexity as seen in Table 3.5.

The weakening of constituency constraints makes embedding a normal (broadly defined in Definition 3.3.9) rather than an exceptional phenomena (strictly defined in Definition 3.2.5).

An approach to describe units outside the rank-scale was suggested by Fawcett (2000) and Butler (1985). Fawcett proposes replacing it with the filling probabilities to guide the unit composition simply mapping elements to a set of legal unit classes that may fill it. Units are carriers of a grammatical pattern, they can be described in terms of their internal structure instead of their potential for operation in the unit above. Nonetheless I do not abandon the rank scale completely and I use it as the top level classifier of grammatical units (see Figure 3.10) falling in line with more traditional syntactic classes.

3.4.2 Approach to structure formation

The *unit* and *structure* are two out of the four fundamental categories in the systemic theories of grammar. The Sydney and Cardiff theories vary in their perspectives on *unit* and *structure* influencing how units are defined and identified.

For Halliday, the *structure* (Definition 3.2.6) characterises each unit as a carrier of a pattern of a particular order of *elements*. The order is not necessarily a linear realisation sequence but a theoretical relation of relative or absolute placement. This perspective has been demonstrated to be useful in generation where unit placement emerges out of the realisation process.

The Cardiff School takes a bottom up approach and defines class in terms of its internal structure describing a relative or absolute order of elements. This sort of syntagmatic account is precisely what is deemed useful in parsing and is the one adopted in this thesis. In this work, as motivated in the Introduction, generation of the constituency structure is derived from the Stanford dependency parse trees. As it consists of words and relations between them the intuitive approach to form groups, clauses and complexes is by working them out bottom up. The method is to let the unit class emerge from recognition of constituent word classes and dependency relations between or sequence of already formed lower units. The exact mechanism how it is done I explain in Chapter 7. What is important to note here is the bottom up approach which is in line with Cardiff way of defining unit classes in contrast to top down approach of Sydney school.

3.4.3 Relation typology in the system networks

As system is expanded in delicacy to forms a systemic network of choices, choice of a feature in one system becomes the *entry condition* for choices in more delicate systems below. Halliday states that the relation on the systemic cline of delicacy is essentially of *sub-categorisation* (see Definition 3.2.11). In this subsection argue for occurrence of multiple kinds of inter-systemic relations. I also call them *activations relations* because in the traversal process from less to more delicate systems, when choices are made in the former then choice making is enabled or *activated* in the latter.

Next I present a distinction between two activation relations: the *sub-categorisation* and *choice enabling*. that are of interest in the present thesis but this is by no means exhaustive distinction and more work is needed here.

Lets take as example the polarity system represented in figure 3.8. It contains two choices either positive or negative. An increase in delicacy can be seen as a taxonomic "is a" relationship between features of higher systems and lower systems like in the case of POLARITY TYPE and NEGATIVE TYPE in figure 3.8 and in fact for the rest of the network. As a side note, the delicacy in a system network is akin to sub-classification relation, which was originally the intended one and the predominant one. In practice, however, a few kinds of abstraction relations can be encountered (e.g abstraction as information reduction, as approximation, as idealisation etc.) extensively treated by Saitta & Zucker (2013). This discussion however beyond the scope of the current work.

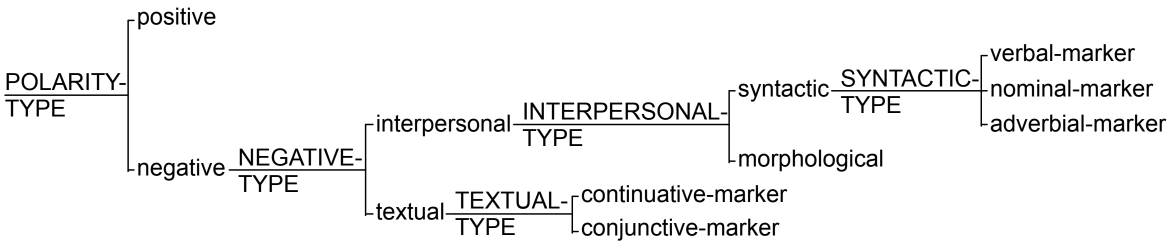


Fig. 3.8 System network of POLARITY

But the activation relation among systems in the cline of delicacy is not always taxonomic. Another relation is "enables selection of" without any sub-categorisation implied. As an example see the FINITENESS system in Figure 3.9 where in case that the finite option is selected then what this choice enables is not subtypes of finite but merely other systems that become available i.e. DEIXIS and INDICATIVE TYPE. The latter is there because selection of finite implies also selection of indicative feature in a sibling of FINITNES system, MOOD-TYPE comprised of options indicative and imperative.

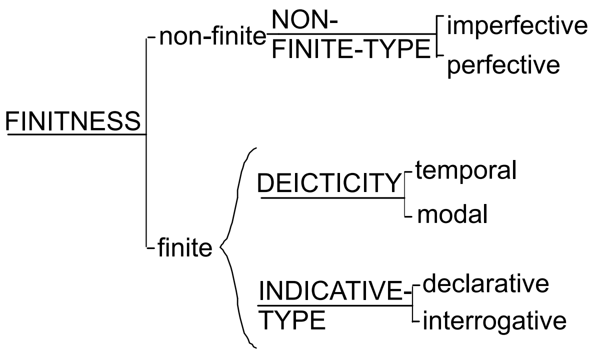


Fig. 3.9 A fraction of the finiteness system where increase of delicacy is not a "is a" relation

The distinction in the systemic relations is incorporated into the technical data structure definitions and traversal algorithms proposed in the Chapter 6.

3.4.4 Unit classes

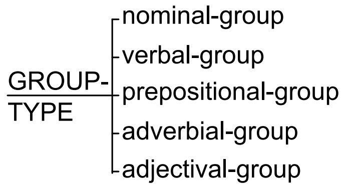
In SFL at large there is the consensus that linguistic forms and meanings are intertwined and mutually determined just like for any sign in a Saussurean approach to language. Both Halliday (quote below) and Fawcett (Definition 3.3.1) adopt this position.

> . . . something that is distinctly non-arbitrary [in language] is the way different kinds of meaning in language are expressed by different kinds of grammatical structure, as appears when linguistic structure is interpreted in functional terms (Halliday 2003a).

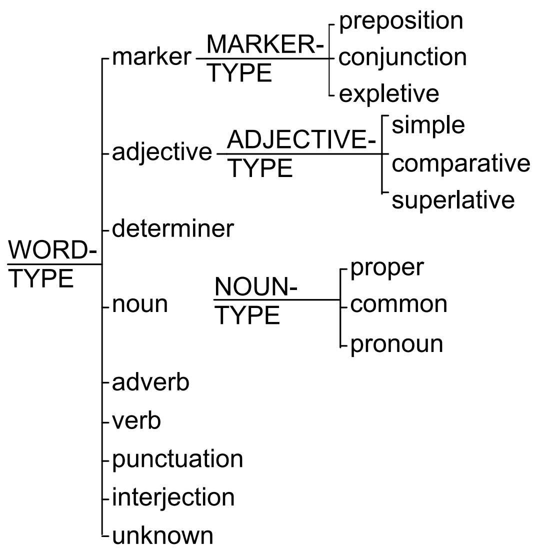
When it comes to establishing the lexicogrammatical classes the two schools diverge. Halliday adopts the traditional grammar *word classes* or *parts of speech*: noun, verb, adjective etc. He then derives a set of groups (e.g. nominal group, verbal group, adverbial group etc.) that share properties of the word classes. In fact the class, in Halliday's words, "indicates the in a general way its potential range of grammatical functions" (Halliday & Matthiessen 2013: 76). For example the nominal group is a formation that functions as a noun may do and expresses same kind of meaning.

Following the idea that major semantic classes of entities (situations, things, qualities and quantities) correspond to the major syntactic units, Fawcett decided to mirror them into the lexicogrammar. This lead to a semantically based classification of syntactic units: clause, nominal group, prepositional group, quality group and quantity group (Fawcett 2000: 193–194) along with a set of minor classes such as genitive and proper name clusters. This is, in a way, a tight coupling of the grammatical units with an ontology which may be subject to change in the future. The converse may also be stated that the traditional part of speech are disconnected from the semantics in the sense that there is no one to one correspondence (as Fawcett attempts) but rather complex set of mappings. Establishing the exact interface of syntax and semantics is a hot ongoing theoretical exploration across the entire linguistic discipline a difficult task in practice. This discussion however is beyond the scope here.

In the current work I side with the Sydney classification of syntactic units that is close in line with traditional syntactic classifications (Quirk et al. 1985). I adopt the clause as a unit plus the four group classes of the Sidney grammar depicted in Figure 3.10.



(a) The group classes



(b) The word classes

Fig. 3.10 The group and word classes

The word classes or part of speech tags that I adopt here are the ones employed to annotate Penn Treebank corpora called the Penn tag set (Marcus et al. 1993) which, like Sidney unit classes, are also in line with the traditional grammar. This tag set has become a widely accepted standard in mainstream computational linguistics and there are multiple implementation of the part of speech taggers. The Stanford Parser which plays an important role in the software implementation of this thesis and is described in the Chapter 4, employs precisely the Penn tag set.

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

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

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

Third, Penn POS set allows multiple tags per word, meaning that the annotators may be unsure of which one to choose in certain cases. There are 36 main POS and 12 other tags in the Penn tag set. A detailed description of the schema, the design principles and annotation guidelines are described in (Santorini 1990). Figure 3.10 depicts a classification summarising the Penn tag set.

3.4.5 Syntactic and semantic heads

In SFG the heads may be motivated by semantic or syntactic criteria (simply called here semantic or syntactic heads). In most cases they coincide but there are exceptions when they differ or even diverge. This topic is especially important in the discussions of the **nominal group** structure (continued in Section 3.5.3) on which Halliday & Matthiessen (2013) offers a thorough examination and Fawcett (2000) provides a more generic perspective.

In this discussion I show few examples when the syntactic and semantic heads diverge and argue my position on the group formation on two points. First, the class of the Head (in Sydney school) or pivotal element (in Cardiff school) is not always raised to establish the group class but the whole underlying structure determines the group class. Second, that syntactically motivated heads are easy to establish because they are based solely on the formal grounds whereas the semantic heads require an evaluation at the level of entire group, once one is established, employing additional lexical semantic resources. This can be a two step process but in the current implementation only the group structure on syntactic grounds is provided.

As mentioned before in Section 3.2.5, Sydney grammar fulfils the exhaustiveness principle (Generalization 3.2.3) through multiple parallel structures while Cardiff grammar through a single syntactic structure resembling a mixture of the former.

Let's briefly return to the Example 20 analysed with Sydney grammar in Table 3.3 and 3.4 that reflect the nominal group logical and experiential structures Halliday & Matthiessen (2013: 391). When the Head (called here the *syntactic head* of the nominal group) coincides with the Thing (called here the *semantic head*) we say that they are conflated (Definition 3.3.10) and examples as this one may lead to the assumption that the Head, which is motivated by syntactic criteria, is also always the Thing which is motivated by the semantic criteria, but this is not so.

The logical structure is a Head-Modifier structure and "represents the generalised logical-semantic relations that are encoded in the natural language" (Halliday & Matthiessen 2013: 388). The experiential structure of the nominal group as a whole has the function of specifying the class of things, through the Thing element, and some category of membership in this class, through the rest of the elements. In the nominal group there is always a Head but the Thing may be missing and so the Head element is conflated with either Epithet, Numerative, Classifier or Deictic instead.

(22) (Have) a cup of tea.

(23) The old shall pass first.

(24) I'll give you three.

Consider Example 22 analysed with Sydney and Cardiff grammars in Table 3.7. In the Sydney Grammar the semantic and the syntactic heads differ. In the experiential analysis the semantic head is "tea" which functions as Thing, while in the logical analysis the syntactic head is "cup" which functions as Head. Cardiff Grammar does not offer multi-structural analysis and there is no Head/Thing distinction. The functional elements are already established based on semantic criteria and this is further discussed in Section 3.5.3.

		a	*cup*	*of*	tea
Sidney Grammar	experiential	Numerative			Thing
	interpersonal	Pre-Modifier	Head	Post-Modifier	
Cardiff Grammar		Quantifying Determiner		Selector	Head

Table 3.7 Analysis of Example 22 with Sydney and Cardiff grammars: diverging semantic and syntactic heads.

In the nominal group "The old" which is Subject in Example 23, the Head is adjective "old" and not a noun as would normally be expected. The noun modified by the adjective "old", also the *pivotal element* of the group defined in Section 3.3.2, is left covert and it should consequently be recoverable anaphorically or cataphorically from the context. We can insert a generic noun "one" to form a canonical noun group "the old one". In such cases when the pivotal noun is missing, the logical Head is conflated with other element in this case the Epithet. The group class is not raised from the word class to quality group but is identified by internal structure of the whole group and in this case the presence of determiner signals a nominal class. Similarly, in Example 24, "three" in Sydney grammar is a nominal group where the Thing is missing and the Head has shifted left towards the Numeral. With examples as these ones, Fawcett argues that none of the constituting elements of the unit is mandatorily realised, even the so called pivotal element which is the group defining element. In Chapter 5 is provided an in depth description of the recovering mechanisms for covert nominal elements at the level of the clause.

In this work I adopt the principles for establishing the logical structure of Sydney Grammar. It resonates closely with the traditional "semantically blinded" grammars because and it always provides a Head element even if it differs from the syntactically motivated pivotal element in Cardiff Grammar. Moreover these logical Heads correspond to dependency heads established in the Stanford dependency parse. Chapter 4 provides the grounds for cross-theoretical mappings and the empirical evaluation in Chapter 9 validates it.

In language is not unusual to have nominal groups with the Thing missing or elliptic clauses with the Main verb missing therefore no rigid correspondence can be established between the logical Head and unit class. In this thesis the structure creation is performed in two steps first establishing the group boundaries and the unambiguous unit elements through a top down perspective (that is Sydney approach to unit creation) and second for each established group evaluate the internal structure in order to establish the group class (that is Cardiff approach to group formation). This process is detailed in the Chapter 7.

The evaluation in the second step, besides finalising the syntactically motivated unit structure, can as well assign semantically motivated unit structure. This part however is left out in the current thesis for the groups and only the clauses receive semantic role labels and process types described in Chapter 8.

3.4.6 Coordination as unit complexing

In Sydney Grammar unit complexes fill an important part of the grammar along with the *taxis relations* (Definition 3.2.14) which express the interdependency relations in unit complexes. *Parataxis* relations bind units of equal status while the *Hypotaxis* ones bind the dominant and the dependant units. Fawcett bypasses the taxis relations replacing them with coordination and embedding (Fawcett 2000: 271) leading to abandonment of unit complexing entirely. While embedding elegantly accounts for the depth and complexity of syntax, this approach to coordination is problematic.

Hereafter I discuss the utility and even necessity of keeping unit complexes in parsing. In particular I address the treatment of group and clause *coordination* but the same principle applies to fixed idiomatic structures such as *comparatives*, *conditionals* or *appositions*.

Treatment of the coordination phenomena is a challenge not only for SFL but for other linguistic theories as well. Sydney Grammar approaches it through unit complexing and taxis relations while Cardiff Grammar treats this phenomena as multiple distinct units filling or expounding the same element.

Table 3.8 illustrates an example analysis where the Complement is filled by a homogeneous nominal group complex held together through *paratactic extension* where the first element is a nominal group and the second is nominal group together with the conjunction which is not part of the experiential structure but remains accounted only in the logical structure of the nexus.

Ike	*washed*	*his*	*shirt*	*and*	*his*	*jeans*
Subject	Predicate/Finite	Complement				
		1		+2		
		Deictic	Thing		Deictic	Thing

Table 3.8 Clause with nominal group complex

In Table 3.9 the Epithet is filled by a nexus of paratactic extension. The first element of the nexus is the word "immediate" and the second element is the set of words "and not so far distant". The "not so far distant" is an adverbial group with a logical structure of sub-modifiers already discussed in Section 3.4.1 and the conjunction "and" is left implicitly part of the logical structure of the nexus creating a gap in the structure that is addressed in this discussion. Also note that, in Sydney Grammar the coordination is accounted as unit complex ensuring that only one unit fills an element of the parent, in contrast, as we will see below, to Cardiff Grammar.

the	*immediate*	*and*	*not*	*so*	*far*	*distant*	*future*
Modifier							Head
γ	β						α
Deictic	Epithet						Thing
	1	+2					
			Sub-Modifier			Sub-Head	
			δ	γ	β	α	

Table 3.9 Nominal group with word complex from (Halliday & Matthiessen 2013: 564)

In Table 3.10 is presented an example of analysis with Cardiff Grammar. The Complement is filled by two *sibling* nominal groups "his shirt" and "and his jeans" that are both of them fill the same element in accordance to Definition refdef:coordination. The conjunction "and" is accounted directly as part of the nominal group structure.

Opinions are divided (between Sydney and Cardiff schools) whether to invite the notion of a complex unit to handle coordination or not. If we side with Cardiff grammar and dismiss the unit complex then we allow an element to be filled by more than

Ike	*washed*	*his*	*shirt*	*and*	*his*	*jeans*
Subject	Main Verb	Complement				
		Deictic Determiner	Head	&	Deictic Determiner	Head

Table 3.10 Coordination analysis in Cardiff Grammar

one units. And this is a problem because if we do not account unit elements each in a unique place within the unit structure then we loose the capacity to order them. Therefore in this thesis I adopt the Sydney definition of structure (Definition 3.2.6) that constraints each element into a single place that is filled by another unit. Therefore the conjunction must be a nexus acting as a single unit filling a single element.

I argue for adoption of such unit type in order to ensure that maximum one unit can fill the place of an element. In the theory of grammar, only units are accounted for structure while the elements can only be filled by an unit (see Figure 3.2). Allowing multiple units to fill an element requires accounting at least for the *order* if not also for the relation between the filler units. The structure as it is described in theories of grammar by Halliday (Halliday 2002) and Fawcett (Fawcett 2000) is defined by the unit and not the element. There is no direct reference in the theory to the unit ordering. Instead, the order relation is accounted in the structure through the concept of place. A unit has a specific possible structure in terms of places of elements which hold absolute position in the unit structure or relative one to each other. Therefore if an element is filled by two units simultaneously it constitutes a violation of the above principle as the order of those units is not accounted for but it matters which is easy to show in the following examples.

(25) (Both my wife and her friend) arrived late.

(26) * (And her friend both my wife) arrived late.

(27) I want the front wall (either in blue or in green).

(28) * I want the front wall (or in green either in blue).

If the order would not have mattered then we could say that the conjunctions from the example 25 can be reformulated into 26 and the one from 27 into 28. But such reformulations are grammatically incorrect. Obviously the places do matter and they need to be accounted in the unit structure as one element per place with no more than a single unit filling it.

I turn now to the role and position of lexical items signalling the conjunction which I consider having no place in the structure of the conjuncted units but outside of them, that way forming together a higher order unit, the *complex unit*. This is contrary to what is being described in Cardiff and Sydney grammars for different reasons.

Fawcett present the Linker elements (&) which are filled by conjunctions as parts of virtually any unit class placed in the first position of the unit. For example in the "or in green" the presence of "or" signals the presence at least of one more unit of the same nature and does not contribute to the meaning of the prepositional group but to the meaning outside the group requiring presence of a sibling. Even more, the lack of a sibling most of the time would constitute an ungrammatical formulation. The only potential objection here is for the perfectly acceptable cases of clauses/sentences starting with a conjunction such as "and" or "but". In those cases the conjunction plays a textual function and still invites the presence of a sibling clause/sentence preceding the current one to be resolved in a clause complex or discourse level.

Halliday omits to discuss in IFG (Halliday & Matthiessen 2013) the place of Linkers. He implicitly proposes the same as Fawcett through his examples of paratactic relations at various rank levels (Halliday & Matthiessen 2013: 422, 534, 564, 566) that the lexical items signalling conjunction are included in the units they precede in the logical structure but not the experiential one. The main insufficiency here is that the logical structure does not provide any meaningful elements or unit class but some sort of proto-elements that resemble rather places than any functions. In this sense I consider treatment of conjunctions insufficiently accounted in IFG.

So conjunctions and pre-conjunctions shall not be placed inside the conjuncted units because they do not contribute to their meaning. They shall be enclosed as Linkers into unit complexes. But if we adopt the unit complexing then we need to define a unit structure. Hence I propose the following generic structure for the *coordination unit*.

Pre-Linker	Initiating Conjunct	... Conjunct ...	Linker	Conjunct
1		+ 2 ... + n_{-1}	+ n	

Table 3.11 Generic structure of the coordination unit

In Table 3.11 the first row presents a series of Conjuncts where the first one is initiating or the head and the rest are continuation Conjuncts of the former. In the first place there may be a Pre-Linker element such as "both" or "either" for example but it is optional and in the place before the last one is located the Linker element that determines the type of coordination. On the second row I provide, for orientation purposes, the Sydney logical structure of a paratactic expansion applied to the coordination unit complex. Note that the Pre-Linker and the Linker elements are merged with the conjuncted units.

Applying this structure to the previous example yields analysis such as in Table 3.12. The nominal group has the Epithet element filled by a coordination group formed of two Conjuncts and a Linker.

the	*immediate*	*and*	*not*	*so*	*far*	*distant*	*future*
Determiner	Epithet						Head
	Initiating Conjunct	Linker	Conjunct				

Table 3.12 Example analysis with coordination unit complex structure

Adopting the unit complex and in particular coordination unit requires two more clarifications (1) does the complex unit carry a syntactic class, and if so according to which criteria is it established? (2) Does it have any intrinsic features or all of them are inherited from the conjuncts?

Zhang states in her thesis that the coordinating constructions do not have any categorial features thus there is no need to provide a new unit type. Instead the categorial properties of the conjuncts are transferred upwards (Zhang 2010). For example if two nominal groups are conjuncted then the complex receives the nominal class.

This principle holds for most of the cases however there are rare cases when the units are of different classes. Consider 29 analysed in Table 3.13 where the conjuncts are a nominal group "last Monday" and a prepositional group "during the previous weekend".

(29) I lost it (either last Monday or during the previous weekend).

either	*last*	*Monday*	*or*	*during*	*the*	*previous*	*weekend*
Pre-Linker	Initiating Conjunct		Linker	Conjunct			

Table 3.13 Coordination group with mixed class conjuncts

In this case there are two unit types that can be raised and it is not clear how to resolve this case. Options are (a) to leave the generic class *coordination complex*, (b) transfer the class of the first unit upwards, or (c) semantically resolve the class as both represent temporal circumstances even if they are realised through two different syntactic categories. This means that if no sub-classification is provided based on the constituent units below than there is no need to project/transfer upward the class of the conjunct units. In this work I decided to leave the class generic and leave for the future an extensive unit complex classification.

I turn now to the last issue of this discussion, specifically whether the complex unit may have intrinsic features emerging from the conjunct elements.

In the example 30 the conjunction of two singular noun groups requires plural agreement with the verb. Even though semantic interpretation that only one item is selected at a time, syntactically both items are listed in the clause and attempting third person singular verb forms in 31 is grammatically incorrect. That leads to the conclusion that the coordination complex can have categorial features which none of the constituting units has.

(30) A pencil or a pen **are** equally good as a smart-phone.

(31) * A pencil or a pen **is** equally good as a smart-phone.

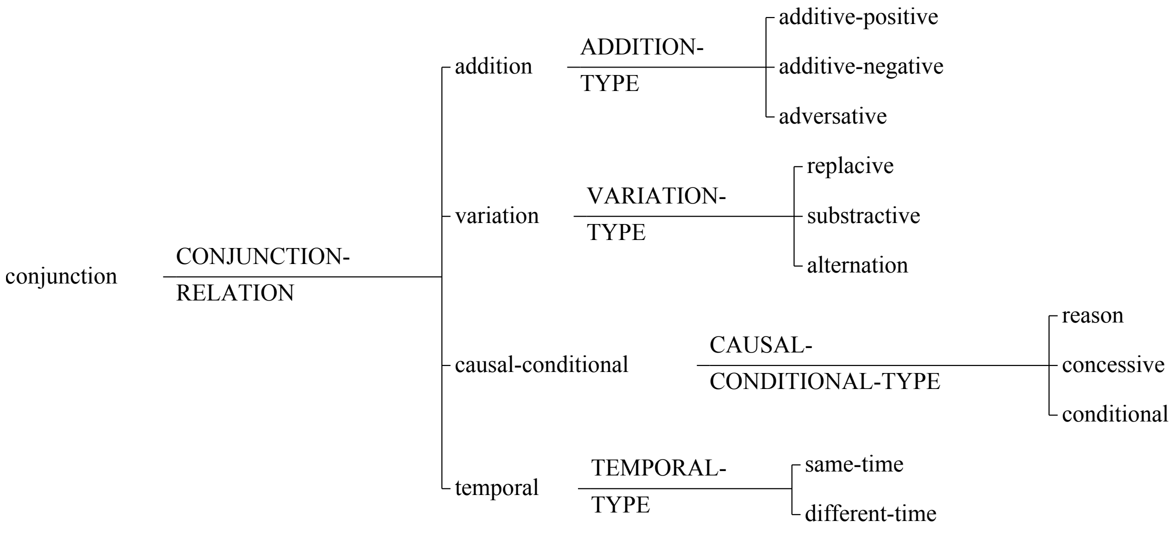


Fig. 3.11 Systemic network of coordination types

In the case of nominal group conjunction we can see that the plural feature emerges even if each individual unit is singular. For other unit classes it is not so obvious whether there are any linguistic features that emerge at the conjunction level. The meaning variation is semantic as for example conjunction of two verbs or clauses might mean very different things such as consecutive actions, concomitant actions or presence of two states at the same time and so on. This brings us to another feature of the coordination complex - the type of the relationship it constructs. The lexical items expounding the Linker and Pre-Linker (e.g. *and*, *or*, *but*, *yet*, *for*, *nor* or *so*) are indicator of relationship among the conjuncts and together can be systematised as the relationship types in the systemic network in Figure 3.11.

This section laid out how and why I treat the coordination phenomena in parsing. I adopt the unit complexing mechanism with taxis relations described in Sydney grammar in order to account for a new unit class, the *coordination unit*. I do that to ensure that each element of a unit is filled by no more than one other unit contrary to what Cardiff grammar proposes (see Definition 3.3.7). But taxis relations in Sydney grammars are represented via logical structures which is not rich enough to account for internal structure of the coordination unit. Therefore I also propose here a unit structure in terms of ordered functional elements just as for the rest of unit classes.

3.5 The grammatical units for parsing

Now that the important theoretical details have been covered I would like focus on the specific grammars of the two schools. They have common parts and also differ in large parts on their paradigmatic and syntagmatic descriptions. This section discusses the main units considered in each of the grammars. As in the previous section I argue on pragmatic grounds for the adoption of unit structures from one or the other grammar. The argument runs along the lines that some unit structures are closer to the traditional syntactic analysis and thus are easier to detect and parse and the other ones may be a level of abstraction higher falling more on semantic grounds and thus becoming more difficult to capture in structural variance and requiring lexico-semantic resources.

3.5.1 Verbal group and clause boundaries

In Sydney Grammar the verbal group is described as an expansion of a verb just like the nominal group is the expansion of the noun(Halliday & Matthiessen 2013: 396). There are certainly words that are closely related and syntactically dependent on the verb all together forming a unit that functions as a whole. For example the auxiliary verbs, adverbs or the negation particles are words that are directly linked to a lexical verb. The verb group functions as Finite + Predicator elements of the clause in Mood structure and as Process in Transitivity structure.

In Cardiff Grammar the verb group is dissolved moving the Main Verb as the pivotal element of the Clause unit. All the elements that form the clause structure and those that form the verb group structure are brought up together to the same level as elements of a clause. The clause structure in Cardiff Grammar comprises elements with clause related functions(like Subject, Adjunct, Complement etc.) and other elements with Main Verb related functions(Auxiliary, Negation particle, Finite operator etc.).

Regarded from the Hallidayan rank scale perspective, merging the elements of the verb group into clause structure is not permitted because the units are at different ranks. However it is not a problem for the relaxed rank scale version presented in subsection 3.4.1. The reason for adopting such an approach is best illustrated via complex verb groups with more than one non-auxiliary verb such as in examples 32–34.

I begin by addressing the impact of this merger on (a) the clause structure (b) the clause boundaries and (c) semantic role distribution within the clause.

(32) (The commission **started to investigate** two cases of overfishing in Norway.)

(33) (The commission **started** (**to investigate** two cases of overfishing in Norway.))

(34) (The commission **started** (**to finish** (**investigating** two cases of overfishing in Norway.)))

In Sydney Grammar "started to investigate" (in 32) is considered a single predicate of investigation which has specified the aspect of event incipiency despite the fact that there are two lexical verbs within the same verbal group. The "starting" doesn't constitute any kind of process in semantic terms but rather specifies aspectual information about the investigation process. This is argued by looking at the conditions on the participants and it is equivalent in a formal approach to looking at where the selection restrictions for complements come from. The boundaries of the clause governed by this predicate stretch to the entire sentence.

Semantically it is a sound approach because despite the presence of two lexical verbs there is only one event. However allowing such compositions leads to unwanted syntactic analysis for multiple lexical verb cases in examples such as 34. To solve this kind of problem Fawcett dismisses the verb groups and merges their elements into clause structure. He proposes the syntactically elegant principle of *one main verb per clause* (Fawcett 2008). Applying this principle to the same sentence yields a structure of two clauses illustrated in example 33 where the main clause is governed by the verb "to start" and the embedded one by the verb "to investigate". Note the conflict between "one main verb per clause" with Halliday's principle that only whole units form the constituency of others (the (c) principle of rank scale described in subsection 3.4.1). So allowing incomplete groups into the constituency structure would breach the entire idea of unit based constituency.

Semantically the clause in SFL is a description of an event or situation as a figure with a process, participants and eventually circumstances where the process is realised through a lexical verb. Looking back to our examples, does the verb "to start" really describes a process or merely an aspect of it? Halliday treats such verbs as aspectual and when co-occurring with other lexical verbs they are considered to form a single predicate. Accommodating Fawcett's stance, mentioned above and contradicting Halliday's approach, requires weakening the semantic requirement and allowing aspectual verbs to form clauses that contribute *aspectually or modally* to the embedded ones. I mention also the modal contribution because some verbs like *want, wish, hope* and others behave syntactically like the aspectual ones. Moreover, Fawcett introduces into the Cardiff Grammar Transitivity network an *influential* process type including all categories of meanings that semantically function as process modifiers: tentative, failing, starting, ending etc.

I adopt here Fawcett's "one main verb per clause" principle which as a consequence changes the way clauses are partitioned, leads to abolition of the verbal group and introduces the "influential" process type. Next I discuss it's impact on the structure of the clause unit.

3.5.2 The Clause

It is commonly agreed in linguistic communities that the unit of the clause is one of the core elements in human language. The main clause constituents are roughly the same in SFL as the ones in the traditional grammar (Quirk et al. 1985), transformational grammar (Chomsky 1957a) an indirectly in dependency grammar (Hudson 2010).

In current work I adopt the Cardiff Grammar clause structure with the *Main Verb* as pivotal element. Though there is no element that is obligatorily realised in English, I consider in the current work, the Main Verb to be way to flag a clause. In SFL are described clauses without a main verb such as minor clauses (exclamations, calls, greetings and alarms) that occur in conversational contexts and elliptical clauses Halliday & Matthiessen (2013) such as the one in example 35, none of which are covered in the present work.

(35) They were in the bar, *Dave in the restroom and Sarah by the bar.*

I adopt Cardiff Grammar clause structure for English. It is formed of the *Subject*, *Finite*, *Main Verb*, up to two *Complements* and a various number of *Adjuncts*. All the elements of the assumed verbal group are part of the clause as well such as Auxiliary Verbs, Main Verb Extensions, Negators etc. (see Appendix **??** for a complete list).

3.5.3 The Nominal Group

The nominal group expresses things, classes of things or a selection of instances in that class. This section argues for adoption of the Sydney grammar noun group structure with a specific extension. It is that the elements of the nominal group can be filled, in addition to word classes, by the groups as well. This possibility is opened by the rank scale relaxation (Section 3.4.1) and Cardiff embedding principle (Definition 3.3.9). In addition to that I argue below for working on semantic and syntactic heads in two steps. First create the structure with the syntactic one (the Head) and then derive the semantic one (the Thing).

those	*two*	*old*	*electric*	*trains*	*from Luxembourg*
Pre-Modifier				Head	Post-Modifier
Deictic	Numerative	Epithet	Classifier	Thing	Qualifyier
determiner	numeral	adjective	adjective	noun	prepositional phrase

Table 3.14 An example of a nominal group in the Sydney Grammar (Halliday & Matthiessen 2013: 264)

In Table 3.14 an example analysis is presented of the nominal group proposed in the Sydney grammar (Halliday & Matthiessen 2013: 364–369). The Sydney nominal group is constituted by a head nominal item modified by descriptors or selectors such as: *Deictic*, *Numerative*, *Epithet*, *Classifier*, *Thing* and *Qualifier*. Each element has a fairly stable correspondence to the word classes, expected to be expounded by lexical items. Table 3.15 presents the mappings between the elements of nominal group and the word classes.

Experiential function in noun group	**class (of word or unit)**
Deictic	determiner, predeterminer, pronoun, adjective
Numerative	numeral(ordinal or cardinal)
Epithet	adjective
Classifier	adjective, noun
Thing	noun
Qualifier	prepositional phrase, clause

Table 3.15 Mapping of noun group elements to classes (Halliday & Matthiessen 2013: 379)

Inspired from Cardiff grammar, in addition to word classes, the elements of the nominal group can also be filled by the group classes corresponding to each word class above. This way the Numerative, in addition to words, can be filled by a noun group, Epithet by an adjectival group, Classifier by an adjective or noun group and finally each of the elements can be filled by a coordination group discussed in Section 3.4.6.

The elements in Cardiff Grammar differ from those of Sydney Grammar. Table 3.16 exemplifies a noun group analysed with Cardiff Grammar covering all the possible elements. Table 3.17 provides a legend for the Cardiff Grammar acronyms along with mappings to unit and word classes that can fill each element.

or	*a photo*	*of*	*part*	*of*	*one*	*of*	*the best*	*of*	*the*	*fine*	*new*	*taxis*	*in Kew*	*,*
pre-modifiers												head	post-modifiers	
&	rd	v	pd	v	qd	v	sd or od	v	dd	m	m	h	q	e

Table 3.16 The example of a nominal group in Cardiff Grammar

symbol	**function meaning**	**class (of word or unit)**
rd	representational determiner	noun, noun group
v	selector "of"	preposition
pd	partitive determiner	noun, noun group
fd	fractional determiner	noun, noun group, quantity group
qd	quantifying determiner	noun, noun group, quantity group
sd	superlative determiner	noun, noun group, quality group, quantity group
od	ordinative determiner	noun, noun group, quality group
td	tipic determiner	noun, noun group
dd	deictic determiner	determiner, pronoun, genitive cluster
m	modifier	adjective, noun, quality group, genitive cluster
h	head	noun, genitive cluster
q	qualifier	prepositional phrase, clause
&	linker	conjunction
e	ender	punctuation mark

Table 3.17 The mapping of noun group elements to classes in Cardiff grammar

The elements in Cardiff Grammar are based on semantic criteria supported by lexical and syntactic choices. Consequently some elements cannot be derived based on solely syntactic criteria, requiring semantically motivated lexical resources. Semantically bound elements which are a challenge are predominantly determiners *Representational, Partitive, Fractional, Superlative, Typic Determiners* while the rest of the elements:

Head, Qualifier, Selector, Modifier and Deictic, Ordinative and Quantifying Determiners can be determined solely on the syntactic criteria. The latter correspond fairly well to the Sydney version of nominal group which is adopted in the present work with the benefits of the relaxed rank system replacing the sub-structures with embedded units and simplifying the syntactic structures.

Another simplification is renouncing to distinction between the Head and Thing (Halliday & Matthiessen 2013: 390–396) discussed in Section 3.4.5. Thus if the logical Head of the nominal group is a noun then it is labelled as the Thing leaving the semantic discernment as a secondary process and out of the current scope. Otherwise, in cases of nominal groups without the Thing element, if the Head is a pronoun (other than personal), numeral or adjective (mainly superlatives) then they function as Deictic, Numerative or Epithet. So, as will be described in Chapter 7, I propose to parse the nominal groups in two steps: first determine the main constituting chunks and assign functions to the unambiguous ones and second perform a semantically driven evaluation for the less certain units.

Next I explain the two step process using for illustration cases when the Thing is present but it is different from the Head such as in examples 36–38.

(36) (a cup) of (tea)

(37) (some) of (those youngsters)

(38) (another one) of (those periodic eruptions)

These nominal groups can be analysed in two ways. Either being about the "cup", "some" or "another one" leading to a structure where the first noun is the head succeeded by a prepositional phrase Qualifier; or rather about "tea", "youngsters" and "eruptions" where the second noun is the head and so adopting a structure with complex determiners.

Table 3.18 shows on the first row an analysis with syntactic head i.e. the Head defined in Sydney grammar and on the second row an analysis with semantic head i.e. the Head defined in Cardiff grammar that also coincides with the Thing from Sydney grammar.

The syntactic Head is always the first noun in the nominal group. In the semantic evaluation phase special attention is given to Qualifiers filled by prepositional phrases starting with "of" preposition and whether the nominal group may function as qualifying, quantifying, ordination or other type of determiner.

Cardiff Grammar weakens the assumption that every prepositional phrase acts as Qualifier in a nominal group and it the special case of the preposition "of". It is allowed to act not as the element introducing a prepositional phrase but as a end mark of a determiner-like selector. Thus making the former noun group a determiner in the latter one.

If the above conditions are satisfied, in the semantic evaluation phase, then the prepositional phrase Qualifyier is disassembled, the the preposition "of" is ascribed as a Selector element of the nominal group (in a way an upwards transfer) and the former nominal group (syntactically headed) becomes one of the determiners. This approach shifts the noun group head into the position of semantically based Thing and erases the discrepancy problem between them.

a	*cup*	*of*	*tea*
Determiner	Head	Qualifier	
Quantifying Determiner		Selector	Head/Thing

Table 3.18 Example analysis with syntactic and semantic heads

(39) He was the **confidant** of the prime minister.

(40) It was the **clash** of two cultures.

The above explanation is not a straight forward solution. The distinction between cases when the proposition "of" introduces a Qualifier or ends a Selector/Deictic requires lexical-semantic informed decision answering the question "what is the Thing that this nominal group is about?". And there is a lot of space for variations the syntactic structure. For example in 39 (Head/Thing marked in bold) the preposition "of" introduces Qualifiers.

In this section was discussed in detail the problem of semantic and syntactic heads (started in Section 3.4.5) applied to nominal groups in particular and how to approach parsing them. I conclude that it is fairly unambiguous and straight forward to determine the structure of nominal groups according to Sydney grammar yielding a syntactically founded structure. Once such a structure is ready it serves a a proper basis for a semantic evaluation that can be performed in terms of Cardiff grammar resulting in potential restructuring of the nominal group. The current implementation of the parser only the generation of syntactic structure is implemented leaving the semantically motivated noun groups for the future works.

3.5.4 The Adjectival and Adverbial Groups

This section introduces how the adverbial and adjectival groups are handled by the Sydney grammar and then how their equivalent quality group is structured in the

Cardiff grammar. As the structure of the quality group is semantically motivated some elements may be identified still at the syntactic level whereas some other ones require a more sophisticated lexical-semantic resources. In the last part of the section I estimate the complexity of parsing some the quality group elements elements.

Following the rationale of head-modifier similar to the case of nominal groups, the adjectives and adverbs function as pivotal elements to form groups. The structure of adverbial and adjectival constructions is briefly covered in the Sydney grammar in terms of head-modifier logical structures without an elaborated experiential structure like in the case of nominal groups. While the adverbial group is recognised as a distinct syntactic unit, the adjectival group is treated as a special case of nominal group.

(41) You're *a very lucky boy.*

(42) You're *very lucky.*

(43) *The very lucky (one)* is you.

In the environments where nominal group functions as Attribute, typically in the attributive clauses such as 41, it can take also more contracted forms without the Thing and Deictic where the Head moves left onto the Epithet such as in example 42. One particularity of these nominal groups which here are distinguished as *adjectival group* units is that they cannot function as subject. For the example 43 to be grammatical, where the Attribute is in the Subject position, I had to add a determiner and eventually an unspecified nominal Head.

The *adverbial group*, in Sydney Grammar, has an adverb as Head which may or may not be accompanied by modifying elements (Halliday & Matthiessen 2013: 419). The adverbial groups may fill modal and circumstantial adjunct elements in a clause corresponding to eight semantic classes of: time, place, four types of manner and two types of assessment. The adverbial pre-modifiers express polarity, comparison and intensification along with only one comparison post-modifier (Halliday & Matthiessen 2013: 420–421). The adjectival and adverbial group are covered by the *quality group* unit in Cardiff grammar.

A thorough systemic functional examination in terms of lexis has been provided for the first time by Tucker Tucker (1997, 1998) materialised into a lexical-grammatical systematization of adjectives and the fine grained structure of quality group. He avoids calling the group according to the word class (adjective or adverb) but rather refers to the semantic meaning of what both groups express, i.e. the quality of things, situations or qualities themselves. The qualities of things have adjectives as their head while the qualities of situations an adverb.

In Cardiff Grammar, the head of the quality group is called *Apex* while the set of modifying elements: *Quality Group Deictic, Quality Group Quantifier, Emphasizing Temperer, Degree Temperer, Adjunctival Temperer, Scope* and *Finisher*. The quality group most frequently fills complements and adjuncts in clauses and fill modifiers and superlative determiners in nominal groups but there are also other cases found in the data.

Just like in the case of nominal group the adverbial and adjectival groups in Cardiff grammar are semantically motivated. To automatically identify elements of the quality group would according to this scheme therefore require lexico-semantic resources.

I turn now to discuss some relevant affinities concerning the adverbial groups.

Some adverbs are different from others at least because not all of them can be heads of the adverbial group. Usually the adverbs that cannot act as heads, such as for example *very, much, less, pretty*, function as Emphasizing and Degree Temperers. The same ones also act as adjectival modifiers. A naive attempt to identify these Temperers would be to use a list of frequent words found in these functions.

Other elements of the quality group like the *Scoper* or *Finisher* are more difficult to identify and localize as part of the group only by syntactic cues and/or lists of words because of their inherent semantic nature. The problem is similar to detecting whether a prepositional phrase is filling a qualifier element in the preceding nominal group or is filling a complement or adjunct in the clause. Not surprisingly the Scopers and Finishers are most of the time prepositional phrases.

Another issue is continuity. The question is whether a grammar shall allow at least at a syntactic level discontinuous constituents or not. And then if so how to detect all the parts of the group even if they do not stand in proximity to each other. For example, comparatives, a complex case of a quality group, could be realised in a continuous or discontinuous forms. Compare the analyses presented in Table 3.19 and 3.20. In the first case the comparative structure is a continuous quality group. In the second case the comparative is dissociated and analysed as separate adjuncts.

On one hand it is not a problem treating them as two adjuncts, because that is what they are from the syntactic point of view. However, semantically as Fawcett proposes, there is only one quality group with a discontinuous realisation whose Scope element is placed in a thematic position before the Subject.

I	*am*	*much*	*smarter*	*today*	*than*	*yesterday*
Subject	Main Verb	Adjunct				
pronoun	verb	quality group				
		Emphasizing Temperer	Apex	Scope	Finisher	

Table 3.19 Comparative structure as one quality group adjunct

Today	*I*	*am*	*much*	*smarter*	*than*	*yesterday*
Adjunct	Subject	Main Verb	Adjunct			
adverb	pronoun	verb	quality group			
			Emphasizing Temperer	Apex	Finisher	

Table 3.20 Comparative structure split among two adjuncts

For an automatic process to identify a complex quality group is a difficult task. It needs to pick up cues like a comparative form of the adjective followed by the preposition “than” and then look for two terms being compared. Given some initial syntactic structure such patterns could be modelled and applied but only as a secondary semantically oriented process.

Since both the adverbial and adjectival groups have similar structures, it is syntactically feasible to automatically analyse them in terms of head-modifier structures in a first phase followed by a complementary process which assigns functional roles to the quality group components.

3.6 Selected system networks for parsing

3.6.1 Mood

3.6.2 Transitivity

3.7 Discussion

This chapter has introduced the fundamentals of systemic functional linguistics and presented a consideration of Sydney and Cardiff theories of grammar to the task of parsing.

Because of its bottom up approach to unit structure, rank scale relaxation and accommodation of embedding as a general principle, Cardiff systemic functional theory is more suitable for parsing than the Sydney one. Nonetheless the unit definitions in the Cardiff grammar are deeply semantic in nature. Parsing with such units requires most of the time lexical-semantically informed decisions beyond merely syntactic variations. This is one of the reasons why the parsing attempts by O'Donoghue (1991) and others in the COMMUNAL project were all based on a corpus.

As there was no corpus available and because the parsing approach is based on a syntactic backbone none of the theories could be fully used as such. Section 3.4 and 3.5 attempts to merge and adapt the grammars and theories of grammar to the parsing approach of this thesis.

Next chapter lays the theoretical foundations of Dependency Grammar and introduces the Stanford dependency parser used as a departing point in current parsing pipeline. Because there is a transformation step from dependency to systemic functional consistency structure, the next chapter also covers a theoretical compatibility analysis and how such a transformation should in principle look like.

Chapter 4

The dependency grammar

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

4.1 Origins of the dependency theory

For the first time a complete linguistic theory based on the dependency concept was elaborated by the French linguist Lucien Tesniere in his seminal work *"Elements de syntaxe strusturale"* published in 1959 after his death. He devoted much effort to argue for the adequacy of *dependency* as the organizational principle underlying numerous phenomena and in fact attempting to demonstrate the universality of his syntactic analysis method for human languages. In doing so he introduced a series of concepts and ideas among which the *verb centrality*, *stratification*, *language typology*, *nuclei*, *valency*, *metataxis*, *junction* and *transfer* are the most important ones which I introduce following the connections.

> The sentence is an *organized set*, the constituent elements of which are the words. Each word in a sentence is not isolated as it is in the dictionary. The mind perceives *connections* between a word and its neighbours. The totality of these connections forms the scaffold of the sentence. These connections are not indicated by anything. But it is absolutely crucial that

> they be perceived by the mind; without them the sentence would not be intelligible. (Tesniere 2015: 3)

Tesniere holds the view that the connection, what is know today as *dependencies*, are the foundations of the *structural syntax* known as *dependency grammar* today. According to him “to construct a sentence is to breathe life into an amorphous mass of words, establishing a set of connections between them. Conversely, understanding a sentence involves seizing upon the set of connections that unite the various words” (Tesniere 2015: 4). He introduces the hierarchy of connections as follows.

> Structural connections establish *dependency* relations between words. In principle, each connection unites a superior term and an inferior term. The superior term is called the *governor*, and the inferior term the *subordinate*. We say that the subordinate depends on the governor and that the governor governs the subordinate. [. . .] A word can be both subordinate to a superior word and governor of an inferior word. [. . .] The set of words of a sentence constitutes a veritable *hierarchy*. (Tesniere 2015: 5–6)

Introduction of hierarchy and governor-subordinate dependencies permeates to define now what is a *node* and the *stemma* resembling what is now known as *dependency tree* (although the stemmas do not include labels on the tree edges).

> [. . .] In principle, a subordinate can only depend on a sole governor. A governor, in contrast, can govern multiple subordinates [. . .] Every governor that governs one or more subordinates forms what we call a node. [. . .] it follows that *each subordinate shares the fate of its governor*. (Tesniere 2015: 6)

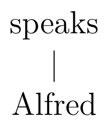


Fig. 4.1 Stemma for “Alfred speaks”

This asymmetry of connection permits construction of a tree-like structure. The diagram of the two word sentence “Alfred speaks” is provided in the Figure 4.1. The word “speaks” is the governor of the word “Alfred”. The connection is depicted bu the vertical line connecting the two. But to make it complete it is important to decide on the root node.

> The node formed by the governor that governs all the subordinates of a sentence is the *node of nodes*, or the central node. It is at the centre of the sentence and ensures its structural unity by tying the diverse elements into a single bundle. It can be identified with a sentence. *The node of nodes is generally verbal* [...] (Tesniere 2015: 7)

The fundamental insight presented above about the nature of the syntactic structure concerns the grouping of words at the clause level. Tesniere rejects the subject-predicate formation that was the de facto syntactic understanding of his time. He argued that this division is belongs to Aristotelian logic and is not associated to linguistics. Instead of the subject-predicate division Tesniere positions the verb at the root of the clause structure making the subject and the object subordinated seedlings. Figure 4.2 depicts the clause structure "Alfred speaks slowly" where both the subject and the object are subordinated to the central verb speaks.

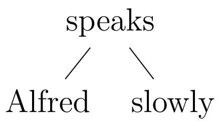


Fig. 4.2 Stemma for "Alfred speaks slowly"

Tesniere is among pioneer linguists recognising that the language is organised at different levels thus advocating a *stratified model of language*. He recognises the to dimensional syntactic representation and the one dimensional chain of spoken language.

> *speaking* a language involves transforming structural order to linear order, and conversely, *understanding* a language involves transforming linear order to structural order. The fundamental principle of transforming structural order to linear order involves changing the connections of structural order into the sequences of linear order. This transformation occurs in such a manner that the elements connected in structural order become immediate neighbours in the spoken chain (Tesniere 2015: 12).

In the structural realm Tesniere goes even deeper and describes the separation between syntax and semantics. To argue for that, he uses an example similar to the famous Chomskian *colourless green ideas sleep furiously* (Chomsky 1957b) (that occurred three years after Tesniere's death). He employed the sentence *the vertebral silence antagonizes the lawful sail.*

> Syntax is distinct from morphology, and it is no less distinct from semantics. The structure of a sentence is one thing, and the idea that it expresses and that constitutes its meaning is another. It is therefore necessary to distinguish between the structural plane and the semantic plane. [. . .] The structural plane and the semantic plane are therefore entirely independent of each other from a theoretic point of view. The best proof is that a sentence can be semantically absurd and at the same time syntactically perfectly correct. (Tesniere 2015: 33).

Tesniere distinguishes between *nodes* and *nuclei.* Initially he defines the node in a way that resembles the phrase or a constituent but after that he changes his mind.

> we define a *node* as a set consisting of a governor and all of the subordinates that are directly or indirectly dependent on the governor and that the governor in a sense links together into a bundle. (Tesniere 2015: 6).

Latter in the book, he uses the term node to mean merely a vertex and even redefines it saying that "The node is nothing more than a geometric point whereas the nucleus is a collection of multiple points . . . " (Tesniere 2015: 39). It is perhaps the inconsistent use of the terminology that lead to the assumption that the dependency grammar does not recognises phrases (i.e. that is the complete subtree of a vertex). In fact he defines nucleus as playing the role of both a semantic and syntactic unit.

> We define the nucleus as the set which joins together, in addition to the structural node itself, all the other elements for which the node is the structural support, starting with the semantic elements. (Tesniere 2015: 38).

A notable contribution to the field of syntax is the concept of *valency.* It is the notion used in other linguistic schools as *transitivity* to express combinatorial properties of verbs and other lexical items. Inspired from natural sciences, Tesniere compares the relationship between verbs and the so called *actants* (a.k.a. *arguments*) to atom's bonds.

> The verb may therefore be compared to a sort of atom, susceptible to attracting a greater or lesser number of actants, according to the number of bonds the verb has available to keep them as dependents. The number of bonds a verb has constitutes what we call the verb's *valency* (Tesniere 2015: 241).

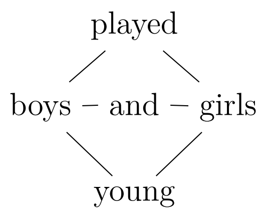
Atoms are not the only metaphor he uses and next I present another one regarding the *verbal node* that is especially important for showing the syntax-semantics interplay.

> The verbal node, found at the centre of the majority of European languages, is a theatrical performance. Like a drama, it obligatorily involves a *process* and most often *actors* and *circumstances*. [...] Transferred from the theatre to structural syntax, the process, the actors, and the circumstances become respectively the *verb*, the *actants*, and the *circumstants* (Tesniere 2015: 97).

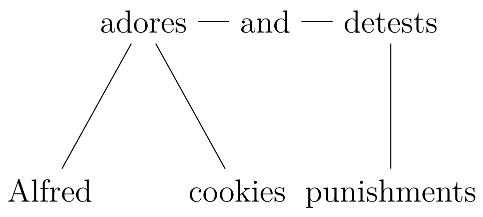
Comparison of the verb to an atom seems to emphasize connection to the syntactic aspect of valency while comparing it to a theatrical performance seems to emphasize the semantic properties of valency. Therefore his theory of valency has semantic and syntactic properties. He believed that the first actant is the agent of the action, identified as the subject in traditional grammar, and the second actant is the one that bears the action, identified as the syntactic object. Tesniere regards both of them as complements to complete the governor verb making, in this sense, the subject indistinguishable from other complements.

There are some phenomena that are deemed quite problematic, namely they are the *coordination* or *apposition*. They constitute a challenge because they are not governor-subordinate relations but are rather orthogonal relations among siblings. Tesniere analyses the coordination, or as he calls it *junction*, as a phenomena used in language to express (semantic) content efficiently.

He viewed the junction as fundamentally different from the subordination and represented it with horizontal lines. Subordination is a principle of organization on the vertical axis whereas the coordination (i.e. junction) on the horizontal axis. Figure 4.3 depicts two example representations for the sentence "Young boys and girls played" and "Alfred adores cookies and detests punishments".



(a) Young boys and girls played



(b) Alfred adores cookies and detests punishments

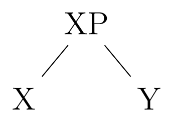
Fig. 4.3 Sample stemmas with *junction* representation

A big part of the Tesniere's *Elements* (Tesniere 1959) is dedicated to the theory of *transfer*. It describes the phenomena when one class of a syntactic unit occupies a position usually devoted to another one. In SFL it is called the grammatical metaphor defined in 3.2.9. For example the noun can be transferred to an adjective by preposition "of", as for example *a linguist of France* where the source *France* is transferred to target *of France* which modify *linguist* that is typically an adjectival function. Transfer is a tool that explains how for example a clause can be embedded into another one or how a verb can be subordinate to another one.

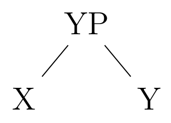
Tesniere splits the words into *function words* or *translatives* (i.e. prepositions, conjunctions, auxiliary verbs and articles) and four basic categories of *content words* (i.e. verbs (I), nouns (O), adverbs (E) and adjectives (A)). The former are empty of content marker transfer of content words from one syntactic category to another one. That is, allowing one word to occupy a position that is generally associated with a word of another category.

One distinguishing trait of the transfer is that the words transferred from source to target category continue to behave as the source category with respect to their dependants and as source category to its governor.

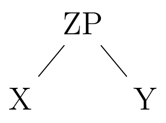
The transfer theory is controversial for the translators of the Elements. They write (Tesniere 2015: liv-lx) that while the transfer schema can not be interpreted in terms of pure dependency it is debatable whether it can be interpreted in terms of constituency. The main distinction is in the number of nodes that one assumes to be in the syntactic structure i.e. whether there are intermediary virtual nodes.



(a) Headed endocentric

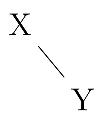


(b) Headed endocentric

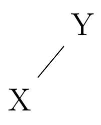


(c) Non-headed exocentric

Fig. 4.4 Constituency structure



(a) Headed endocentric



(b) Headed endocentric

(c) Non-headed exocentric

Fig. 4.5 Dependency structure

Figure 4.4 shows how a sequence of two elements X and Y can be represented in terms of constituency where the Figure 4.4a and 4.4b represent that one element governs the other called *endocentric* structures and in Figure 4.4c a non-headed structure called *exocentric*. Dependency structure depicted below in Figure 4.5, in contrast, cannot represent non-headed structures. Hence there is no correspondent dependency representation to Figure 4.4c in Figure 4.5c.

Bringing back the discussion on the number of nodes, the constituency structure requires three nodes each time whereas dependency structure only two. In this sense the transfer schemas provided by Tesniere in his Elements (Tesniere 1959) resembles constituency structure more than dependency structure simply because it assumes more nodes than words.

4.2 Evolution into the modern dependency theory

Nowadays the dependency theory differs from the original one presented by Tesniere. At the time the original text was written there was no such distinction as dependency and constituency structures and that Tesniere's Elements (Tesniere 1959) in fact contains descriptions of and references to what may nowadays be considered constituency. Next I present which of the initial ideas did not take hold, were not addressed or merely assumed and instead have evolved into the modern dependency theory of grammar.

4.2.1 Definition of dependency

Tesniere's definition of dependency is not satisfiable. His mentalist approach that "the mind perceives connections between the word and it's neighbours"(Tesniere 2015: 3) makes it impossible to falsify his choices hence leaving no means to validate one choice over the other ones.

One way to define dependency relations and structure is by employing the constituency concept. There are efforts by (Bloomfield 1933; Hockett 1958; Harris 1951) in constituency grammar to identify constituents using tests that shed light on which segments should hold together as phrases or whether they should be considered constituents at all. One needs to decide within each constituent which word it is being headed by which means deciding which word controls the distribution of that constituent (Bloomfield 1933; Zwicky 1985) . A word y depends of a word x if and only if y heads the a phrase which is an immediate constituent of the phrase headed by x (Lecerf 1961).

Another way to define dependencies, avoiding constituency, is by using combinations of two words as proposed by Garde (1977) and Mel'čuk (1988). To discern which governs the other one needs to determine which determined the distribution of the two together. This way the governor is the word that determines the environment in which the two together can appear (Tesniere 2015: lxi). In fact the word notion is not necessary to define dependency, it can be abstracted away to the notion of syntactic units. As soon as two units combine one can posit dependency between them whereby the dependency structure is the set of dependencies between the most granular syntactic units (Gerdes & Kahane 2013).

In addition Tesniere did not make distinctions between the dependency types. As discussed in the previous section, he had noticed that there is a difference between syntactic and semantic dependencies and that the former generally corresponds to the latter but not as a strict rule and even some other times the correspondence is in the opposite direction e.g. "the stone frees" vs. "the frozen stone". The dependency based semantic representations have been around since '60s named *semantic networks* (Žolkovskij & Mel'čuk 1967; Mel'čuk 1988) and *conceptual graphs* (Schank 1969; Sowa 1976).

4.2.2 Grammatical function

In the modern linguistics the notion of grammatical functions e.g. subject, object, determiner etc. are attached to the notion of syntactic dependency. They are in fact an essential account in the modern dependency-based approaches because they are the only way to distinguish between various roles the dependents play in relation to their governors. The grammatical functions attached to the dependency relations are primitives of the dependency grammars. This is not the case for Chomskian phrase structure constituency where the functions are derived from the structural configurations. Nevertheless in latter constituency models such as *Lexical Functional Grammars* (**?**) and *Head-Driven Phrase Structure* (Pollard & Sag 1994) have introduces the grammatical functions as grammatical primitives.

The grammatical functions were not important in Tesniere's theory. He mentioned only, in the context of valency theory, three *actant functions* called *first*, *second* and *third* the other verb dependants being *circumstantial*. Most dependecy grammars assume dozens of functions to offer a fine-grained syntactic characterization of language based on distinguishable syntactic properties. This way two elements have the same grammatical function if and only if they have the same *markers, order* (linear position), *agreement properties* and *distribution*. Several grammatical function sets have been developed in the fields of formal dependency grammars, parsers and tree-banks. The most important ones for English Language are the ones of Mel'čuk & Pertsov (1986), of Johnson & Fillmore (2000) and of Marneffe & Manning (2008a,b). In this work is employed the latter as it is part of the Stanford dependency parser described latter in this Chapter.

4.2.3 Projectivity

Central to how the word order is accounted for in dependency grammar is *projectivity*. It is not present in the Elements but it is the basis for identifying *long-distance dependencies* also known as *discontinuities* or *gapping*. The concept is introduced by Lecerf (1961) following publication of the Elements (Tesniere 1959). It is defined in terms of crossing lines when drawing dependency trees where the ones without containing crossing lines are called *projective* and the ones with crossing lines are called *non-projective* i.e. violating projection principle.

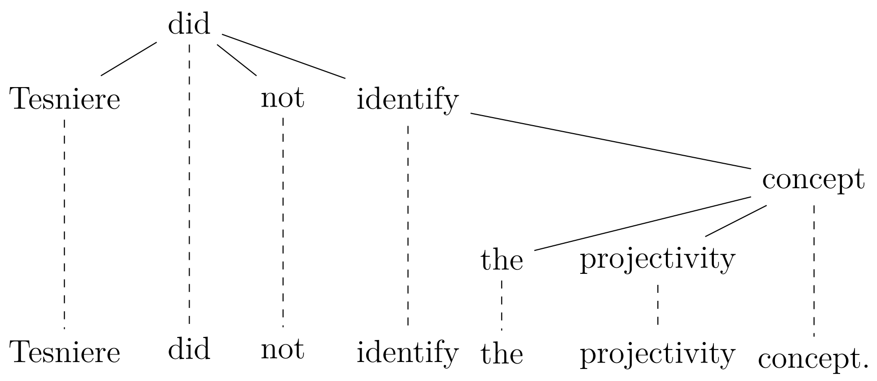


Fig. 4.6 Projective tree

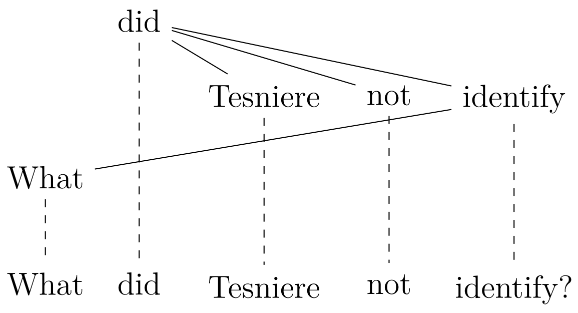


Fig. 4.7 Non-projective tree

To illustrate this principle consider Figure 4.6 where there are no crossing lines whereas Figure 4.7 contains projectivity violation because the word "what" is connected to it's governor "identify" crossing three dashed projection lines. Linguistic phenomena involving non-projecting are: wh-fronting, topicalization, scrambling, and extrapolation.

4.2.4 Function words

Tesniere's transfer theory, despite it's insightfulness, has little if any at all application in modern dependency grammar. The main reason is the implications it has on the hierarchical structure because it does not provide the *translatives* (prepositions, auxiliary verbs, sub-ordinators and conjunctions) with autonomy but a kind of secondary status and thus cannot be constitutive of a nucleus. The issue is reduced to the hierarchical status of such translatives whether they gain node status or not.

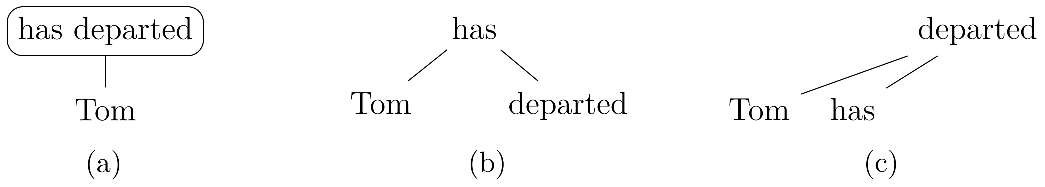


Fig. 4.8 Possible analysis representation for "Tom has departed"

Figure 4.8 represents three possible ways to analyse the word has in "Tom has departed". In Figure 4.8a is represented the original approach Tesniere proposed using transfer schema where the word "has" is enclosed within the full verb node "departed". The two together are granted the status of a dissociated nucleus which means that neither alone can form a nucleus. In contrast the Figure 4.8b and 4.8c the auxiliary has is granted autonomy and corresponds to the modern analysis varying from one model to the other.

As we will see in the next Section, the Stanford dependency schema (Marneffe & Manning 2008a,b) adopts the content words as governors of the function words. This corresponds to the representation in Figure 4.8c. Moreover it provides a collapsed schema where the function words are suffixed to the grammatical functions. For example in "Bob and Jacob" there is a "conj" dependency relation between Jacob and Bob and a "cc" relation between "and" and "Bob". In the collapsed form the relation becomes "conj:and" between between Jacob and Bob integrating the conjunction into the relation name. This is the case for prepositions and conjunctives whereas auxiliary verbs remain nodes in the collapsed form.

4.3 Dependency grammar in automated text processing

Tesniere had no intention in providing a computational theory of grammar and he was neither aware that ideas he was proposing have such potential. Shortly after his death, inspired by Chomsky's Syntactic Structure (Chomsky 1957b), Hays (1960, 1964) makes the first attempts to formalise the dependency grammar with intention to apply it to automated text processing. A year latter his colleague Gaifman (1965) proofs that the *dependency grammar* formalism proposed by Hays is equivalent to Chomsky's *context free grammar* and to *categorial grammars* proposed by Bar-Hillel (1953).

Outshined by Chomskyan grammars, the serious developments in parsing with dependency grammars did not come into being until mid '90s. First efficient parser with the dependency-based model called *Link Grammar* was created by Sleator & Temperley (1995) and ten year latter the dependency parsing gain in popularity yielding remarkable results such as the MaltParser (Nivre 2006; Nivre et al. 2007b), MATE parser (Bohnet 2010) and early Stanford parser (Marneffe et al. 2006) that was generating the dependency trees from phrase structure trees. A summary of dependency parsing techniques is provided by Kübler et al. (2009).

In parallel to parsers, large annotated corpses and treebanks have been developed for parser training and testing and suitable as well for theoretical applications. A treebank is a collection of records consisting of natural language sentences associated with corresponding syntax tree (using a specific grammatical model) and optionally additional annotations such as part of speech tags, named entities, and other annotations. The first treebank was Penn Treebank (Santorini 1990; Marcus et al. 1993) which is a constituency-base treebank. A well known dependency treebank is the Prague Dependency Treebank (Hajic et al. 2001; Böhmová et al. 2003) originally created for Czech but now containing English as well. Recently started an initiative to create a Universal Dependency model (Nivre 2015) and correspondingly with extended efforts was also created a multilingual treebank applying the scheme (Nivre et al. 2016) which continues growing today.

Before arriving to the broadly accepted Universal Dependency, early dependency grammars were quite dispersed. The schemes more often were developed in the context of corpus annotation. An early work (Carroll et al. 1998) towards unification was within the Grammar Evaluation Interest Group (Harrison et al. 1991) also know as *PARSEVAL* initiative that was originally destined for constituency parsers. Carroll et al. (1999) proposed an application independent corpus annotations scheme (see Figure 4.9) specifying the syntactic dependency which holds between each head and its dependent(s) that took into account language phenomena in English, Italian, French and German.

In early 2000 the existing treebanks were still inadequate for evaluating the predicate-argument structure of English clauses. To address this problem, PARC 700 treebank

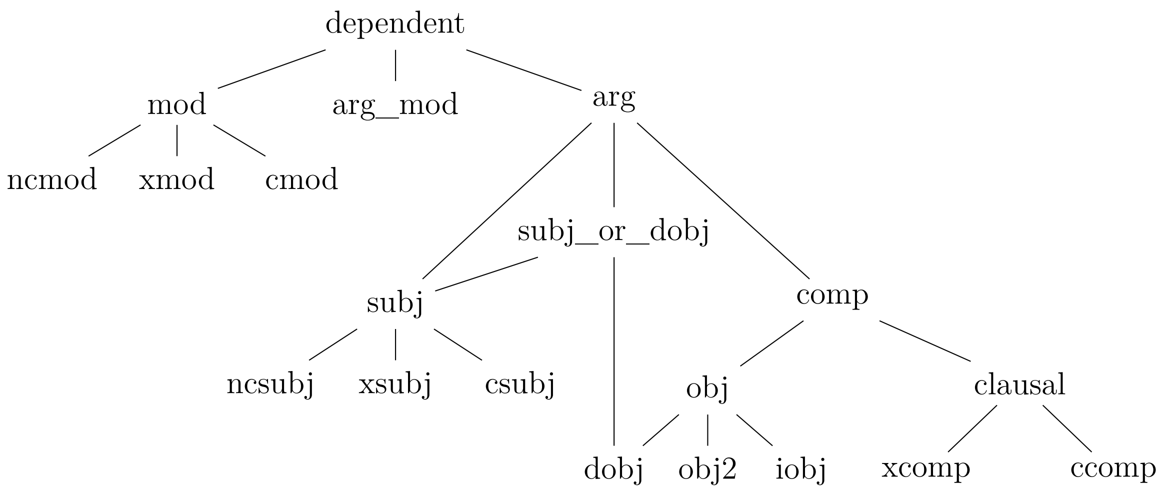


Fig. 4.9 The grammatical relations (GR) hierarchy from Carroll et al. (1999)

(King & Crouch 2003) was created by randomly extracting 700 sentences from Penn treebank, parsed with a Lexical Functional Grammar (LFG), converted into dependency relations and manually corrected by human validators. This scheme has played role in creation of Stanford dependency model that I describe in detail latter.

One advantage of dependency representations is that they can be encoded in a tabular format such as CoNLL (Nivre et al. 2007a) which now is adopted as the standard representation. It is employed in a recurring open competition called "CoNLL shared task" launched for improving and innovating the dependency parsing methods. The most notable are the ones from 2006 on dependency parsing (Buchholz & Marsi 2006) followed in 2007 that included a track for multilingual and one for domain specific dependency parsing. Fast forward to 2017 (Zeman et al. 2017) the task was for parsing from raw text (as previous ones were lemmatised and annotated with part of speech) into universal dependency.

4.4 Stanford dependency model

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

Stanford parser is one of the leaders in the domain of dependency parsing. Since 2006 (Marneffe et al. 2006) for ten years Stanford parser implemented the Stanford dependency model for English (and a few other languages). Then in 2015 Nivre et al. (2016) proposes the language independent Universal Dependency scheme. In this section I present the Stanford dependency model (prior to Universal Dependency) that is used in the current parser.

The design of the Stanford dependency set (Marneffe et al. 2006; Marneffe & Manning 2008a; Marneffe et al. 2014; Silveira et al. 2014) bears a strong intellectual debt to the framework of Lexical Functional Grammars (**?**) from which many relations were adopted. Marneffe et al. (2006) departs from the relation typology described in (Carroll et al. 1999) which was employed in PAREVAL initiative (Harrison et al. 1991) and from the grammatical relations of PARC 700 (King & Crouch 2003) scheme following a style of Lexical Functional Grammar. Marneffe arranges the grammatical relations into a hierarchy rooted in a generic relation *dependent*. This is then classified into a more fine-grained set of relations thet may hold between a head and its dependent following the set of principles (Marneffe & Manning 2008b) stipulated in Generalization 4.4.1.

Generalization 4.4.1 (Design principles for Stanford dependency set)**.**

1. Everything is represented uniformly as binary relation pairs of words.
2. Relations should be semantically contentful and useful to NLP applications.
3. Where possible, relations should use the notions of traditional grammar (Quirk et al. 1985) for easier comprehension by users.
4. To deal with text complexities underspecified relations should be available.
5. When possible content words shall be connected directly, not indirectly mediated by function words (prepositions, conjunctions, auxiliaries, etc.).

When motivating the approach to schema development, Marneffe et al. (2006) insists on practical rather than theoretical concerns proposing that structural configurations be defined as grammatical roles (to be read as grammatical functions)(Marneffe et al. 2006). In the Chomsky tradition **?** the grammatical relations are defined structurally as configurations of phrase structure. Other theories such as Lexical-Functional Grammar reject the adequacy of such an approach (**?**) and advocate a functional representation for syntax at the atomic level. Following the latter approach, she insists that information about functional dependencies between words is very important and shall be explicitly available in the dependency tree.

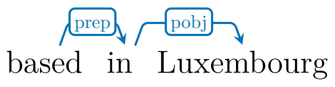
The advantage of explicit relations is that the predicate-argument relations are readily available as edge labels in the dependency structure and can be used off the shelf for real world applications which was an important goal in the schema design.

The grammar had to be suitable for parsing within the context of syntactic pattern learning (Snow et al. 2005), relation extraction, machine translation, question answering and inference rule discovering (Lin & Pantel 2001), domain specific parsing (Clegg & Shepherd 2007), and others. The complete set of dependency relations is Appendix **??**.

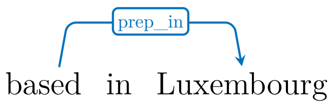
4.5 Stanford dependency representation

The Stanford Dependency Parser generates four types of dependency representations. It produces parse trees with *basic dependencies*, *collapsed dependencies* and *collapsed dependencies with propagation of conjunct* that are not necessarily a tree structure and finally the *collapsed dependencies that preserve a tree structure.* The variant employed in the current work is the collapsed dependencies with propagation of conjunct. This structure concerns preposition, conjunction and relative clause referent nodes, and is generated by a series of transformations after the initial basic dependency parse is ready.

For example, consider fragment "based in Luxembourg". In basic dependency representation, such as is shown in Figure 4.10a, the function words are governing the content words and thus there is a preposition (prep) edge from "based" to a dependent preposition "in" from from which continues a preposition object edge (pobj) to "Luxembourg". In collapsed dependency representation the relation sequences of the type "prep-pobj" are replaced by a direct edge between the two content words labelled with "prep" function concatenated with the intermediary preposition as can be seen in Figure 4.10b. There is a single relation between "based" and "Luxembourg" labelled "prep_in". Similar transformations are done for conjunctions.



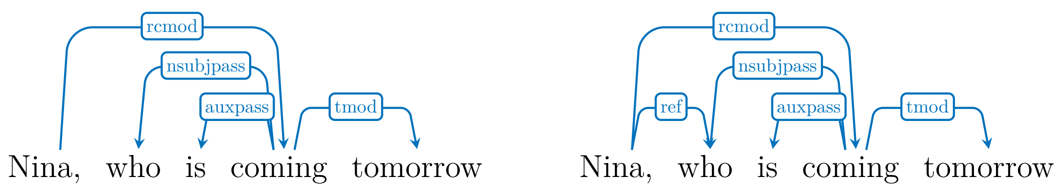
(a) Basic (uncollapsed) preposition dependency



(b) Collapsed preposition dependency

Fig. 4.10 Function words in Stanford dependency model

Besides collapsing prepositions and conjunctions the dependency structure is further processed to introduce more relations even if they break the tree structure. The relative clauses is such a case where the tree structure is broken. Consider Figure 4.11a where the relative clause is introduced by a relative clause modifier relation (rcmod) from the noun "Nina" to the main verb of the relative clause "coming". The clause contains an interrogative pronoun "who" functioning as passive subject (subjpass) and which anaphorically resolves to the clause governor “Nina”. This sort of information about the antecedent of the relative clause is also introduced in the collapsed dependency representation. And thus, as depicted in Figure 4.11b, a new referent relation is added connecting “Nina” to the subordinate subject “who” of the relative clause.



(a) Basic (uncollapsed) relative clause

(b) Collapsed relative clause

Fig. 4.11 Relative clause in Stanford dependency model

There are other language phenomena such as relative clauses that break the tree structure in the collapsed dependency representation by introducing either cycles or nodes with multiple governors. This is the reason why often in this thesis the references ar to dependency graphs and not trees. In fact the fundamental assumption here is that the dependency structures are graphs with a root node. I further develop this assumption in Chapter 6. Nevertheless additional or direct relations between content words (moving accounts of the function words into the graph edges) increase the usability of the dependency graphs for various purposes including the present parse method which is detailed in Chapter 7.

4.6 Cross theoretical bridge from DG to SFG

This section aims at establishing cross theoretical links between Dependency theory of grammar and the Systemic Functional theory of grammar. This cross theoretical bridge is necessary as a fundamental principle for further deriving transformation rules from dependency representation into systemic functional one. Such rules are then enacted in the parsing pipeline for creating the systemic constituency structure which is the aim of this thesis detailed in Chapter 7.

Lets recap what dependency relations are in the Dependecy theory and in Systemic Functional theory of grammar. In the Dependency theory of grammar, as we saw in Section 4.1, the dependency relations are conceptualised as connections between neighbouring words that stand in a governor (superior) and subordinate (inferior) relations to each other, also referred here as *parent-daughter* relations.

In SFL the concept of dependency is less salient than the foundational role it plays in the Dependency theory. They are regarded as orthogonal relations between sibling elements of a unit (Figure 4.13b) and link the *heads* to their *modifiers* in Hallidayan *logical structure*(Halliday & Matthiessen 2013: 388) (discussed in Section 3.4.1). The reason why the elements are siblings and *not* subordinated is due to *componence* based conceptualisation of the unit structure (see Section 3.3.4) as a part-whole linearly ordered set of elements. In SFL, componence, together with filling, embedding and expounding are constituency relations. In this view the subordination is replaced by the componence relation between the element and the unit it is part of and hence making the elements siblings of equal status. Yet one element, the head, plays a special pivotal role to the unit (definer in Section 3.3.2 and discussed in Section 3.4.5) that, in Sydney grammar, is the Head/Thing in Sydney grammar or Head, Apex, Main Verb, etc. in Cardiff grammar.

(44) The witness seemed quite convincing.

Consider the example 44 whose representation as dependency structure is depicted in Figure 4.12 and as Systemic Functional constituency structure in Table 4.1. In Figure 4.12 the structure starts with a root node "seemed" from which span two relations: subject (nsubj) to "witness" and an open clausal complement (xcomp) to "convincing".

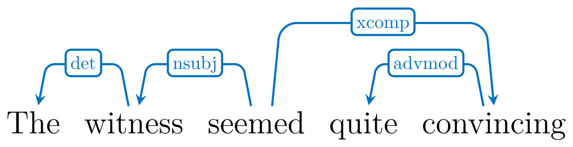


Fig. 4.12 Dependency representation

The	*witness*	*seemed*	*quite*	*convincing*
clause				
Subject		Main Verb	Complement	
nominal group		verb	adjectival group	
Deictic	Head		Temperer	Apex
determiner	noun		adverb	adjective

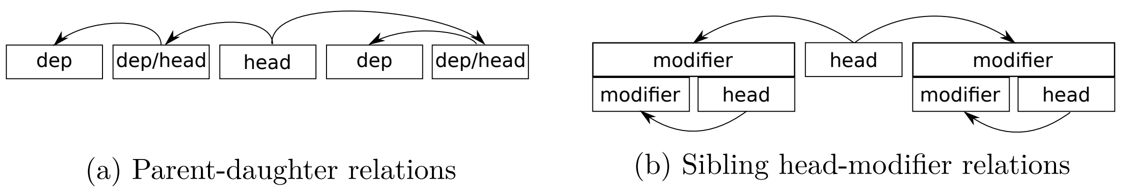
Table 4.1 Example of head-modifier sibling dependency

In Table 4.1, the corresponding structure, is a root clause unit composed of three elements: a Subject a Main Verb and a Complement. The Main Verb is the pivotal element heading the clause unit which means that the inconspicuous dependency relations hold from the Main Verb to the Subject and to the Complement. The Subject and Complement are filled by a nominal and, correspondingly, an adjectival group whereas the Main Verb Element is expounded with a verb item "seemed". This observation is expressed in generic terms but the Generalization 4.6.1.

Next, in Figure 4.12, the determiner relation (det) between "witness" and "the" is of similar as the "subj" holding between a governor and a subordinate. The corresponding constituency structure is that of a nominal group unit with a Head and a Deictic elements. One exception, though, is that the governor also functions as subordinate in another relation and thus has an incoming edge. This dual role of a node has far reaching consequences in the constituency structure. Having an incoming dependency relation corresponds, in constituency structure, to the filling relation between an element of a unit and the unit of the rank right below. Here, the node "witness", acting as a subordinate to the "seemed" node, fills the Subject element of the clause. In fact the dependency node "seemed" projects into constituency structure the expounding relation between the lexical item and the element of the clause.

In a nutshell we see is that the parent-daughter dependency relations in Dependency structure unpacks into multiple relations in the Systemic Functional structure: the componence relation between unit and element, the filling relation between elements and units of the lower rank, the identification of the head of unit element (a.k.a. the pivotal element) and, the (indirect) sibling head-modifier relation.

On the other hand, if we focus on the underlying plain dependency relations between head and dependent we will notice a perfect isomorphism between the two structures. To illustrate that lets reduce the node labels to "head" and "dep" which will correspond, in the dependency representation, to parent (governor) and daughter (subordinate), whereas in constituency representation, to head and modifier siblings.



(a) Parent-daughter relations (b) Sibling head-modifier relations

Fig. 4.13 Plain dependency relations in Dependency and Systemic Functional representation

Figure 4.13 illustrates side by side the parent-daughter and sibling dependency relations in a simplified form. In Figure 4.13a dependencies are the only relations between the units of structure whereas in Figure 4.13b are two levels (ranks) of units where the dependency relations are relevant only between sibling elements at the same level within the structure of a unit. As we have seen in Chapter 3 knowing only the unit elements is not enough to construct the constituency structure, but it is informative enough for deducing the missing parts. What the Figure 4.13 illustrates is that the two structures resemble each other in a suggestive fashion that will be used below to construct a bridge between descriptions.

The intuitions from the above examples can be laid out by and large in Generalizations 4.6.1–4.6.3. Here I use the term *projection* to refer specifically to the correspondences between theoretical primitives of the two grammars. I say that a primitive in in theory A is projected as another primitive in theory B. The first generalisation is on maximally accounting for the dependency nodes in constituency structure.

Generalization 4.6.1 (Structural Completeness)**.** Each node of the dependency representation is projected, in constituency representation, into one or more units and one or more elements at different rank scales.

When translated to a constituency unit, the dependency node, stands for a unit as a whole, the head element of that unit and the word expounding that element. For example, the root verb "seemed" in a dependency graph corresponds to the clause node and, the Main Verb element and the lexical item which fills the Main Verb of the clause. By analogy, the node "witness" stands for the nominal group, the head noun of a Nominal Group and fills the head element of the group. Even the functional words such as prepositions that in collapsed dependency representation remain orphaned (see Figure 4.10b) have to be accounted in the constituency structure.

Generalization 4.6.2 (Functional Projection)**.** Each dependency relation (alone or contextualised by the word classes of the bridged nodes) is projected into the element of the unit corresponding to the subordinate node.

The dependency relation is primarily responsible for determining the element. For example the "nsubj" dependency relation will be projected into a Subject element of a clause unit. Sometimes however, as stated in Generalisation 4.6.2, the dependency relation alone is not enough and the context given by the governor and subordinate nodes is needed to choose the element. For example the adverbial modifier (advmod) relation alone is not enough to determine into which element to project the subordinate. If the word class context is considered then the verb-to-adverb "advmod" relation (VB-advmod-RB) is projected into an Adjunct while the noun-to-adverb "advmod" relation (NN-advmod-RB) into is projected Predeictic.

Generalization 4.6.3 (Substantial Projection)**.** Each dependency node projects either the filling or expounding of an element, by a unit of the rank below or by the lexical item of the node.

Generalisation 4.6.3 provides the link between the projected unit with the element of the rank above. For example the subordinate "witness" receiving nominal subject edge from "seemed" is responsible for projecting that the Subject element of the clause if filled by the nominal group (headed by the "witness" because it is also the governor in the next relation that projects it is the head). The governor "seemed", as the root of the tree, is projected into a lexical item which expounds the Main Verb Element (the head of the clause group). In a similar manner, the governor "witness" of the determiner (det) relation to "the" is projected into a lexical item expounding the Head element (of the nominal group).

some	*very*	*small*	*wooden*	*ones*
nominal group				
Quantifying Determiner	Epithet		Classifier	Head
	quality group			
	Temperer	Apex		

Table 4.2 SF analysis of Example 20 (reproduced from Table 3.5)

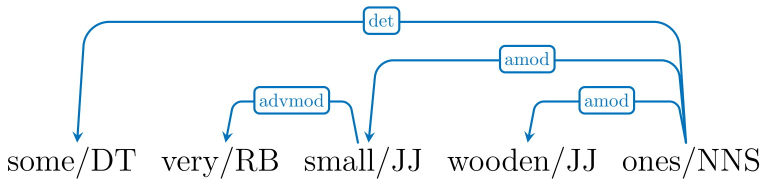


Fig. 4.14 Dependency analysis for Table 4.2

Figure 4.14 and Table 4.2 represent the analysis of a nominal group from Example 20 ("some small very small wooden ones") in the SFG and the Stanford dependency grammar contrasting the two structures. Consider the dependency relation "det" acting as a link between the noun "ones" and the determiner "some". When translated into the SF variant the dependency relation stands within the nominal group between the Head element (filled by word "ones") and the Quantifying Determiner element (filled by the word "some"). As mentioned earlier all the elements in a unit are equal in the structure so the Head and Quantifying Determiner are siblings. So the items (words) filling those elements are also siblings. How then is the dependency relation established?

Lets look at a second example of the two relations “advmod” from “small” to “very” and “amod” from “ones” to “small” in Figure 4.14. The interesting case here is the item “small” which is the head (Apex) of the quality group. I it anchors the meaning of the whole group and the quality group fills the Modifier/Epithet element within the nominal group. What is not covered in previous example is that the Apex “small” not only is a representative of the entire group but it also suitable filler on its own for the Modifier element within nominal group. Using the similar translation mechanism as above, this means that, the incoming dependency needs to be unpacked into three levels: the element within the current group (Modifier), the unit class that element is filling (Quality Group) and finally the head of the filler group (Apex). In fact, to be absolutely correct there is one more level. The elements of a unit are expounded by lexical items, so a fourth relation to unpack is the expounding of the Apex by the word “small”.

I just showed how the dependency relation in dependency structure (Figure 4.13a) can be unpacked into compounding elements of a unit (Figure 4.13b) corresponding to the the sibling dependency considered an indirect relation between the Head and the Modifier (in the Logical metafunction); and then from that using Generalisations 4.6.1–4.6.3 deduce the rest of the constituency structure such as the componence relation between unit head and the compounding elements and the filling/expounding relation between the element and the unit right below.

In practice to achieve this level of unpacking two traversals are needed a bottom-up and a top one. This is because the traversal sequence also creates a context that deals with either instantiation a new unit and establishing its elements or with filling an element of a unit above. But more on how to enact the cross theoretical links from this chapter is provided in Section 7.4.

As motivated elsewhere, I present an account for the unrealised, covert (Null) elements in syntactic structure, using the Governance and Binding Theory. It is also an opportunity to perform a similar cross-theoretical projection exercise.

Chapter 5

Government and binding theory

Transitivity analysis in SFL is similar to what *semantic role labelling*, *thematic* or *θ role analysis* means in other theories. This thesis provides, in Chapter 8, an account of how to perform SF Transitivity parsing resulting in a configuration of a process, participants and circumstances. For an illustration take Example 45 whose Transitivity analysis is available in Table 5.1. Here the entire clause is analysed as a Possessive configuration governed by the verb "receive" where "Albert" plays the *role* of the Affected-Carrier and "a phone call" is the thing being Possessed.

(45) Albert received a phone call.

(46) He asked to go home immediately.

Albert	*received*	*a*	*phone*	*call*
Possessive configuration				
Affected Carrier	Process	Posessed		

Table 5.1 Transitivity analysis with Cardiff grammar of Example 45

Example 46 is slightly more complex and illustrates the main motivation behind the current chapter. It is analysed in Table 5.2, according to Cardiff grammar, as a Three Role Cognition configuration with "ask" being the process, "he" the Agent and "to go home immediately" the cognised Phenomenon. The Phenomenon is filled by a non-finite clause "to go home immediately" which is, in Transitivity account, a Directional configuration governed by the verb "go" and has as participants the Destination "home" and the Agent Carrier in an empty Subject position that is said to be *non-realised*, *empty*, or *covert*. This is a case when the empty constituent is recoverable from the clause above and corresponds to the Subject "He". This way, the constituent "He" plays two roles: first as Agent in the Cognition process of the top clause and second as Agent Carrier in the Directional process of the embedded clause. In this work, the way to assign a second role coming from the lower clause, is by detecting and making explicit the empty constituents and resolving them locally with a link to the corresponding constituent.

He	*asked*	*[empty subject]*	*to*	*go*	*home*	*immediately*
Three Role Cognition configuration						
Agent	Process	Phenomena				
		Directional Configuration				
		Agent-Carrier		Process	Destination	

Table 5.2 Transitivity analysis with Cardiff grammar of Example 46

In language there are many cases where constituents are empty but recoverable from the immediate vicinity relying in most cases on syntactic means and in a few cases additional lexical-semantic resources are required. The mechanisms of detecting and resolving the empty constituents are captured in the Government and Binding Theory (GBT) developed in (Chomsky 1981, 1982, 1986) and based on the phrase structure grammar. GBT explains how some constituents can *move* from one place to another, where are the places of *non-overt constituents* and what constituents do they refer to i.e. what are their *antecedents*.

The GBT approach explains grammatical phenomena using *phrase structures* (PS). This is more distant from SFG than the approach taken by the dependency grammar. Section 5.2 briefly introduces the theoretical context of GBT and then formulates the principles and generalisations relevant for current work. Then Section 5.3 translates the introduced principles and generalisations into Dependency Grammar rules and patterns. To lay the ground for the two sections, I first place GBT into the context of transformational grammar and introduce the basic concepts.

5.1 Introduction to GBT

This section is set as introduction to the fundamental concepts from Government and Binding Theory. It belongs to the family of Transformational grammars (TG) or transformational-generative grammars (TGG). It is part of the theory of generative grammar that considers grammar to be a system of rules that generate exactly those combinations of words which form grammatical sentences in a given language (Chomsky

1965). TG involves the use of defined operations called transformations to produce new sentences from existing ones.

Chomsky developed a formal theory of grammar (Chomsky 1956) where transformations manipulated not just the surface strings, but the parse tree associated with them, making transformational grammar a system of tree automata (Stockwell et al. 1973).

A transformational-generative (or simply transformational) grammar thus involved two types of productive rules: *phrase structure rules*, such as “S -> NP VP” (meaning that a sentence may consist of a noun phrase followed by a verb phrase) etc., which could be used to generate grammatical sentences with associated parse trees (phrase markers, or P markers); and *transformational rules*, such as rules for converting statements to questions or active to passive voice, which acted on the phrase markers to produce further grammatically correct sentences (Bach 1966: 59-66). This notion of transformation proved adequate for subsequent versions including the “extended”, “revised extended” and Government-Binding (GB) versions of generative grammar, but may no longer be sufficient for the latest “minimalist” grammar (Chomsky 1993a). It require a formal definition that goes beyond the tree manipulation. For the purpose of the current work, however, the GBT employing the idea of transformations is perfectly suitable. I selected it because of clear and extensive descriptions of the mechanisms for identification of *null elements* (also known as *empty categories*) and how to provide them with an interpretation.

5.1.1 Phrase structure

The notion of structure in a generative grammar refers to the way words are combined together to form phrases and sentences. *Merging* is the technical term used in GBT for the operation of bringing two words together into a phrase. In this operation one word will always be more prominent and is therefore called the *head* of the phrase. The resulting combination is a new constituent and is called a *projection* of the head. This is known as *X-bar theory* (often denoted as X’ or $\bar{X}$) and embodies two primary claims: (a) that the phrases may contain intermediary constituents projected from a head X and (b) that this system of projected constituency may be common to more than one category (such as N, V, A, P etc.).

These combinations of words and projections can be represented using the *labelled bracketing notation* where the labels denote constituent categories. The bracketed notation is a representation equivalent to a hierarchical tree of constituent parts or *parse tree* (also known as *syntactic tree*, *phrase structure*, *derivation tree*). The parse tree represents the syntactic structure of a string according to some grammar. The equivalence between a bracketed notation and parse tree is exemplified in the following two representations of 47 from (Haegeman 1991: 83).

(47) Poirot will abandon the investigation.

(48) $\left[_S \left[_{NP} \left[_N Poirot\right]\right] \left[_{AUX} will\right] \left[_{VP} \left[_V abandon\right] \left[_{NP} \left[_{Det} the\right] \left[_N investigation\right]\right]\right]\right]$

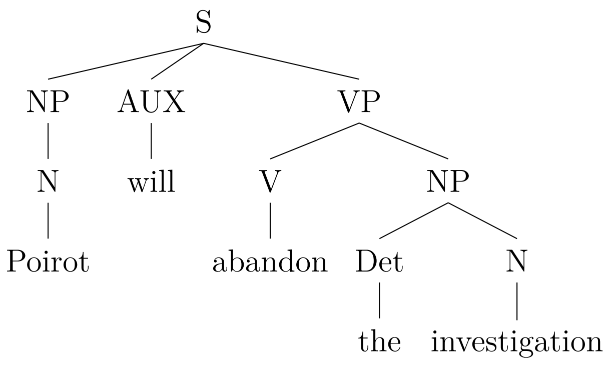


Fig. 5.1 The parse tree of Example 47 from (Haegeman 1991: 83)

A node is said to be *non-branching* if there is a single line starting below and it is called *branching* if there are more than one line going downwards. The children of a branching node are said to be bound by a *sisterhood* relation and in relation to a *parent* or *mother* node. In a phrase structure the vertical relations are referred as *dominance* relations defined below.

Definition 5.1.1 (Dominance)**.** Node A dominates node B if and only if A is higher up in the tree than B and if you can trace a line from A to B going only downwards (Haegeman 1991: 85).

Looking at the tree diagram along the horizontal axis, GBT describes left-to-right ordering of constituents using the *precedence* relation.

Definition 5.1.2 (Precedence)**.** Node A precedes node B if and only if A is to the left of B and neither A dominates B nor B dominates A (Haegeman 1991: 85).

In Figure 5.1 NP, AUX and VP nodes are sisters, they precede each other and are dominated by S parent node. A more specific type of dominance, that will be employed latter in this chapter, is the *immediate dominance* which is when there is no intermediary node between A and B. In this case, the node “Poirot” is also dominated by S but only the grandparent NP is immediately dominated by S. The same holds for precedence: the *immediate precedence* is when a node A precedes a node B and there is no intervening node in between. Node NP precedes VP but only AUX is immediately preceded.

Generalization 5.1.1 (Projection principle)**.** Lexical information is syntactically represented.

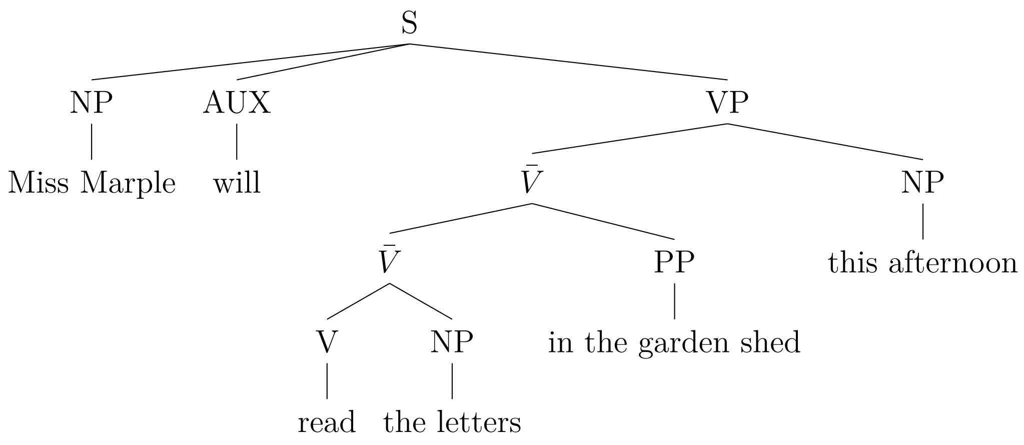


Fig. 5.2 Example of projections from (Haegeman 1991: 90)

An important principle in GBT is that of *projection* formulated in Generalisation 5.1.1. For example in Figure 5.2, projections of V that are dominated by more comprehensive projections of V are called *intermediate projections* while the node labelled VP is the *maximal projection* of V. Maximal projections are also barrier to government (see Definition 5.1.5 below). The role of lexicon in syntax from to GBT perspective is discussed at large in Stowell & Wehrli (1992).

5.1.2 Theta theory

This section introduces which constituents are minimally required to form a sentence and why. Traditionally three types of verbs are recognised: *transitive*, *di-transitive* and *intransitive*. This distinction is based on how many complements a verb requires to form a minimal complete sentence. If a verb is transitive then one NP direct object is required. If the verb is di-transitive then two NP or one NP and one PP direct and indirect objects are required. Finally, if it is intransitive then no NP complement is allowed.

Logicians for a long time have been concerned with formulating representations corresponding to semantic structure of sentences or *propositions*. Like Tesniere (Tesniere 2015: 97) discussed in Section 4.1, Haegeman employs the metaphor of a theatre play when discussing the argument structure of predicates. A play not only describes the number of participants but also what corresponding roles they play. The specific semantic relations between the verb and its arguments is comparable with the identification of characters in a play script (Haegeman 1991: 49).

In logical notation such as in Example 49 a proposition comprises of a predicate (P) that takes a certain number of *arguments* (here a and b). By analogy to the logical tradition, in GBT, the verb is said to be like the predicate while the subject together with complements are like the arguments that the predicate requires.

In Example 50 Maigret, taking subject position, is the Agent in the process of killing while Poirot in the complement position is the Patient that receives the effects of the process of killing. The generic argument structure for the verb "to kill" can be expressed as in Example 51. The first argument is of NP category and takes the role of an Agent while the second argument is also an NP but it takes the Patient role. The transitivity of a verb dictates how many arguments there should be.

(49) P(a, b)

(50) Maigret killed Poirot.

(51) V kill: 1 (NP:Agent), 2 (NP:Patient)

In literature these relations between the verb and the arguments are called *thematic roles* or theta-roles (θ-roles). It is said that the verb *theta marks* its arguments. The component of the grammar that regulates the assignment of thematic roles is called *theta theory*.

In GBT the theory of thematic roles is very sketchy and does not go beyond distinction of several thematic roles (Agent/Actor, Patient, Theme, Experiencer, Beneficiary, Goal, Source, Location and the controversial Theme) (Haegeman 1991: 50). The theta theory has a central criterion that is stipulated in Generalisation 5.1.2.

Generalization 5.1.2 (Theta criterion)**.** Theta criterion requires that:

- each argument is associated one and only one theta role
- each theta role is assigned to one and only one argument (Haegeman 1991: 54)

(52) *It* surprised Jeeves that the pig had been stolen.

In English, however, there is a special case, that of *expletives*, when the subject argument is filled by the pronoun *it* that receives no thematic role and acts rather as a dummy slot filler without any semantic contribution to the meaning of the sentence (Haegeman 1991: 62). Worth noticing is also the fact that *auxiliary verbs* and *copula verbs* do not assign thematic roles (Pollock 1989).

The verb that assigns a theta role does not need to specify which syntactic category it shall be realised by. In more technical it means that the categorial selection (*c-selection*) follows from semantic relation (*s-selection*). When a theta role can be assigned to an argument it is said that it is saturated. In order to identify the assignment of respective role the arguments are identified by the means of an index provided as subscript in the sentence.

(53) Maigret$_i$ killed the burglar$_j$.

(54) Maigret$_i$ said that he$_i$ was ill.

In the Example 53 Maigret has the index i and the burglar j meaning they are distinct referents. On contrary, in Example 54, "he" receives the same index as Maigret because they are interpreted as referring to the same entity. We say that the two are *coindexed*.

5.1.3 Government and Binding

Using the terminology from the traditional grammar it is said that a verb governs its object. This is generalised in GBT as a rule that the head of a phrase, called *governor*, *governs* its complement, called the *governee*. This relation is loosely defined in Definition 5.1.3 below and formally in Definition 5.1.5.

Definition 5.1.3 (government i)**.** A governs B if

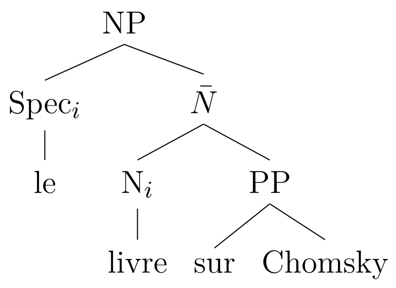
- A is a governor;
- A and B are sisters

Governors are heads (Haegeman 1991: 86).

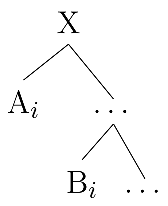
In Figure 5.1 the verb "abandon" is the head of the verb phrase (VP) and governs the direct object - nominal phrase (NP) "the investigation". V does not govern the subject NP "Poirot". All the constituents governed by a node constitute the *governing domain* of that node. In this case VP is the governing domain of V.

Before providing the next definition of government, I first introduce the notion of C-command which provides a general pattern of how the agreeing elements relate to each other in the parse tree. C-command is formally defined in Definition 5.1.4.

When considering the geometrical relation between the agreeing elements, one always is higher in the tree than the other one in a manner depicted in Figure 5.3b. In Figure 5.3a the co-subscripted nodes indicate agreement. Here the [Spec, NP] c-commands all the nodes dominated by the NP. These nodes constitute the *c-command domain* of Spec element (Haegeman 1991: 134).



(a) Example of agreement in French (Haegeman 1991: 132)



(b) Schematic representation of the c-command (Haegeman 1991: 133)

Fig. 5.3 Agreement example and schematic representation

Definition 5.1.4 (c-command)**.** A node A c-commands a node B if and only if

- A does not dominate B;
- B does not dominate A;
- the first branching node dominating A also dominates B (Haegeman 1991: 212).

Definition 5.1.5 (Government)**.** X governs Y if and only if

- X is either of the category A, N, V, P, I;
  or
  X and Y are coindexed
- X c-commands Y;
- no barrier intervenes between X and Y;
- there is no Z such that Z satisfies the points above and X c-commands Z (Haegeman 1991: 557).

IN GBT three types of NP are distinguished: *full noun phrases* (e.g. Maigret, the doctor, etc.), *pronouns* (e.g. he, me, us, etc.), and *anaphors* comprised of reflexives (e.g. myself, herself, etc.) plus reciprocals (e.g. each other, one another,). Pronouns and anaphors (reflexives and referential) lack inherent reference. Anaphors need an antecedent for their interpretations whereas pronouns do not. Pronoun indicate some inherent features of the referent so that they can be identified from the contextual information. The full noun phrases called Referential expressions or *R-expression*, for short, are inherently referential and do not need an antecedent, moreover they do not tolerate an antecedent (Haegeman 1991: 226). The NP types can be defined in terms of features Anaphor and Pronominal (systematised together with the empty categories in the Table 5.3 below). This way the Pronouns have features [+Pronominal,-Anaphor], the Anaphors [+Pronominal,-Anaphor] and the R-expressions [-Pronominal,-Anaphor]. The last combination [+Pronominal,+Anaphor] corresponds to *PRO empty category* which will be discussed in the Section 5.2.1 coming up next.

The module of the grammar regulating interpretation of the noun phrase (NP) interpretation is referred, in GBT, as the *binding theory*. It is formally defined in terms of c-command in Definition 5.1.6. And because the BT is essentially concerned with binding of NPs in argument positions (*A-position*) then it is rather the **A-binding** (see Definition 5.1.7) of interest here. An A-position is a position in the tree to which a theta role can (but not necessarily) be assigned (Haegeman 1991: 115).

Definition 5.1.6 (Binding)**.** A binds B if and only if

- A c-commands B;
- A and B are coindexed (Haegeman 1991: 212).

Definition 5.1.7 (A-Binding)**.** A A-binds B if and only if

- A is in A-position;
- A c-commands B;
- A and B are coindexed (Haegeman 1991: 240).

Each of these NP types have an associated binding principle (ways in which to interpret, if needed, the reference of the NP) provided in Generalisations 5.1.3, 5.1.4 and 5.1.5 below. These principle use the idea of *governing category* which for a node A is the minimal domain containing it, its governor and an accessible subject. A subject A is said to be accessible for B if the co-indexation of A and B does not violate any grammatical principle (Haegeman 1991: 241).

Generalization 5.1.3 (Principle A of binding theory)**.** An anaphor (i.e. a NP with the feature [+Anaphor] covering reflexives and reciprocals) must be bound in its governing category (Haegeman 1991: 224).

Generalization 5.1.4 (Principle B of binding theory)**.** The pronoun (i.e. a NP with feature [+Pronominal]) must be free in its governing category (Haegeman 1991: 225).

Generalization 5.1.5 (Principle C of binding theory)**.** An R-expression (i.e. a NP with independent reference) must be free everywhere (Haegeman 1991: 227).

5.2 On Null Elements

In certain schools of linguistics, in the study of syntax, an *empty category* is a nominal element that does not have any phonological content and is therefore unpronounced. Empty categories may also be referred to as *covert nouns*, in contrast to overt nouns which are pronounced (Chomsky 1993b). Some empty categories are governed by the *empty category principle* (see Definition 5.2.1). When representing empty categories in trees, linguists use a null symbol to depict the idea that there is a mental category at the level being represented, even if the word(s) are being left out of overt speech.

GBT recognises four main types of empty categories: *NP-trace*, *Wh-trace*, *PRO*, and *pro*. They are subject to Principles A, B and C of the binding theory provided above and differentiated, like the over NPs, by two binding features: the anaphoric feature [a] and the pronominal feature [p]. The four possible combinations of plus (+) or minus (-) values for these features yield four types of empty categories.

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

Table 5.3 Four types of empty categories (adaptation from (Haegeman 1991: 436))

In Table 5.3, [+a] refers to the anaphoric feature, meaning that the particular element must be bound within its governing category whereas [+p] refers to the pronominal feature which shows that the empty category is taking the place of an overt pronoun.

Definition 5.2.1 (Empty Category Principle (ECP))**.**

- Traces must eb properly governed.
- A properly governs B if and only if A theta-governs B or A antecedent-governs B (Chomsky 1986: 17).

- A theta-governs B if and only if A governs B and A theta-marks B.
- A antecedent-governs B if and only if A governs B and A is coindexed with B (Haegeman 1991: 442).

Next I describe in detail each empty category and the properties of corresponding overt noun type.

5.2.1 PRO Subjects and control theory

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

Definition 5.2.2 (Control)**.** Control is a term used to refer to a relation of referential dependency between an unexpressed subject (the control element) and an expressed or unexpressed constituent (controller). The referential properties of the controlled element are determined by those of the controller (Bresnan 1982).

Control can be *optional* or *obligatory*. While *Obligatory control* has a single interpretation, that of PRO being bound to its controller, the *optional control* allows for two interpretations: *bound* or *free*. In Example 55 the PRO is controlled, thus bound, by the subject "John" of the matrix clause (i.e. higher clause) whereas in 56 it is an arbitrary interpretation where PRO refers to "oneself" or "himself". In 57 and 58 PRO must be controlled by the subject of the higher clause and does not allow for the arbitrary interpretation.

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

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

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

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

Sometimes the controller is the subject (as in Examples 55, 56, 57) and sometimes it is the object (Example 58) of the higher clause. Haegeman (1991: 278) proposes that there are two types of verbs, verbs of *subject* and of *object control*. The following set of generalizations from Haegeman (1991) are instrumental in identifying places where a PRO constituent can be said to occur and identifies its corresponding binding element.

Generalization 5.2.1. Each clause has a subject. If a clause doesn't have an overt subject then it is covertly (non-overtly) represented as PRO (Haegeman 1991: 263).

Generalization 5.2.2. A PRO subject can be bound, i.e. it takes a specific referent or can be arbitrary (equivalent to pronoun "one") (Haegeman 1991: 263). In case of obligatory control, a PRO subject is bound to a NP and must be c-commanded by its controller (Haegeman 1991: 278).

Generalization 5.2.3. PRO must be in ungoverned position. This means that (a) PRO does not occur in object position (b) PRO cannot be subject of a finite clause (Haegeman 1991: 279).

Generalization 5.2.4. PRO does not occur in the non-finite clauses introduced by *if* and *for* complementizers, but it can occur in those introduced by *whether* (Haegeman 1991: 279).

Examples 59 and 60 illustrate Generalisation 5.2.4

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

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

Generalization 5.2.5. PRO can be subject of complement, subject and adjunct clauses (Haegeman 1991: 278).

Generalization 5.2.6. When PRO is the subject of a declarative complement clause it must be controlled by an NP, i.e. arbitrary interpretation is excluded (Haegeman 1991: 280).

Generalization 5.2.7. The object of active clause becomes subject when it is passivized and also controls the PRO element in complement clause (Haegeman 1991: 281).

Generalization 5.2.8. PRO is obligatorily controlled in adjunct clauses that are not introduced by a marker (Haegeman 1991: 283).

Adjuncts (clauses or phrases) often are introduced via prepositions. Nonetheless there are rare cases of adjunct clauses free of preposition. Examples 61 and 62 illustrate such marker-free adjunct clauses.

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

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

Generalization 5.2.9. PRO in a subject clause is optionally controlled; thus by default it takes arbitrary interpretation (Haegeman 1991: 283).

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

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

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

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

A default assumption is to assign arbitrary "one" interpretation to each PRO subject in subject clauses. However, there are cases when it may be bound (resolved) to a pronominal NP in the complement of the higher clause. The binding element can be either the entire complement or a *pronominal* part of it like the qualifier or the possessor. Example 63 illustrates that PRO has only arbitrary interpretation since it cannot be bound to the complement "health". Moreover PRO can also be bound to (a) the possessive element of a higher clause - example 64, (b) the complement of the higher clause - example 65 and (c) either the possessives in lower or higher clause, Example 66.

5.2.2 NP-traces

In GBT, *movement* is a kind of transformation used to explain discontinuity or displacement phenomena in language. It is based on the idea that some constituents appear to have been displaced from the position where they receive important features of interpretation.

GBT distinguishes three types of movement: (a) *head-movement* - the movement of auxiliaries from I to C , *Wh-movement* - when the wh-constituent lands in Spec position of a CP (i.e. [Spec, CP]) and (c) NP-movement when a NP is moved into an empty subject position. NP-movement in GB theory is used to explain *passivization*, *subject movement* (in interrogatives) and *raising*. The raising phenomenon (Definition 5.2.4) is the one that is of interest for us here as it is the one involving an empty constituent.

When an NP moves it is said to leave *traces* (Definition 5.2.3). The moved constituent is called *antecedent* of a trace. Both the trace(s) and the antecedent are coindexed and form what is called a *chain* (Haegeman 1991: 309).

Definition 5.2.3 (Trace). A trace is an empty category which encodes the base position of a moved constituent and is indicated as t (Haegeman 1991: 309).

Definition 5.2.4 (NP-raising). NP-raising is the NP-movement of a subject of a lower clause into subject position of a higher clause (Haegeman 1991: 306).

Consider Examples 67 to 70 where I used square brackets to indicate boundaries of an embedded clause. There are two cases of expletives (67 and 69) and their non-expletive counterparts (68 and 70) where the subject of the lower clause is moved to the subject position of the matrix clause by replacing the expletive. The movement of NPs are described in GB as leaving traces which here are marked as *t*. This phenomena is called *raising* (Definition 5.2.4) or as Postal (1974) calls it the *subject-to-subject raising*.

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

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

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

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

The subjects "It" and "Poirot" in none of the examples 67–70 receive a semantic role from the main clause. "It" is an expletive and never receives a thematic role while "Poirot" in 68 and 70 takes an Agent role from "destroy" and is not the Experiencer neither of "believe" nor of "seem". So verbs "believe" and "seem" do not theta mark their subjects in these examples.

Raising is very similar to obligatory subject control with a difference in thematic role distribution. In the case of subject control, both the PRO element and it's binder (the subject of higher clause) receive thematic roles in both clauses. However in the case of raising, the NP is moved and it leaves a trace which is theta marked but not to the antecedent (Haegeman 1991: 314). This is expressed in generalization 5.2.10.

Generalization 5.2.10. The landing site for a moved NP is an empty A-position. The chain formed by a NP-movement is assigned only one theta role and it is assigned on the foot of the chain, i.e. the lowest trace (Haegeman 1991: 314).

So, in case of raising, the landing sites for a moved NP are empty subject positions or the ones for expletives. As a result of movement, these positions are filled or expletives are replaced with the moved NP. The last type of movement is the one of NPs that have a *wh-element* described in the next section.

5.2.3 Wh-traces

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

(71) [What] will Poirot eat?

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

(73) [Whose pigs] must Wooster feed?

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

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

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

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

Haegeman (1991: 375) offers the Examples 71 – 77 of *Wh-constituents* which are any NPs or PPs that contain a *Wh-word* in their componence. Thus the wh-constituent can be a single word or a *Wh-phrase*. The Wh-phrase is then the NP or PP which is the maximal projection from a Wh-word. She treats each Wh-word as the head of the Wh-phrase and as we will see in Section 5.3.3 below, I provide a different definition of Wh-group such that it is congruent with the Systemic Functional Grammars.

In GBT the *case* occupies an important place in the grammar. The rules governing the case are known as *case theory*. Verbs dictate the case of their arguments, a property called *case marking*. The Subjects are marked with Nominative case while the Complements of the verb are marked with Accusative case.

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

Example 78 shows that the Accusative (*whom*) is disallowed when its trace is in Subject position since this requires Nominative (*who*). In example 79 the reverse holds and the Nominative form is disallowed as the Wh-element moved from Complement position requires Accusative case.

(78) Who$_i$/*Whom$_i$ do you think [t$_i$ will arrive first?]

(79) Whom$_i$/*Who$_i$ do you think [Lord Emsworth will invite t$_i$?]

Another important distinction to be made among English Wh-elements is the *theta-marking*, i.e. the argument and non-argument distinction. Some wh-constituents will be in A-positions (i.e. functioning as subject or complement) such as in Example 71 – 73 or in non A-position (i.e. functioning as adjunct) such as in Example **??** and 76.

When the Wh-constituent moves, There are two places where it can land: (a) either in the subject position of the matrix clause changing its mood to interrogative (example

80) or (b) subject position of the embedded clause creating embedded questions (example 81). However regardless of the landing site, the movement principle is subject to what Haegeman describes as *that-trace filter* expressed in Generalisation 5.2.11 and the *Subjacency condition* (Generalisation 5.2.12). Note that the matrix or embedded clauses correspond to the category of inflectional phrase (IP).

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

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

Generalization 5.2.11 (That-trace filter)**.** The sequence of an overt complementizer "that" followed by a trace is ungrammatical (Haegeman 1991: 399).

The examples in 82 to 85 provided by Haegeman (1991: 398) illustrate how the above generalization applies.

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

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

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

(85) Who do you think will arrive first?

Generalization 5.2.12 (Subjacency condition)**.** Movement cannot cross more than one bounding node, where bounding nodes are IP and NP (Haegeman 1991: 402).

The Subjacency condition captures the grammaticality of NP-movement and exposes two properties of the movement, namely as being *successive* and *cyclic*. Consider the chain creation resulting from Wh-movement in Examples 86 – 88 provided by Haegeman (1991: 403–406). The Wh-movement leaves (intermediate) traces successively jumping each bounding node.

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

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

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

What Generalisation 5.2.12 states is that a Wh-constituent cannot move further than subject position of the clause forming an interrogative form. Or also it can move outside into the subject position of the clause higher above leaving a Wh-trace as can be seen in the Example 87 and 88.

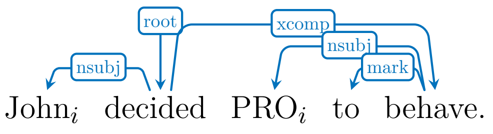
Now that the kinds of null elements have been concisely described laying out the main rules governing their behaviour, I turn next to discuss what how these elements can be identified in terms of Dependency grammar.

5.3 Placing Null Elements into the Stanford dependency grammar

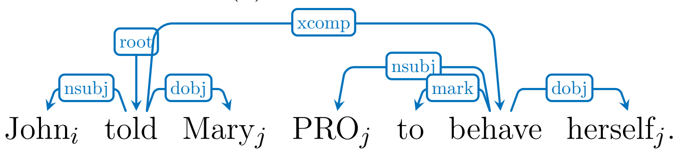
This section provides a selective translation of principles, rules and generalisations captured in GB theory into the context of dependency grammar. The selections mainly address the identification of places where (and by which relations) the null elements should be injected into dependency structure that will later help the semantic parsing process described in Chapter 8.

5.3.1 PRO subject

Coming back to definition of PRO element in Section 5.2.1, it is strictly framed by the non-finite subordinate clauses. In dependency grammar the non-finite complement clauses are typically linked to their parent via *xcomp* relation which is defined in Marneffe & Manning (2008a) as introducing an open clausal complement of a VP or ADJP without its own subject, whose reference is determined by an external reference as can be seen in Figure 5.4.



(a)



(b)

Fig. 5.4 Dependency structure with a PRO subject

These complements are always non-finite. Following the principles stated in Generalisation 5.2.1 and 5.2.3 the non-finite complement clause introduced by *xcomp* relation would receive by default a PRO subject (controlled or arbitrary).

The markers (conjunctions, prepositions or Wh-elements) at the beginning of the embedded clause are no longer connected via *xcomp* relation but instead via either *prepc*, *rcmod*, *partmod* and *infmod* and a slight variation in clause features and constituency. Those cases are no longer treated under the PRO null element considerations and will be discussed later in this chapter since they correspond to other types of empty elements. The only exception, however, is the *prepc* relation only with the preposition "whether" which also introduces a complement clause with PRO element.

I have explained earlier how to identify the place where a PRO element should be created. Before creating it we need to find out (1) whether PRO is arbitrary (equivalent to pronoun "one") or it is bound to another constituent. And if it is bound then decide (2) whether it is bound to (and coindexed with) subject or object (case in which we say that PRO is subject or object controlled) as can be seen in Figure 5.4.

Generalisation 5.2.6 above introduced test whether the complement clause is interrogative or declarative (which resembles mood determination in SFG). Many grammars, including SFG, do not consider that the non-finite clauses can have interrogative/declarative variation (called by Halliday & Matthiessen (2004: 107-167) *mood* feature). Nonetheless, in GBT, even if a clause is non-finite such a distinction is useful. The complement clauses can have structural variation resembling a declarative or interrogative mood for the reason that a complement clause can start with a Wh-constituent which turns it into an interrogative one. Thus the test is whether there is a Wh-marker (who, whom, why, when, how) or the preposition "whether". The presence of any marker like in example 89 at the beginning of the complement clause will change the dependency relation from *xcomp* to another one and sometimes the structure of the dependent clause as well. The only case when the complement clause remains subjectless non-finite is the case it is introduced via *prepc* relation and the preposition "whether". However if any such marker is missing then the clause is declarative and thus it must be controlled by a NP, so the arbitrary PRO is excluded. The cases of Wh-marked non-finite clauses will be treated in the Section 5.3.4 about the Wh-element movement.

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

Based on the above I propose Generalisation 5.3.1 enforcing obligatory control for *xcomp* clauses.

Generalization 5.3.1. If a clause is introduced by *xcomp* relation then it must have a PRO element which is bound to either subject or object of parent clause.

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

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

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

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

The generalization 5.2.7 required a test for passivization (also know in dependency grammar and SFL as *voice*). Knowing the voice of the parent clause is necessary in order to determine what NP is controlling the PRO element in the complement clause. Consider Example 90 and it's passive form 91. In both cases there is only one NP that can command PRO and it is the subject of the parent clause "Albert". So we can generalise that the voice does not play any role in controller selection in one argument clauses (i.e. clauses without a nominal complement). In Examples 92 and 93 the parent clause takes two semantic arguments. Second part of principle 5.2.2 states that in case of obligatory control PRO must be *c-commanded* by an NP. In 92 both NPs("Albert" and 'Wendy") c-command PRO element, however according to Minimality Condition(Haegeman 1991: 479) "Albert" is excluded as the commander of PRO because there is a closer NP that c-commands PRO. In case of 93 the only NP that c-commands PRO is the subject "Albert" because "by Mary" is a PP (prepositional phrase) and also only NPs can control a PRO as stated in principle 5.2.6. In the process of passivization the complement becomes subject and the subject becomes a prepositional (PP-by) complement then the latter is automatically excluded from control candidates, thus conforming to Generalisation 5.2.7 (Haegeman 1991: 281).

The above can be synthesised into Generalisation 5.3.2 appealing to linear proximity of the words. But the linear order dimension is beyond the borders of dependency grammar in the sense that the word order is not being accounted explicitly as a relation. Rather, the solution is technical: each word receives an index for the position it occupies within a sentence which suffices to implement Generalisation 5.3.2 presented in Chapter 8.

Generalization 5.3.2. The controller of PRO element in a lower clause is the closest nominal constituent of the higher clause.

The adjunct non-finite clauses such as the ones in Example 61 ("John hired Mery [PRO to fire Bill]") and 62 ("John abandoned the investigation [PRO to save money]") shall be treated exactly as the non-finite complement clauses are. Generalisation 5.2.8 emphasises obligatory control for them. The only difference between the adjunct and complement clauses is dictated by the verb of the higher clause and whether it theta marks or not the lower clause. In dependency grammar the adjunct clauses are also introduced via *xcomp* and *prepc* relations, so syntactically there is no distinction between the two patterns.

The *prepc* relation in dependency grammar introduces a prepositional clausal modifier for a verb (VN), noun (NN) or adjective (JJ). Adjective and noun modification are cases of copulative clauses. Such configuration are not relevant to the context of this work because, as we will see in Section 7.2, the dependency graphs are normalised.

This process involves, among others, transforming the copulas into verb predicated clauses instead of adjective or noun predicated clauses.

The last subordinate type concerned with the PRO element is the subject clause such as the one in Example 63 ("PRO$_i$ smoking is bad for the health$_j$"). In dependency structure, the subject non-finite clauses are introduced via *csubj* relation. They are quite different from complement and adjunct clauses because, according to generalization 5.2.9 the PRO is optionally controlled. Since in this case it is not possible to bind PRO solely on syntactic grounds, the generalization 5.2.9 proposes arbitrary interpretation discussed in Section 5.2.1. Next I turn to identifying the second type of null elements (NP-traces) in the dependency structure of a sentence.

5.3.2 NP-traces and Process Type Database

Syntactically, NP raising can occur only when there is a complement clause by moving the subject of a lower clause into a position of a higher clause. The subject of the higher clause c-commands the subject of the lower clause. This is exactly the same syntactic configuration as in the case of PRO subjects. In dependency grammar the lower clause is introduced via *xcomp* (as explained in Section 5.2.1).

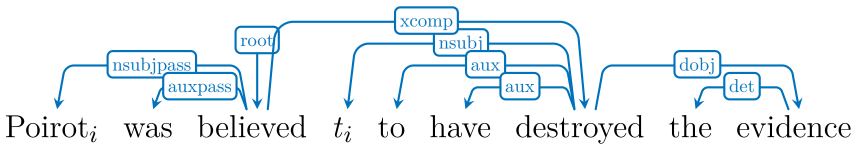

xcomp
root
nsubj
nsubjpass
aux
dobj
auxpass
aux
det
Poirot$_i$ was believed t_i to have destroyed the evidence

Fig. 5.5 Dependency parse for Example 68

Figures 5.5 and 5.6 represent a dependency parses for Examples 68 and 70. In such cases the subject position in the embedded clause is a NP-trace coindexed with the subject position in the higher clause (whether it is also a trace that is a part of a chain or an overt element). This is the case only if the embedded clause, of course, does not have already an over subject, if it is not introduced with the conditional marker "if" or with preposition "for" and if the higher clause has no nominal complement between the subject and the embedded complement clause. However, just by doing so it is not possible to distinguish whether the empty subject is a PRO or a NP trace *t*.

Table 5.4 represents the semantic role distribution for verb senses in examples above. Example 68 is a passive clause with an embedded complement clause. The subjects and complements switch places in passive clauses and so do the semantic roles. That would mean that Cognizant role goes is to be assigned to embedded/complement

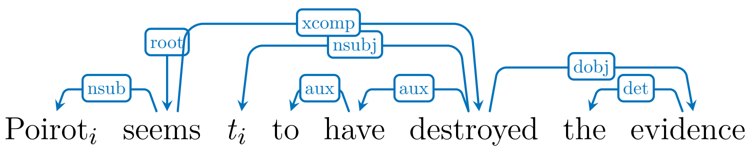


Fig. 5.6 Dependency parse for Example 70

clause. However Phenomena is the only semantic role that can be filled by a clause, all other roles take nominal, prepositional or adjectival groups.

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

Table 5.4 Semantic role distribution for verbs "believe" and "seem"

In example 70 the verb "seem" assigns roles only to complements and as the embedded clauses can take only the phenomenon role, the cognizant is left unassigned and so does not assign any thematic role to the subject which then can be filled either by an expletive or a moved NP.

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

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

Making such distinction directly involves checking role distribution of upper and lower clause verb and deciding the type of relationship between the empty category and its antecedent. If such a distinction shall be made at this stage of parsing or postponed to Transitivity analysis (i.e. semantic role labeling) is under discussion because each approach introduces different problem.

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

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

Otherwise, a second approach is to decide the element type before Transitivity analysis (semantic role labeling) and remove the 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 8.3), (b) identifying the main verbs of higher and lower clauses and correspondingly the set of possible configurations for each clause(by inquiry to process type database(PTDB) described in the transitivity analysis Section 8.2).

Generalization 5.3.3. To distinguish the *t*-traces check the following conditions:

- the subject control pattern matches the case AND

- the process type of the higher clause is two or three role cognition, perception or emotion process (considering constraints on Cognizant and Phenomenon roles). AND
- among the configurations of higher clause there is one with:
  - an expletive subject OR
  - the Phenomenon role in subject position OR
  - Cognizant in subject position AND the clause has passive voice or interrogative mood (cases of movement).

If conditions from generalization 5.3.3 are met then the empty constituent is a subject controlled *t*-trace. Now we need a set of simple rules to mark which constituents shall receive a thematic role. These rules are presented in the generalization 5.3.4 below.

Generalization 5.3.4. Constituents receiving thematic roles shall be marked with "thematic-role" label, those that do not receive a thematic role shall be marked with "non-thematic-role" and those that might receive thematic role with "unknown-thematic-role". So in each clause:

- the subject constituent is marked with "thematic-role" label unless (a) it is an expletive or (b) it is the antecedent of a *t*-trace then marked "non-thematic-role"
- the complement constituent that is an nominal group (NP) or an embedded complement clause is marked with ”thematic-role“ label.
- the complement that is a prepositional group (PP) is marked with "unknown-thematic-role".
- the complement that is a prepositional clause is marked with "unknown-thematic-role" label unless they are introduced via "that" and "whether" markers then it is marked with ”thematic-role“ label.
- the adjunct constituents are marked with "non-thematic-role"

5.3.3 Wh-trances

Let's turn now to how Wh-movement and relative clauses are represented and behave in dependency grammar and SF grammars. Figures 5.7, 5.8 present dependency parses for Wh-movement from subject and object positions of lower clause while 5.9 from adjunct position.

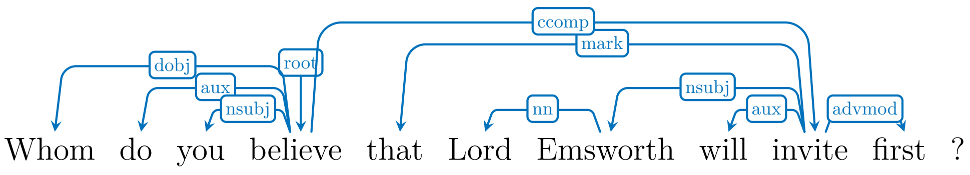


Fig. 5.7 Dependency parse for Example 79

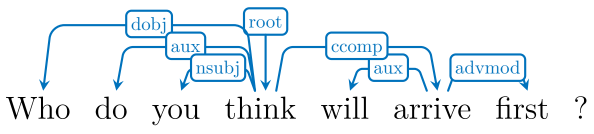


Fig. 5.8 Dependency parse for Example 78

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

For clarity purposes I define the Wh-group and Wh-element as given in Definition 5.3.1. Note that Wh-group is not a new unit class in the grammar but a constituent feature that can span across three unit classes.

Definition 5.3.1 (Wh-group, Wh-element)**.** *Wh-group* is a nominal, prepositional or adverbial group that contains Wh-element either as head or as modifier. The *Wh-element* is a unit element filled by any of the following words or their morphological derivations (by adding suffixes *-ever*, *-soever*): *who, whom, what, why, where, when, how.*

The Functional distribution for Wh-elements and Wh-groups is presented in the table 5.5 below.

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

Table 5.5 Functions and features of Wh-elements and groups

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

The essential features for capturing the Wh-movement in dependency graphs are (a) the finite complement clause identified by *ccomp* relation between the matrix and embedded clause (b) the Wh-element/group plays a complement function in higher clause which is identifiable by *dobj*, *prep* or *advmod* relations to the main verb (c) the function of the Wh-trace in lower clause is either Subject(e.g. 79), Object(e.g. 78) or Adjunct.

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

Regardless whether the syntactic function of the traces in the lower clause is Subject or Object, in the higher clause the Wh-group takes Object function and is bound to the main verb via *dobj* relation but is positioned before the main verb and the Subject (a structure corresponding to Wh-interrogatives). Wh-group can take also Subject function in the higher clause, but then it is not a case of Wh-movement and is irrelevant for us at this point because there is no empty element, i.e. Wh-trace. The attribution of clause function to the Wh-trace is based on either the case of the Wh-group or the missing functional constituent in the lower clause.

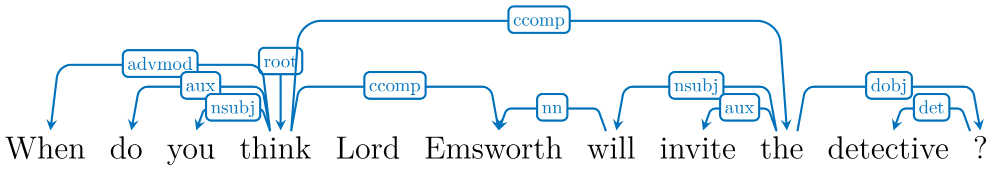


Fig. 5.9 Example dependecy parse with Adjunct Wh-element

In case of Wh-traces with Adjunct function in the lower clause, like in figure 5.9, their antecedents also receive adjunct function in the higher clause. Adjunct Wh-group cannot bind to the clause it resides in if the clause has (generic) present simple tense and thus it has to be antecedent for a trace in lower clause which has other tense or modality than present simple. The reader can experiemnt with changing tense in the example 5.9.

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

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

Not always the Wh-groups are movements from lower clause. It is possible that the trace of the moved element to reside in the higher clause (complement) or even have a case of no movement when Wh-element takes the subject function in the higher clause. 94 and 95 are good examples of a *short* movement (in clause movement). However the short movement in dependency grammar has no relevance because the dependency grammar is order free and the functions are already assigned accordingly so short movement is not a subject of interest for the current chapter.

5.3.4 Chaining of Wh-trances

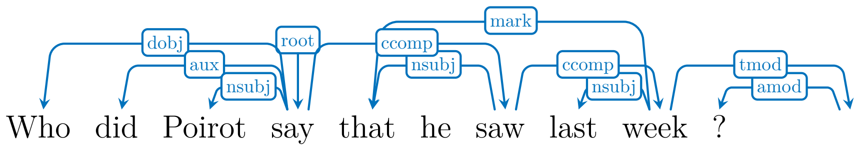


Fig. 5.10 Dependency parse for Example 88

Recall the *cyclic* and *successive* properties of Wh-movement from previous section underlined by example 88 and its dependecy parse in figure 5.10. GBT suggests that the Wh-movement leaves traces in all the intermediary clauses. In dependency grammar these properties shall be treated instrumentally for determining intermediary hops in search for the foot of the chain and none of the intermediary traces shall be created. There simply is no further purpose for them as they do not receive a thematic role in the intermediary clauses.

5.3.5 Wh-trances in relative clauses

In GB theory the Wh-elements that form relative clauses (*who, whom, which, whose*) are considered moved. In dependency grammar such movement is redundant since the Wh-element and its trace are collapsed and take the same place and function. The Wh-elements function either as subject or complement. When the relative clause is introduced by a Wh-group there is no empty element to be detected, rather there is anaphoric indexing relation to a noun it refers to. Focusing now on the relative clauses, there are three more possible constructions that introduce them: (a) a prepositional group that contains a Wh-element, (b) “that” complementizer which behaves like a relative pronoun (c) the Zero Wh-element which is an empty element and which functions the same way as a overt Wh-element. The table 5.6 lists possible elements that introduce a relative clause, their features and the functions they can take.

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

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

Table 5.6 The Wh-elements introducing a relative clause.

Compare the dependency parse with an overt Wh-element in Figure 5.11 and the covert one in 5.12.

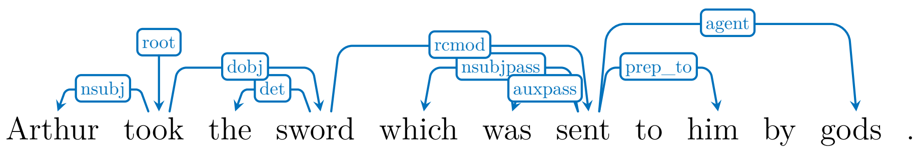


Fig. 5.11 Dependency parse for “Arthur took the sword which was sent to him by gods.”

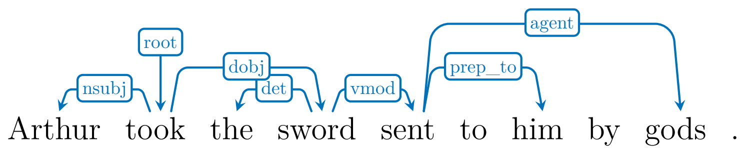


Fig. 5.12 Dependency parse for "Arthur took the sword sent to him by gods."

Relative clauses in dependency graphs are introduced by *rcmod*, *partmod*, *infmod* and *vmod* relations. The *rcmod* introduces relative clauses containing a Wh-group while the *partmod* and *infmod* introduce finite and non-finite relative clauses with Zero Wh-element. After the Version 3.3 of Stanford parser the *partmod* and *infmod* relations have been merged into *vmod*. So the dependency relation is the main signaller if empty Wh-elements even though they are subject to corrections in preprocessing (Section 7.2) phase to ensure a uniform treatment of each relation type.

The Zero Wh-element behaves exactly like the *PRO element* in the case of non finite complement clauses discussed in Section 5.2.1. It receives thematic roles in both the higher clause and in lower clause and is not a part of a chain like the cases of NP/Wh-movement.

5.4 Discussion

This chapter treats the identification of the *null elements* in syntactic structures. Section 5.2 presents how GBT theory handles null elements and then Section 5.3 shows how the same principles translate into Stanford dependency graphs.

Identification of null elements is important for the semantic role labeling process described in Chapter 8 because usually the missing elements are participant roles (theta roles) shaping the semantic configuration. The semantic configurations (gathered in a database) are matched against the syntactic structure and missing elements lead to failing (false negatives) or erroneous matches (false positives). Therefore to increase the accuracy of semantic role labeling spotting null elements is a prerequisite.

This Chapter also contributes to establishing cross theoretical connections that is among current thesis objectives. Specifically it provides translations of necessary principles and generalizations from GB theory into the context of dependency grammar. These results are directly used in Section 8.1 for generating graph patterns.

Chapter 6

On Graphs, Feature Structures and Systemic Networks

The parsing algorithm operates mainly with operations on graphs, attribute-value matrices and ordered lists with logical operators. This chapter defines the main types of graphs, their structure and how they are used in the algorithm. It also covers the operations relevant to the parsing algorithm: *conditional traversal and querying* of nodes and edges, *graph matching,pattern-graph matching* and *pattern-based node selection, insertion and update.*

While developing the Parsimonious Vole parser a set of representational and operational requirements arose. These requirements are as follows:

- arbitrary relations (i.e. typed and untyped edges)
- description rich (i.e. features of nodes and edges)
- linear ordering and configurations (i.e. syntagmatic and compositional)
- hierarchical tree-like structure (with a root node) and also orthogonal relations among siblings and non-siblings
- statements of absence of a node or edge (i.e. negative statements in pattern graphs)
- disjunctive descriptions (needed for polysemy and uncertainty)
- conjunctive descriptions (needed for multiple feature selections in recursive systems)
- (conditional) pattern specifications (i.e. define patterns of graphs)

- operational pattern specifications (i.e. a functional description to be executed in pattern graphs)

6.1 Basic Definitions

The general approach to construct an SFG parse structure revolves around the graph pattern matching and graph traversal. In this section I present the instruments used for building such structures, starting from generic computer scientific definition of graphs and moving towards specific graph types covering also the feature structures and conditional sets.

At this point you may ask why graphs and not trees since in the field of computational linguistics trees has been taken as the de facto data representation for parse structures. Firstly, the trees are a special kind of graphs and anything expressed as a graph could as well be expressed as a tree. Secondly, we gain a higher degree of expressiveness even if at the expense of computational complexity, a point to which we will come back latter in this chapter. This expressiveness is needed when dealing with interconnection of various linguistic theories.

Definition 6.1.1 (Graph). A *graph* $G = (V, E)$ is a data structure consisting of non-empty set V of nodes and a set $E \subseteq V \times V$ of edges connecting nodes.

Definition 6.1.2 (Digraph). A *digraph* is a graph with directed edges. A directed edge $(u, v) \in E$ is an ordered pair that has a start node u and an end node v (with $u, v \in V$)

Carl Pollard & Ivan Sag (1987) have formally described the useful concepts for grammatical representations of HPSG. Taking on board and extending the typed feature structure theory and extending it in several useful forms. Including definitions of hierarchy, feature structure recursion, logical evaluation and compositions and unification the key operation parsing feature structured grammars. In this thesis however I simplified feature structures to only a few concepts. This si due to mixing concept of *graph* as structure carriers with the concept of *feature structure* as mainly description carrier. This mix does not exist in (Carl Pollard & Ivan Sag 1987) who use feature structure as both structure and description carriers. Nevertheless, besides a few instrumental concepts, I proceed defining feature structure as follows:

Definition 6.1.3 (Feature Structure (FS)). A *feature structure* F is a finite set of attribute-value pairs $f_i \in F$. A *feature* $f_i = (a, v)$ is a mapping between an identifier a

(a symbol) and a value v which is either a symbol, an ordered set or another feature structure. The function $att(f_i)$ is a function returning the feature identifier $att(f_i) = a$ and the function $val(f_i)$ is a function returning the ascribed value of a feature $(f_i) = v$

Definition 6.1.4 (Ordered Set)**.** An *ordered set* $S = \{o_1, o_2, ..., o_n\}$ is a finite well defined collection of distinct objects o_i arranged in a sequence such that $\forall o_{i-1}, o_i \in S$: $o_{i-1} < oi$

Definition 6.1.5 (Conjunction Set)**.** A *Conjunction Set* $S_{conj} = (S, conj)$ is an ordered set S whose interpretation is given by the logical operand $conj$ (also denoting the type of the set) such that $\forall o_i, o_j \in S : conj(o_i, o_j)$ holds.

The conjunction sets used in current work are *AND-set* (S_{AND}), *OR-set* (S_{OR}), *XOR-set* (S_{XOR}) and *NAND-set* (S_{NAND}). The assigned logical operands play a role in the functional interpretation of conjunction sets.Formally these sets are defined as follows.

Definition 6.1.6 (AND-Set)**.** $AND - Set$ (also called *conjunctive set*) is a conjunction set $S_{AND} = \{a, b, c...\}$ that is interpreted as a logical conjunction of its elements $a \wedge b \wedge c \wedge ...$

Definition 6.1.7 (NAND-Set)**.** $AND - Set$ (also called *negative conjunctive set*) is a conjunction set $S_{NAND} = \{a, b, c...\}$ that is interpreted as a negation of the logical conjunction of its elements $a \uparrow b \uparrow c \uparrow ...$ equivalent to $\neg(a \wedge b \wedge c \wedge ...)$

Definition 6.1.8 (OR-Set)**.** $OR - Set$ (also called *disjunctive set*) is a conjunction set $S_{OR} = \{a, b, c...\}$ that is interpreted as a logical disjunction of its elements $a \vee b \vee c \vee ...$

Definition 6.1.9 (XOR-Set)**.** $OR - Set$ (also called *exclusive disjunctive set*) is a conjunction set $S_{XOR} = \{a, b, c...\}$ that is interpreted as a logical exclusive disjunction of its elements $a \bigoplus b \bigoplus c \bigoplus ...$ equivalent to $(a \wedge \neg(b \wedge c \wedge ...)) \vee (b \wedge \neg(a \wedge c \wedge ...)) \vee (c \wedge \neg(a \wedge b \wedge ...))$

When conjunction sets are used as values in FSs then the type of logical operand dictates the interpretation of the FS. When the set type is S_{AND} then all the set elements hold simultaneously as feature values. If it is a S_{OR} then one or more of the set elements hold as values. If is S_{XOR} then one and only one of set elements holds and finally if it is a S_{NAND} set then none of elements hold as feature values.

I use τ function defined $\tau : S \to \{S_{AND}, S_{OR}, S_{XOR}, S_{NAND}\}$ to return the type of the conjunction set and *size* function defined $size : S \to \mathbb{N}$ to return the number of elements in the set.

Definition 6.1.10 (Feature Rich Graph (FRG))**.** A *feature rich graph* is a digraph whose nodes V are feature structures and whose edges $(u,v,f) \in E$ are three valued tuples with $u,v \in V$ and $f \in F$ an arbitrary feature structure.

Further on, when I refer to a graph I will a feature rich digraph, unless otherwise stated. The parsing algorithm operates with three feature graph types: *Dependency Graphs* (DG) (example figure 6.1), *Constituency Graphs* (CG) also called Mood Constituency Graphs (MCG) (example figure 6.2) and Pattern Graphs (PG) which can also be viewed as *Query Graphs* (QG).

Definition 6.1.11 (Dependency Graph)**.** The *dependency graph* is a feature rich digraph whose nodes V correspond to words, morphemes or punctuation marks in the text and carry but not limited to the following features: *word*, *lemma*, part of speech (*pos*) and when appropriate the named entity type (net); while the edges E describe the dependency relation (*rel*).

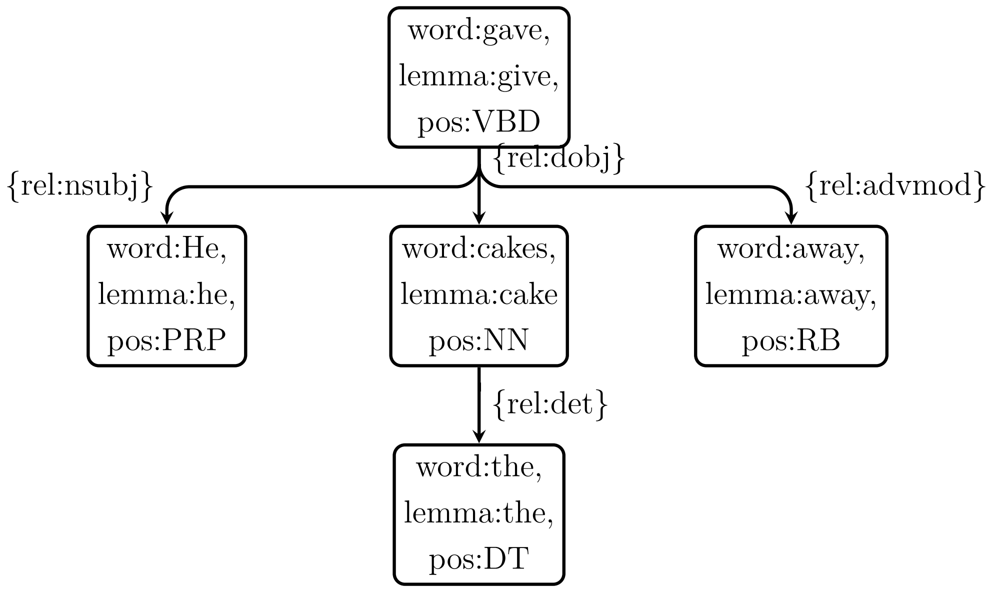


Fig. 6.1 Dependency graph example with FS nodes and edges

Definition 6.1.12 (Constituency Graph)**.** The *constituency graph* is a feature rich digraph whose nodes V correspond to SFL *units* and carry but not limited to the unit class and the function within the parent unit (except for the root node); while the edges E carry mainly the constituency relation between parent and daughter nodes but also potentially other relation types.

The basic features of a constituent node are the *unit class* and the function(s) it takes which is to say the *element(s)* it fills in the parent unit (see theoretical aspects of SFL in Chapter 3). The root node (usually a clause) is an exception and it does not act as a functional element because it does't have a parent unit. The leaf nodes carry the same features as the DG nodes plus the class feature which correspond to the traditional grammar part of speech.

Apart from the essential features of class and function, the CG nodes carry additional class specific features selected from the relevant system network. For example gender, number and animacy are specific to nominal unit classes such as nouns and nominal group; while tense and modality are specific mainly to clause class.

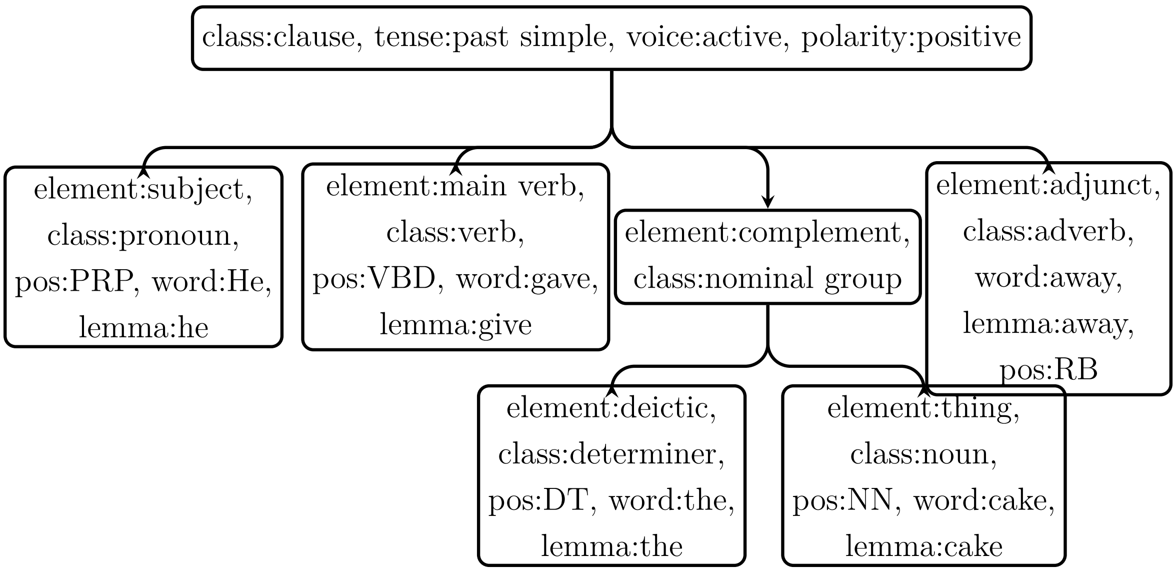

class:clause, tense:past simple, voice:active, polarity:positive
element:subject, class:pronoun, pos:PRP, word:He, lemma:he
element:main verb, class:verb, pos:VBD, word:gave, lemma:give
element:complement, class:nominal group
element:adjunct, class:adverb, word:away, lemma:away, pos:RB
element:deictic, class:determiner, pos:DT, word:the, lemma:the
element:thing, class:noun, pos:NN, word:cake, lemma:cake

Fig. 6.2 Constituency graph example

Regardless of the graph type, constituency or dependency, it is essential to express patters over those graphs in order to facilitate various operations. The PG (defined in 6.1.13) are special kinds of graphs meant to represent small (repeatable) parts of parse graphs that stand for a certain grammatical feature. The patterning is described as both graph structure and feature presence (or absence) in node or edge. Such pattern graphs can be viewed as the conditional part of a query language if not the entire query statements (i.e. select, insert, update, delete operation descriptions). This kinds of graphs need last four requirements listed in the beginning of this section.

Definition 6.1.13 (Pattern Graph)**.** A *pattern graph* (PG) describes regularities in node-edge configuration and feature structure including descriptions of *negated nodes or edges* (i.e. absence of), logical operators over feature sets (AND, OR, XOR and NAND) and operations once the pattern is identified in a target graph (select, insert, delete and update).

The feature structure of a PG are always *underspecified* as compared to the dependency or constituency graph in the sense that irrelevant attribute-value pairs are omitted sometimes down to an empty FS. However often it is useful to specify more than one value for the same feature as a list of disjunctive values allowing the pattern to cover larger set of possible cases. I will call these FS as being *over-specified*.

An example of PG is depicted in figure 6.6 in the Section 6.6. It deals with pattern graph matching and other operations with in detail. But before that I will first briefly cover generic operations on graphs and the problem of graph matching also known in computer science as the *graph isomorphism* problem.

6.2 More on Pattern Graphs

As already mentioned in the previous section we are dealing with a special case of graph isomorphism precisely because the graphs we consider are feature rich graphs. Specifically, besides the graph structure, the node and edge identity checking is a secondary check to consider.

Before I dive into pattern graph matching and operations with pattern graphs I will first discuss two example of pattern graphs. Consider the case of present perfect continuous tense which traditional grammar defines as in Table 6.1 regardless of the element order.

has/have	+	*been*	+	*Vb-ing*
to have, present simple		to be, past participle		verb, present participle

Table 6.1 Present perfect continuous pattern

Examples 96-98 show variations of this tense in a simple clause according to the mood and "has" contraction. Of course there are also voice variations but I skipped them in this example because it adds combinatorialy to the number examples. The Figures 6.3-6.5 represent dependency parses for the above examples.

(96) He has been reading a text.

(97) He’s been reading a text.

(98) Has he been reading a text?

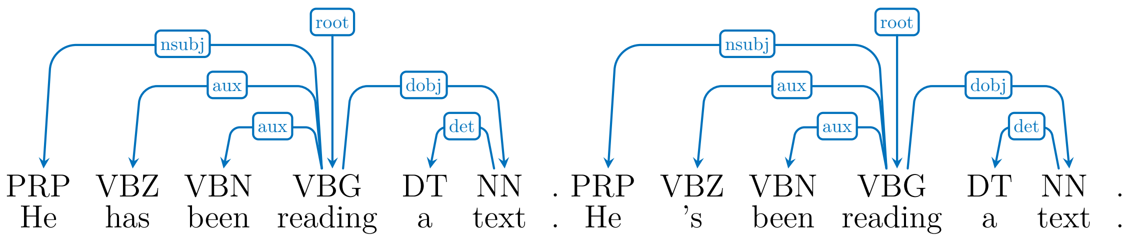


Fig. 6.4 Present perfect continuous: indicative mood, contracted “has”

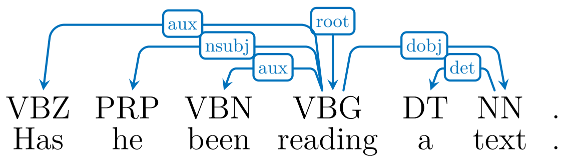


Fig. 6.5 Present perfect continuous: interrogative mood, un-contracted “has”

The present perfect continuous tense can be formulated as a pattern graph (including voice) over the dependency structure as illustrated in Figure 6.6 below.

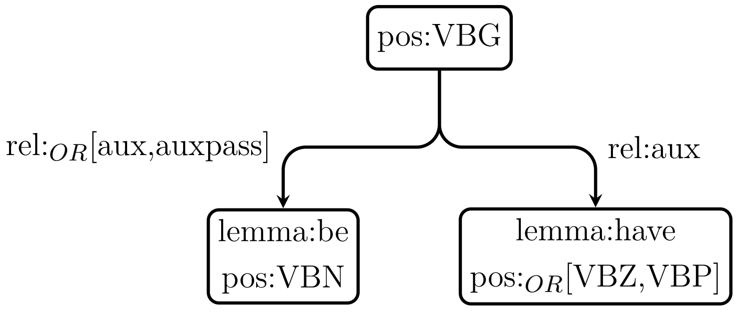


Fig. 6.6 The graph pattern capturing features of the present perfect continuous tense

In this a pattern the main lexical verb is *present participle* indicated via *VBG* part of speech. It is accompanied by two auxiliary verbs: *to be* in *past participle* (*VBN*) form and *to have* in *present simple* form specified by either *VBZ* for 3rd person or *VBP* for non-3rd person. Also the *to be* can be either connected by *aux* relation or in case of passive form by *auxpass* relation. Note that the pattern in Figure 6.6 over-specifies the edge type (using the OR set notation) to the verb *to be* which can be either *aux* or *auxpass* and the part of speech of the verb *to have* which can be *VBZ* or *VBP*.

One of the fundamental features of language is its sequentiality and directionality. Place and order of constituent elements play an important role in SFG (see Section 3.2.1). Unfortunately capturing the aspect of order is not straight forward in graphs which inherently lack the capacity to capture linear order specifically. In the simplest form, they just describe connections between nodes and are agnostic to any meaning or interpretation.

To demonstrate how the order is specified in the graph patterns, lets turn now to the clause mood and capture specifically the distinction between declarative and Yes/No interrogative moods. In SFG this features is described in terms of relative order of clause elements. If the finite is before the subject then mood is Yes/No-interrogative whereas when the finite succeeds subject then mood is declarative. The example 98 clearly contrasts in mood with 96 and 97.

Order can be specified in absolute or relative terns, partially or exhaustively. To overcome this problem I introduce three special features that cove all cases: the node *id*, *precede* and *position*. Node id take a token to uniquely identifies a node, the precede feature takes ordered sets to indicate the (partial) precedence of node ids, and the position feature indicates the absolute position of a node.

One way to introduce order among nodes is marking them with an absolute position. The parse graphs i.e. DGs and MCGs are automatically assigned at the creation time the absolute position of the node in the sentence text via the feature *position*. Only the leaf nodes can have a position assigned. The leaf nodes position corresponds to the order number in which they occur in the sentence text while the non-leaf node's position is calculated to the lowest of it's constituent nodes. The absolute position description rarely is used in the PGs, mainly for stating that a constituent is the first or the last one in the sentence.

Another way to specify node order is through relative positions for which the node id and the precedence features are introduced.

Continuing the example of mood features Figures 6.7 and 6.8 illustrate the use of relative and absolute node ordering constrains for declarative mood. In PGs, the relative order is preferred to absolute one but either is usable. Figure 6.9 depicts the PG for the Yes/No interrogative mood using the relative node ordering.

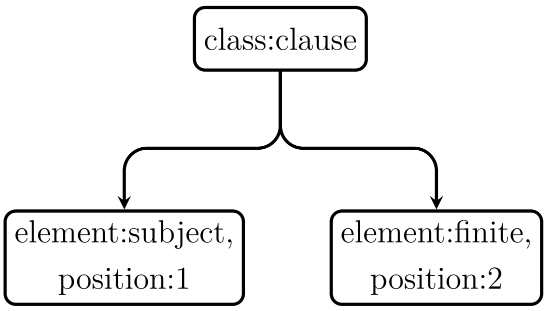
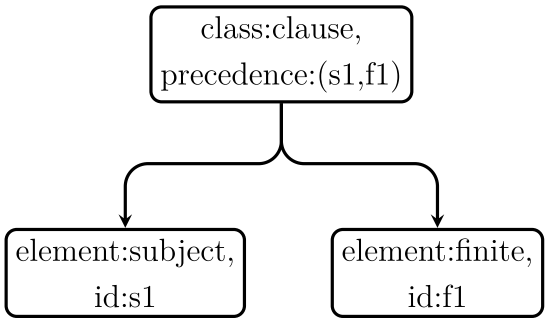


Fig. 6.8 Declarative mood pattern graph with absolute element order



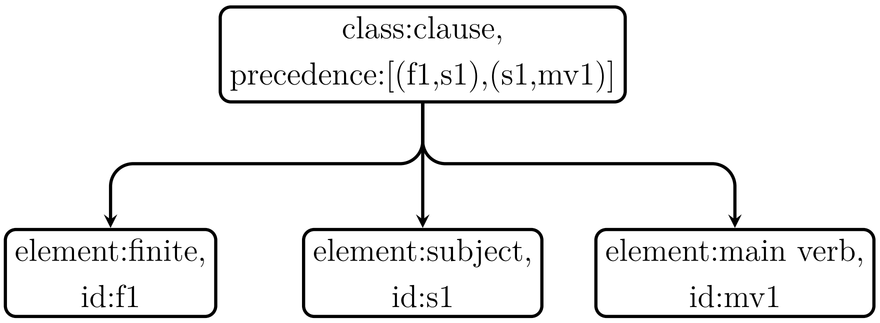

class:clause, precedence:[(f1,s1),(s1,mv1)]
element:finite, id:f1
element:subject, id:s1
element:main verb, id:mv1

Fig. 6.9 Pattern graph for Yes/No-interrogative mood with a redundant main verb node

remove the main verb nod

From the usability point of view there are few technicalities I shall emphasize. First, the precedence feature can be used on any node. When matched, the precedence declarations are collected from all nodes into a single set before being checked. However I recommend as a good practice to specify the order of nodes on the parent constituent.

Second, the notation in Figure 6.9 follows the Python bracketing meaning i.e. the round brakes signify tuples while the square ones lists (ordered sets). So the main verb element is redundant and is introduced to demonstrate multiple order specifications. However the order can be either specified as a set of binary tuples or as an ordered set (i.e. a Python list). So precedence:[(f1,s1),(s1,mv1)] is equivalent to precedence:[f1,s1,mv1].

Thirdly the ordering can be defined in absolute terms via position or in relative terms. Note that in the case of PGs the absolute ordering of nodes is interpreted relatively so PG in figure 6.7 is identical to 6.8.

Patterns like the ones explained above can be created for many other grammatical features and tested via graph pattern matching operation whether the feature is found in the parse graph or not. Once the pattern is identified it can act as a triggering condition to various operation affecting the parse graph or other external structures which I describe in the next sections. For example once the pattern 6.6 is identified then the clause can be marked with the *tense* feature or in the case of dependency structure clause corresponding to the dominant verb node.

6.3 Graph Operations

Definition 6.3.1 (Atomic Query)**.** *Querying* $q_V(F,G)$ or $q_E(F,G)$ over the nodes V or edges E of a graph G is an operation that returns a set of nodes or edges filtered by the conditional feature structure F.

For example, for a given dependency graph, to select all the determiners we query the nodes with condition that part of speech has DT value i.e. *pos=NP* or to select all the edges connection a noun to it's determiner then the query is formulated for all edges whose relation type is *rel=det*.

The graph traversal defined in 6.3.2 is another important operation. For example DG traversal is used in bootstrapping the CG as a parallel structure. Another usage is conditional sub-graph selection of a given DG or CG. For example in the semantic enrichment phase (see Section 7.5), to ensure that the semantic patterns are applied iteratively to each clause, from a multi-clause MCG graphs are selected each individual clause sub-graphs without including the embedded (dependent) clauses. Using sub-graphs for performing the pattern matching, like in the case of semantic enrichment, decreases drastically the complexity of the graph isomorphism problem (described in Section 6.4) thus increasing the overall performance.

Definition 6.3.2 (Traversal)**.** Traversal $t(V_S,G)$ of a graph G starting from node V_S is a recursive operation that returns a set of sequentially visited nodes the neighbouring each other in either *breadth first* (t_{BF}) or *width first* (t_{WF}) orders.

Definition 6.3.3 (Conditional Traversal)**.** *Conditional traversal* $t(F_V,F_E,V_S,G)$ of the graph G starting from node V_S under node conditions F_V and edge conditions F_E is a traversal operation where a node is visited if and only if its feature structure conditionally fulfils the F_V and the edge that leads to this node conditionally fulfils the F_E.

The graph traversal can be used in various ways, either for searching a node, an edge or finding a sub-graph that fulfils certain conditions on its nodes and edges if it is a conditional traversal. A traversal can be also used to execute generative operations on parallel data structure.

Definition 6.3.4 (Generative Traversal)**.** *Generative traversal* $m(M,G)$ *of a graph* G *via a operation matrix* M *is an operation resulting in creation of another graph* H *by contextually applying generative operations. The operation matrix* M *is a set of tuples* (ctx,o,p) *that link the visited node context* ctx *(as features of the node, the edge and previously visited neighbour) to a certain operation* o *that shall be executed on the target graph* H *with parameters* p*.*

Now that generative traversal is defined, you may point out that similarly (or by analogy) update, insert and delete traversals can be defined on the source or target graph by using the same mechanism of *operation matrices* mapping contexts of visited nodes and edges to update, insert and delete operations.

But such traversal operations cannot be easily defined. While traversal context may be sufficient for generative operations on a new structure, it is insufficient for executing affecting operations on the traversed graph. To overcome this limitation, instead of using traversal context I take a different approach: the *pattern graphs*, defined in previous section combined with generic graph matching algorithm. This mechanism offers a similar algorithmic independence of mapping structural context to operation(s) triggered by it.

6.4 Graph Matching

The graph matching problem is known in computer science as *graph isomorphism problem* which is an NP-complete problem. A peculiar characteristic of such problems is that given a solution then it can be very quickly *verified* in polynomial time but the time required to find a solution increases exponentially with the size of the problem. Therefore to prove whether or not such problems can be solved quickly is one of the main unsolved problems in computer science and the performance of of algorithms solving NP-complete problems is an important issue and requires careful investigation.

Definition 6.4.1 (Graph Matching)**.** *For two given graphs* G *and* H *where* $H \leq G$*, graph matching* (or *graph isomorphism*) *is an operation of finding an sub-graph* $G_1 \subseteq G$ *that is structurally isomorphic to* H*.*

To the moment no algorithm exist to solve the graph isomorphism problem in polynomial time, however the latest available algorithms such as VF2 Cordella et al.

(2001, 2004) or QuickSI Shang et al. (2008) performs the task quickly when the addressed graphs are of limited size.

Next I present some estimate calculations and compare to benchmarking study of Lee et al. (2013).

The graphs used in benchmarking tests described by (Lee et al. 2013) on AIDS dataset are composed of 2–215 nodes and 1–217 edges on which VF2 algorithm performs the isomorphism problem on average in 20–25 milliseconds for sub-graphs sizing between 4–24 edges. The NASA dataset (used in the same benchmarking study) which contains 36790 graphs sizing between 2–889 nodes and 1–888 edges the VF2 algorithm performs on average in 250 milliseconds for sub-graphs of 4 edges.

To put it into the context we have to answer the questions: how big the sentence graphs are, what size are the patterns and what would be a rule of thumb estimation of performance?

According to (Koeva et al. 2012), on average, an English sentence is composed of 12-20 words(n) with about 1.6 clauses per sentence . The parse graph of an average English sentence is a tree or very close to a tree whose number of nodes is within the limits between $n+1$ and $2n-1$ for a n number of leaf nodes (in our case the words). So for a sentence of 20 words the parse tree would be maximum 39 nodes.

Lets assume the size of a sentence is ten times the average i.e. 200 words and a maximum estimate of 399 nodes in the parse tree. The patterns used in current work are 1–5 edges. Overall in the parsing algorithm, the graph matching is mostly applied at the clause level which on average in English is of 6 8 words yields an average maximum of 15 nodes per parse graph which is 0.38 of the average sentence and 0.01 of the unusually big sentence. As used in the current implementation and given relatively small graphs, the performance VF2 algorithm fits well within reasonable time limits.

6.5 Rich Graph Matching

In order to accommodate feature rich graphs (FRG), VF2 algorithm is extended to perform custom identity checks. In the original implementation two node V_1 and V_2 are said to be equal if the nodes are of simple data types (e.g. integer or string) and they carry the same value. In our case, feature structures are attached to edges and stand for graph nodes. And there are cases when two nodes, even if the have somehow different structures, to be considered the same.

Therefore identity of complex structures becomes an elastic concept. Of course strict identify checking function is important, but we can derive more power from a nuanced check instead of a strict identity.

The functions that decide the identity of two objects (i.e. which is to say that they are the same) are called *morphisms*.

Definition 6.5.1 (Morphism)**.** A morphism (also called *identity function*) $f: X \to Y$ is a structure preserving map from one object X to the other Y where the objects are complex structures such as sets, feature structures or graphs.

Definition 6.5.2 (Identity Morphism)**.** for every object X, there exists a morphism $id_X : X \to X$, called *identity morphism* on X, such that for every morphism $f: A \to B$ we have $id_B \circ f = f = f \circ id_A$

In this work, for *identity morphism* I use interchangeably the terms *identity checking function* or *identity function*.

Definition 6.5.3 (Isomorphism)**.** The morphism $f: X \to Y$ is called *isomorphism* if there exists a morphism $g: Y \to X$ such that $f \circ g = id_X$ and $g \circ f = id_X$

I already have defined (somehow ahead) what is graph isomorphism; and that the graph matching should always be an isomorphic function. However the nodes and edges shall be rather loosely identical or they shall only be considered identical but not necessarily be so. Therefore the node and edge morphism functions shall rather be *asymmetric* or simply a morphisms without any assumptions of exact identity. Now I can define the rich graph matching as follows.

Definition 6.5.4 (Rich Graph Matching)**.** For two given graphs G and H where $H \leq G$ and two *morphism functions* f_V and f_E, the *rich graph matching* is the function that finds a structural isomorphism between H and $G_1 \subseteq G$ provided that for all nodes $V_i \in H$ their *morphism function* $V_j \in G_1$ satisfies the identity function $f_V(V_i) = V_j$

So, the functions that compare whether two node or edge are the same in fact compare their feature structures and in fact the sameness is defined in accordance with the goals of the particular task. That's why identity checking function is provided as a parameter to the rich graph matching algorithm.

How is the node and edge identity checking done (or how are the morphism functions defined) is covered in the next section. What is important to mention here is the complexity of such nuanced checks since we have discussed the complexity of VF2 algorithm only on graphs where the edges and nodes are simple data structures.

The comparison of two FS is a PTIME problem that is efficiently solvable in polynomial time. Of course, this (slightly) increase the complexity of the matching process as a whole but still this lies well within the limits of practical computability.

The current implementation of VF2 algorithm includes the custom morphism functions for nodes and edges. So far I have not encountered performance issues with it. For the future however it would be of tremendous value to know exactly how much is the original VF2 algorithm burdened by such extra checks and what are the reasonable upper limits for the node size (i.e the complexity of the node feature structure). And perform some stress testing for the whole enterprise.

6.6 Pattern Graph Matching

An extension and particular case of rich graph matching is the *pattern graph matching* where H (Definition 6.5.4) is a pattern graph (Definition 6.1.13) and the identity checking function(s) are not strict but permissive to feature over-specification of node and edge feature structures.

I will define now how the identity checking function (identity morphism) used for pattern graph matching. It operates on feature structure values, which can be either atomic types *simple* of one of the conjunctive sets: S_{AND}, S_{OR}, S_{XOR} and S_{NAND}. I use both set theoretic notations for inclusion $\subseteq$, intersection $\cup$ and element belonging to a set $\in$ and the logical notations for conjunction $\wedge$ and tautology $\top$. The unary function $T(x)$ returns the type of x element.

When checking the identity of two feature values, three cases can be asserted: $x = y$ i.e. x is definitely equal to y, $x \neq y$ i.e. x is definitely different from y and $x \sim y$ i.e. x is maybe (or could be) equal to y.

I define below two identity morphisms: (a) *permissive* $I_{permissive}$ (defined by Equation 6.2) which includes the uncertain cases and (b) *strict* I_{strict} (defined by Equation 6.1) which excludes the uncertain cases. The main difference between the two morphism functions is whether on the right side (the instance graph) any uncertainty is accepted. This is to say any of the disjunctive sets s_{OR} and S_{XOR}.

Definition 6.6.1 (Strict Pattern Graph Matching)**.** *Strict pattern graph matching* is a rich graph matching where a morphism for the pattern graph H is found in the target graph G given that $H \leq G$ and that for any node $p \in H$ there is a node $r \in G$ satisfying the strict identity morphism $I_{strict} : p \to r$

$$I_{strict}: p \rightarrow r \models \begin{cases} p = r, & \text{if } T(p) = simple \land T(r) = simple \\ p \in r, & \text{if } T(p) = simple \land T(r) = S_{AND} \\ p \subseteq r, & \text{if } T(p) = S_{AND} \land T(r) = S_{AND} \\ p \cap r \neq \varnothing, & \text{if } T(p) = S_{OR} \land T(r) = S_{AND} \\ r \in p, & \text{if } T(p) = S_{OR} \land T(r) = simple \\ r \in p, & \text{if } T(p) = S_{XOR} \land T(r) = simple \\ p \cap r = \varnothing, & \text{if } T(p) = S_{NAND} \land T(r) \in \{S_{AND}, S_{OR}, S_{XOR}\} \\ r \notin p, & \text{if } T(p) = S_{NAND} \land T(r) = simple \\ \top, & \text{if } T(p) = S_{NAND} \land T(r) = S_{NAND} \end{cases} \tag{6.1}$$

Definition 6.6.2 (Permissive Pattern Graph Matching)**.** *Permissive pattern graph matching* is a rich graph matching where a morphism for the pattern graph H is found in the target graph G given that $H \leq G$ and that for any node $p \in H$ there is a node $r \in G$ satisfying the permissive identity morphism $I_{permissive}: p \rightarrow r$

$$I_{permissive}: p \rightarrow r \models \begin{cases} p = r, & \text{if } T(p) = simple \land T(r) = simple \\ p \in r, & \text{if } T(p) = simple \land T(r) \in \{S_{AND}, S_{OR}, S_{XOR}\} \\ p \subseteq r, & \text{if } T(p) = S_{AND} \land T(r) = S_{AND} \\ p \cap r \neq \varnothing, & \text{if } T(p) = S_{OR} \land T(r) \in \{S_{AND}, S_{OR}, S_{XOR}\} \\ r \in p, & \text{if } T(p) = S_{OR} \land T(r) = simple \\ p \cap r \neq \varnothing, & \text{if } T(p) = S_{XOR} \land T(r) \in \{S_{OR}, S_{XOR}\} \\ r \in p, & \text{if } T(p) = S_{XOR} \land T(r) = simple \\ p \cap r = \varnothing, & \text{if } T(p) = S_{NAND} \land T(r) \in \{S_{AND}, S_{OR}, S_{XOR}\} \\ r \notin p, & \text{if } T(p) = S_{NAND} \land T(r) = simple \\ \top, & \text{if } T(p) = S_{NAND} \land T(r) = S_{NAND} \end{cases} \tag{6.2}$$

Of course these two are not the only identity morphisms that can be defined for the pattern matching. The implementation accepts any binary function that returns a truth value meaning that the two arguments shall be considered the same or not. Moreover the identity function can be provided for edges as well, but I skip it in the current thesis because I do not use. In the future it would be useful to define the edge morphism functions and identify the use cases that employ them.

Now that I have defined how patterns are identified in the graphs, lets take a look at more advanced applications of it. In the next section I explain how the graph isomorphism can be enacted once they are identified.

6.7 Pattern-Based Operations

The patterns are searched for in a graph always for a purpose. Graph isomorphism is only a precondition for another operation, be it a simple selection (i.e. non-affecting operation) or an affecting operation such as feature structure enrichment (on either nodes or edges), inserting or deleting a node or drawing a new connection between nodes. So it seems only natural that the end goal is embedded into the pattern, so that when it is identified, also the desired operation(s) is(are) triggered. I call such graph patterns *affordance patterns* (Definition 6.7.1). Next I explain how to embed the operations into the graph pattern and how they are used in the algorithm.

The operational aspect of the pattern graph is specified in the node FS via three special features: *id*, *operation* and *arg*. The *id* feature (the same as for relative node ordering) is used to mark the node for further referencing as argument of an operation, the *operation* feature names the function to be executed once the pattern is identified and the *arg* feature specifies the function arguments if any required and they are tightly coupled with function implementation. So far the implemented operations are *insert*, *delete* and *update*. But anyone that finds appropriate can extend it with any other operations that may be useful.

Definition 6.7.1 (Affordance Graph Pattern)**.** An *affordance graph pattern* is a graph pattern that, at least one node or edge, has *operation* and *arg* features.

I say that the affordance patterns are enacted once they are tested for isomorphism in another graph and if one is found then all the defined operations are executed accordingly.

Definition 6.7.2 (Affordance Graph Enacting)**.** For an affordance graph H and a target graph G, *affordance graph enacting* is a two step operation that first performs the permissive or strict pattern graph matching and if any isomorphism graph $G_1 \subseteq G$ is identified and second for every node $p \in H$ with an operation features, executes that operation on the corresponding node $r \in G_1$ of the isomorphism.

6.7.1 Pattern-Based Node Selection

It is often needed to select nodes from a graph that have certain properties and are placed in a particular configuration. This operation is very similar to the *atomic graph query* defined in 6.3.1. The main difference is ability to specify that the node is a part of the a certain structural configuration which is not possible via the atomic query.

For example let's say that we are interested in all nodes in a dependency graph that can take semantic roles specifically subjects, and complements of the clause. For the sake of simplicity example I exclude prepositional phrases and embedded clauses that sometimes can also take semantic roles. The pattern identifying such nodes looks like the one in Figure 6.10. It selects all the nodes that are connected via *nsubj*, *nsubjpass*, *iobj*, *dobj* and *agent* edges to a VB node.

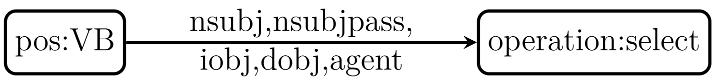


Fig. 6.10 Graph pattern that selects all the nodes that can receive semantic roles

6.7.2 Pattern-Based Node (FS) Update

There are other cases when the FS of the nodes needs to be updated either by adding or altering a feature value. This can be achieved via *pattern-based update* operation. For example, consider the example analysis 6.2 and the task to assign *Agent* feature to the subject node and *Possessed* feature to the complement. PG depicted in figure 6.11 fulfils exactly this purpose.

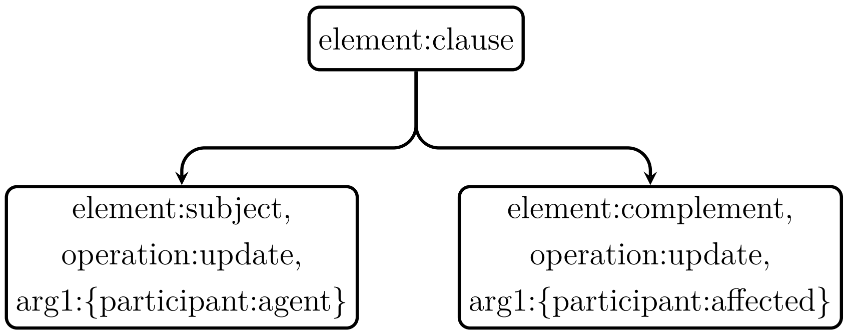


Fig. 6.11 Graph pattern for inserting the agent and affected participant role features to subject and direct object nodes.

Consider the very same pattern, but applied to a sentence in the Table 6.3. The clause has two complements and they are by no means distinguished in the pattern

class:clause				
element:subject	element: main verb	element:complement		element:adjunct
He	gave	the	cake	away.

Table 6.2 MCG with a transitive verb

graph. When such cases are encountered the PG yields two matches, (each with another complement) and the update operation is executed to both of the complements. To overcome such cases from happening PG allow defining *negative nodes*, meaning that those are nodes that shall be missing in the target graph.

For example to solve previous case I define the PG depicted in figure 6.12 whose second complement is a negative node and it is marked with dashed line. This pattern is matched only against clauses with exactly one complement leaving aside the di-transitive ones because of the second complement.

class:clause				
element:subject	element: main verb	element:complement	element:complement	
He	gave	her	the	cake.

Table 6.3 MCG with a di-transitive verb

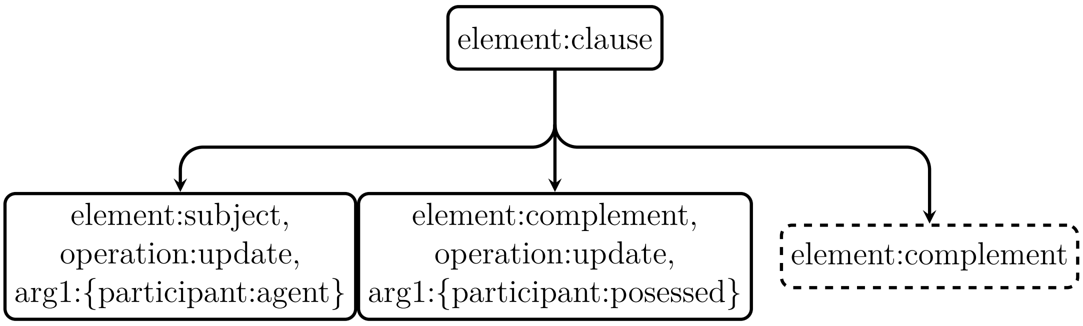


Fig. 6.12 PG for inserting agent and possessed participant roles to subject and complement nodes only if there is no second complement.

The current implementation of matching the patterns that contain negative nodes is performed in two steps. First the matching is performed with the PG without the negative nodes and in case of success another matching is attempted with the negative nodes included. If the second time the matching yields success then the whole matching process is unsuccessful but if the second phase fails then the whole matching process is successful because no configuration with negative nodes is detected.

For the sake of explanation I call the pattern graph with all the nodes (turned positive) *big* and the pattern graph without the nodes marked negative *small.* So then, matching a pattern with negative nodes means that matching the *big* pattern (with negative nodes turned into positive) shall fail while matching the *small* one (without the negative nodes) shall yield success.

6.7.3 Pattern-Based Node Insertion

In English language there are cases when an constituent is missing because it is implied by the (grammatical) context. These are the cases of Null Elements treated in the Chapter 5.

(99) Albert asked [∅ to go alone].

Consider the Example 99. There are two clauses: first in which Albert asks something and the second where he goes alone. So it is Albert that goes alone, however it is not made explicit through a subject constituent in the second clause. Such implied elements are called *null or empty constituents* discussed in detail in the Section 5.2. The table 6.4 provides a constituency analysis for the example and the null elements (in italic) are appended for the explicit grammatical account. In the Section 5.3 I offer the grammatical account of the graph patterns that insert these null elements into the parse graphs (so in fact extensively using the pattern based node insertion treated here).

class:clause					
element: subject	element: main verb	element: complement, class:clause			
		element: subject	element: main verb		element: adjunct
Albert	asked	*Albert*	to	go	alone.

Table 6.4 The constituency analysis that takes null elements into consideration

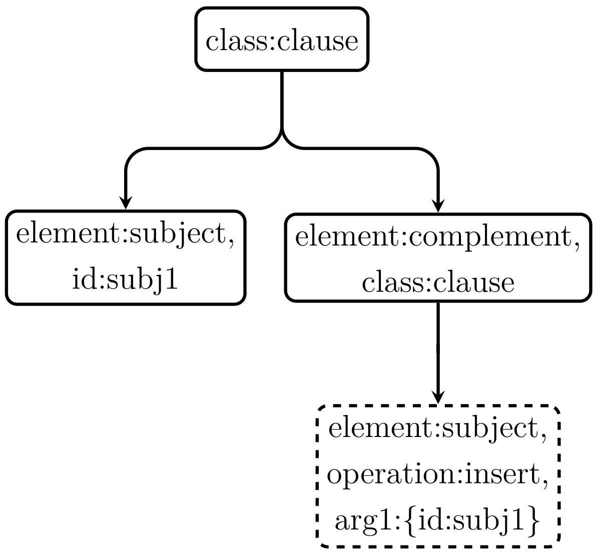

class:clause
element:subject, id:subj1
element:complement, class:clause
element:subject, operation:insert, arg1:{id:subj1}

Fig. 6.13 A graph pattern to insert a reference node

To insert a new node the, PG needs to specify that (1) the inserted node does not already exist, so it is marked as negative node, (2) specify *operation:insert* in the FS of the same and (3) provide id of the referenced node as FS argument (arg1) if one shall be taken.

In operational terms, the insertion operation means that the whole pattern will first go through a matching process. If there is a match then the new node is created. A peculiar thing about the created node is that it may keep a reference to another node or not. In our example it does keep a reference to the subject of dominant clause. If so, then all the features of the referee node are inherited by the new node. And if any are additionally provided then the new node overrides the inherited ones.

This section concludes our journey in the world of graph patterns, isomorphisms and graph based operations. Leaving only one more important data structure to cover: the system networks.

6.8 Systems and Systemic Networks

In the Section 3.2.4 I present the basic definition of System and System Network and the notations as formulated in the SF theory of grammar. In this section I formalise them in terms of what may be represented and instantiated in computational terms. In addition I cover few more useful concepts for implementation of system networks applied to enrichment of constituents with systemic features.

First I would like to introduce abstract concept of *hierarchy* defined in a computer scientific way by Carl Pollard & Ivan Sag (1987). This is a formal rephrasing of Definition 3.2.1 that Haliday provides.

Definition 6.8.1 (Hierarchy)**.** A hierarchy is finite bounded complete partial order $(\varDelta, \prec)$.

The next concept that required higher order of formalization os that of a System first established in Definition 3.2.10. For precision purposes, this one has a narrower scope without considering the system networks or precondition constraints which are introduced shortly afterwards building upon current one.

Definition 6.8.2 (System)**.** A *system* $\Sigma = (p, C)$ is defined by a finite disjoint set of distinct and mutually defining terms called a *choice set* C and an *entry condition* p establishing the delicacy relations within a system network; subject to the following conditions:

1. the choice set is a S_{OR} or S_{XOR} conjunction set.
2. the entry condition is a S_{OR}, S_{XOR} or S_{AND} conjunction set.
3. $$\infty > size(C) \geq \begin{cases} 2, & \text{if } T(C) = S_{XOR} \\ 3, & \text{if } T(C) = S_{OR} \end{cases}$$

There is a set of functions applied to system: $label(\Sigma) = l$ is a function returning the system name, $choices(\Sigma) = C$ is a function returning the choice set, $precondition(\Sigma) = p$ is a function returning the entry condition, and the $size(\Sigma)$ return the number of elements in the system choice set.

Definition 6.8.3 (Systemic delicacy)**.** We say that a system S_1 is more delicate than S_2 denoted as $S_1 \prec S_2$ if

1. both system belong to the same system network: $S_1, S_2 \in SN$
2. there is at least a feature but not all of S_1 which belong to the entry condition of S_2

Systems are rarely if ever used in isolation. SF grammars often are vast networks of interconnected systems defined as follows.

Definition 6.8.4 (System Network)**.** A *system network* $SN = (r, SS)$ is defined as a hierarchy within set of systems SS where the order is that of systemic delicacy where:

1. S_i is an arbitrary system within the hierarchy $S_i \in SS$
2. $r \in S_i$ is the unique root of the system network with empty precondition $precondition(r) = \varnothing$
3. $p_i = precondition(S_i)$ the entry condition of system S_i.
4. $\tau : f \times S_i \to S_j$ a transition function from a feature $f \in precondition(S_i)$ to a less delicate system $S_j, f \in choices(S_j)$. We say that $S_j \prec S_i$

subject to the following conditions:

1. $\forall x \in \cup\{P_i | \forall P_i \in SN\}, \exists y \in \cup\{choices(S_i) | \forall S_i \in SN\} : x = y$ every precondition value is among the choice values
2. $\forall x \in \cup\{P_i | \forall P_i \in SN\}$ there is a path π (i.e. a sequence of systems) such that $\tau(x, \pi) = r$ (ensuring the connectedness of entire systemic network and a unique root)
3. $\nexists x \in \cup\{P_i | \forall P_i \in SN\}$ and $\nexists \pi$ such that $\exists S_j = \tau(x, \pi)$ and that $S_j \in \pi \vee x \in values(S_j)$ (ensuring the system network is no cyclical)

Now you may ask a pertinent question: what is the basis on which is the systemic selection made? To answer it I must first introduce two types of constraints. First, The systems are interconnected with each other by a set of preselection (entry) conditions forming systemic networks (Definition 6.8.4). Second, is an aspect not always mentioned in the SFL literature, the systemic *realisation statements* which are shaping the context where the system is applied. These aspects are covered in Section 6.9 talking about execution of system networks.

The notation for writing system networks from (Halliday & Matthiessen 2013) uses colon (:) to symbolize entry condition leading to terms in systems, slash (/) for systemic contrast (disjunction) and ampersand (&) for systemic combination (conjunction). So a sample network will be written as follows:

(100) $\varnothing : i_1/i_2/i_3$

(101) $i_1 : i_4/i_5$

(102) $i_2 \ \& \ i_4 : i_6/i_7$

However in this thesis we need to account for the disjunction type and system name. So we adopt a slightly different notation of three slots separated by colon (:) where the first slot signifies the system name, second the set of system features and the third is the entry condition. Examples 103 to 105 show three systems definitions (without selection functions i.e. no realization statements).

(103) $S_1 : OR(i_1, i_2, i_3) : \varnothing$

(104) $S_2 : XOR(i_4, i_5) : OR(i_1)$

(105) $S_3 : XOR(i_6, i_7) : AND(i_2, i_4)$

The system network can be represented as a graph where each node is a system and edges represent precondition dependencies. All system features must be unique in the network i.e. $\forall S_1, S_2 \in SN : choice\_feature\_set(S_1) \cap choice\_feature\_set(S_2) = \varnothing$ and there must be no dependency loops in the system definitions.

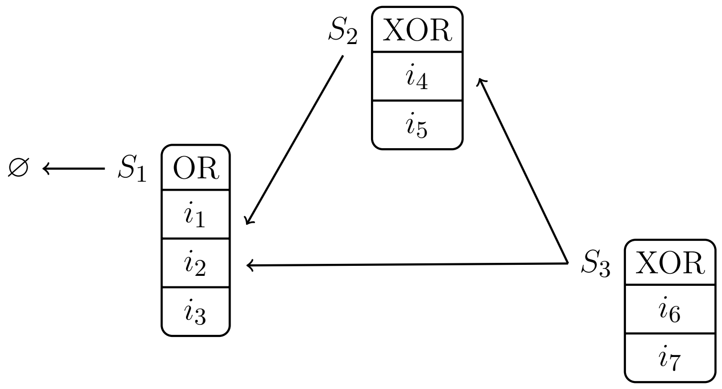


Fig. 6.14 Example System Network presented as graphs

In a systemic network SN where a system S_l depends on the choices in another system S_e (i.e. the preconditions of S_l are features of S_e) we call the S_e and *early(older) system* and the S_l a *late(younger) system*. This is just another way to refer to order systems according to their delicacy but applying this ordering to execution of systemic selection.

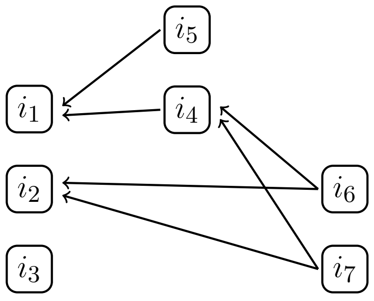
When the features are selected from systems within a network they form a path. It is often useful to check whether a set of arbitrary features belong to a *consistent* and *complete selection path*. Next I introduce a few concepts useful in addressing this task.

First a system network can be reduced to a graph of features called feature network (Definition 6.8.5 sometimes referred to as *maximal selection graph*) interconnected by system entry conditions.

Definition 6.8.5 (Feature Network). We call *Feature Network* $FN(N,E)$ a directed graph whose nodes N are the union of choice sets of the systems in the network and edges E connect choice features with the entry condition features. Formally it can be expressed as follows:

1. $N = \bigcup choices(\Sigma_i)$ where $\Sigma_i \in SN$ for $0 < isize(SN)$
2. $E = \{(f_m, f_n)\}$ where $f_m \in choices(\Sigma_i), f_n \in precondition(\Sigma_i)$

The Feature Network in fact is an expansion of the System Network. The former is a network of interconnected features while the latter a network of systems.



$$\begin{tikzcd}
 & & i_5 \arrow[dll] & \\
i_1 & & i_4 \arrow[ll] & \\
i_2 & & & i_6 \arrow[lll] \arrow[ul] \\
i_3 & & & i_7 \arrow[ulll] \arrow[uul]
\end{tikzcd}$$

Fig. 6.15 Example Feature Network

Definition 6.8.6 (Selection Path). A *Selection Path* $SP(N,E)$ is a connected subgraph of the Feature Network representing system network instantiation through choice making traversal.

Definition 6.8.7 (Complete Selection Path). A *Complete Selection Path* is a selection path starting from the network root and ending in one of the leafs.

We use terms related to age to underline order in which systems activated i.e. older systems must be chosen from before younger ones.

Definition 6.8.8 (System Network Instance). A *System Network Instance* SNI of a constituent node n is a directed graph representing the union of all Complete Selection Paths applicable to a constituent.

Let's come back to Figure 6.15. As you can notice this is a handy device for efficiently checking the path completeness (whether the path is from head to tail of a feature network), consistency with respect to the order of elements (whether such a path exits). There is one aspect that cannot be checked in feature network and it is the conjunctive entry conditions which require that both system networks precede any choice in the current one. In other words, a conjunctive entry states that two paths merging into one and they can only be checked in isolation as two distinct paths, which happen to share a common portion. This shorcoming will be dressed in future work.

In this section there were mentions to selection, instantiation and traversal processes but no specific definition were provided. Next, let's turn our attention towards the system network instantiation through traversal and selection.

6.9 Systemic Network Execution

Every node from a constituency graph is enriched with feature selections grammatically characterising it. This is an important stage in the parsing algorithm discussed in Section 7.5. The enrichment stage is in fact system network instantiation and ascription of complete selection paths to each constituent node .

Executing a system network is an incremental process that builds selection paths by making choices in the system networks. There are two ways to *instantiate* (or execute) a system network: either by *forward activation* or *backward induction* processes which both imply a different order of network traversal.

When it comes to traversing system networks and making choices there is a specific mechanism responsible for this instantiation process. The *choice makers* are selector functions associated to (some) systems. Selector functions implement realization statements corresponding to a system S_i and represents the instantiation mechanism turning the generic set of alternative choices into a concrete choice for a specific context.

Each node in a constituency graph carries features whose names and values are constrained to the set of systems defined in the grammar. In this sense, systems represent constraint definitions for what features may be used and what values those features can take. The algorithm has to evaluate these constraints in order to select the set of relevant features for a given constituent. Traversing system by system within the systemic network, with a known previously selected set of features and a given syntagmatic structure a selector function is executed to make the systemic choice.

Definition 6.9.1 (Selector Function)**.** A *selector function* $\sigma_{ctx} : S \to R$ is defined from a system S to a feature structure R within a given context ctx where:

1. the context $ctx = (G, fn)$ is a binary tuple of a constituency graph G and a focus node $fn \in G$ belonging to it

2. *preselection feature set* (PFS) is the already assigned set of features to the focus node $pfs = featureSet(fn)$

3. $size(R) \in \{0, 1\}$ meaning that there is either no choice made and an empty feature structure is returned or there is a choice made and a feature structure is returned with one feature bearing values from the system choice set

subject to the following condition:

1. if $size(R) = 1$ then for the only $f_i \in R$ it holds that $att(f_i) = name(S) \wedge val(f_i) \subset choices(S) \wedge val(f_i) \neq \varnothing$

If the PFS is an *OR set* then it requires that any of the features (at least one) must be in a Selection Path (Definition 6.8.6). If the PFS is an *AND set* then it requires that all of the features must be in a Selection Path.

6.9.1 Forward Activation

Forward activation is a process that enables systems to be executed (chosen from by selection function) only after choices from an older system has been already added to a selection path. In other words the selection path is constructed from older to younger systems/features.

We say that a system S_y *activates* another system S_o if and only if $\forall S_o, S_y : S_o < S_y, precondition(S_y) \cap choices(S_o) \neq \varnothing$. Activation process is the process that ensures advancement from an older to a younger system. This implies checking and ensuring entry condition is satisfied and executing the selection function. If the entry condition of the younger system is simple then the choice in the old system suffices, however if the entry condition is a complex conjunction, then first the older sibling systems have to be selected from before entering the younger one.

Algorithm 1: Forward Activation Algorithm

```
input : sp (current selection path), sn (system network), node (constituent),
        cg (constituency graph)
def forward_activate(sp, sn, node, cg):
    for system in sn systems activated by the last sp feature:
        get choice set by executing system selection function (given system, node,
          cg)
        append sp by the choice set
    if sp has changed:
        forward_activate (updated sp, sn, node, cg)
```

Algorithm 1 outlines how the forward activation is executed recursively. The systems that are active at a particular moment of the depend on the configuration of the *selection_path*. *activated_systems* function returns a set of systems from the system network whose preconditions are satisfied and their choices are not in the selection path (or the system has not yet been executed) $\forall S \in SN : precondition(S) \subset selection\_path$, $choice\_set(S) \cap selection\_path = \varnothing$.

For each activated system, its selector function is executed returning a selection set. The result selection is is used to extend the the selection_path thus potentially fulfilling preconditions of younger systems. If the path has been changed then the same procedure is applied recursively to the updated path until no more changes are done to the selection_path.

6.9.2 Backwards Induction

Backwards induction is a process opposite to forward activation. If a system is executed yielding a selection set then the preconditions of this system are induced as valid selections in the older systems defining those precondition features, and so on until a system is reached with no preconditions.

Algorithm 2: Naive Backwards induction

```
input : sp (current selection path), sn (system network), node (constituent),
        cg (constituency graph)
def backwards_induction_naive(sp, sn, node, cg):
    for system in sn systems preconditioning selection sp features:
        get choice set by executing system selection function (given system, node,
         cg)
        for induced_system in dependecy_chain(act_sys,sn):
            choice_set.add( precondition_set(induced_sys) )
        selection_path += create_selection_path_from(choice_set)
    return selection_path
```

The naive approach to is represented in Algorithm 2 which executes the selection functions of leaf systems and the yielded selections induce choices in the older systems through the precondition chain down to the oldest systems of the network.

So for example if SYNTACTIC-TYPE system in Figure 6.16 is executed and yields *verbal-marker* feature then the Algorithm 2 will add to the selection path the chain $negative \rightarrow interpersonal \rightarrow syntactic \rightarrow verbal - marker$.

This approach works very well in classification networks or networks covering a concise vocabulary such as determiners or pronouns. Such network has selection functions on the leaf systems only. However if in the middle of the selection path there are systems with selection functions then the there may exist a conflict between what is induced through precondition of younger systems and what is yielded by the selection function.

In fact confronting the preconditions with selection function is a good technique to verify whether the SN is well constructed. Following the previous example let's imagine that INTERPERSONAL-TYPE system has it's own selection function and it yields the *morphological* feature same time when the *verbal-marker* is selected in the SYNTACTIC-TYPE. Since the precondition of the latter system is the selection of *syntactic* feature, then we have a mismatch in either the way systems are constructed and the precondition of the latter system needs to be changed or the selection function is poorly implemented in the former system.

The Algorithm 3 implements the verification of whether the induced features match those from the selection function.

Algorithm 3: Backwards Induction with verification mechanism

```
def backwards_induction_verified(list_of_leafs, sn, constituent, mcg):
    for act_sys in list_of_leafs:
        choice_set = execute_selection_function(act_sys)
        if $choice\_set \neq \varnothing$:
            induced_system_set = find_dependent(act_sys,sn)
        for induced_sys in induced_system_set:
            ind_choice_set = selection_function(induced_sys)
            minimal_valid_set = precondition_set(induced_sys) $\cap$
              choice_set(act_sys)
            if $minimal\_valid\_set \subseteq ind\_choice\_set$:
                selection_path += create_selection_path_from(choice_set)
            else:
                raise Execption: The precondition set different from selection
                  function result
        backwards_induction_verified(induced_system_set, sn, constituent,
          mcg)
    return selection_path
```

It is a recursive algorithm that executes a system S_1, and also the systems $S_2..S_n$ which S_1 depends on an then verifies if $\forall S_i \in dependent\_systems(S_1) : precondition\_set(S_1) \cap choice\_set(S_i) \subseteq selection\_function(S_i)$

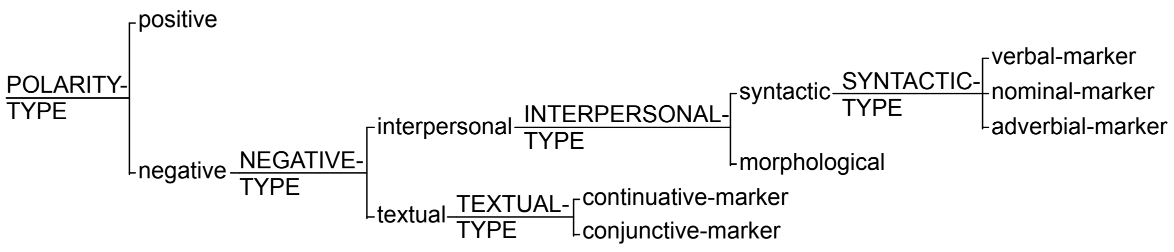


Fig. 6.16 Polarity System

Take for instance the POLARITY system in Figure 6.16. Its default selection is *positive* feature unless there is a negative marker. So we must asses NEGATIVE-TYPE system to resolve POLARITY system. But NEGATIVE-TYPE also must be postponed because we do not know if there is a negative marker unless we run tests for each marker type (i.e. presence of a "no" particle, negative subject or adjunct etc.). So we postpone selection decision and activate further the INTERPERSONAL-TYPE and TEXTUAL-TYPE systems and base the assessment on the selections yielded by the latter two systems. The same story is with INTERPERSONAL-TYPE which can make selections based on what SYNTACTIC-TYPE system yields. If SYNTACTIC-TYPE and TEXTUAL-TYPE systems yield no selection then we return recursively to INTERPERSONAL-TYPE and to NEGATIVE-TYPE and yield no selection in those systems as well. However if, for instance, *verbal-marker* is detected in the clause then the *syntactic* feature is yielded by the INTERPERSONAL-TYPE and *interpersonal* by the NEGATIVE-TYPE and thus *negative* is yielded by the POLARITY-TYPE.

Moreover the negative markers can be of various types and more than one can occur simultaneously without any interdependence between them so the algorithm needs to check presence of every type of negative marker i.e. verbal, nominal adverbial, conjunctive and continuative markers.

That being said the intermediary systems and features i.e. interpersonal, textual, syntactic, morphological are there for the classification purpose only and do not carry any particular algorithmic value making the network from Figure 6.16 reducible to the one in Figure 6.17.

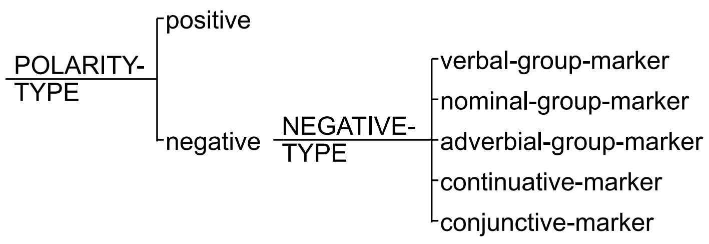


Fig. 6.17 Condensed Polarity System

Execution of system networks is subject to constituent *enrichment phase* of the parsing algorithm. Reducing the POLARITY network to the one in Figure 6.17 would lead to loss of information which may be relevant for choice-making in other systems (e.g. MODALITY) so it is useful to expand the selection set with dependent features to achieve feature rich constituents.

6.10 Discussion

This chapter describes the elemental data structure and the kinds of operations that current implementation applies to generate the SFG parse structures. It lays down the foundations for next chapter which focuses on the parsing pipeline and algorithms.

A central theme covered here are the graphs and graph patterns. They play the key role in identifying grammatical features in dependency and constituency structures. They are also excellent candidate for expressing *systemic realization rules*.

Robin Fawcett recurrently emphasises the role of realization rules in the composition of system networks. He often stresses "no system networks without realization rules". They are important because they formally express ways in which a feature is identified or realised. It is the *instantiation* process that in Halliday's words "is the relation between a semiotic system and the *observable* events or 'acts' of meaning" (Halliday 2003b: emphasis added). The realisation rules for a systemic feature are the statement of operations through which that feature contributes to the structural configuration (that is being either generated or recognised) (Fawcett 2000: p.86).

It is not easy however for linguists and grammarians to provide such statements for the systemic features. Doing so means an explicit formalisation of grammar on top of charting the systemic composition and dependencies which is already a challenging task in its own. The realisation rules most of the time remain in the minds of the interpreters who can recognise a feature when it occurs. Adding the formal specification of the realisation rule requires tools for consistency checking with respect to the rest of the grammar and large corpus query tool to test various rule hypotheses.

Moreover the expression of rules is proposed in terms of atomic operations such as lexify, preselect, insert, order, etc. Which may not always be fully transparent to the grammarian. Expressing realization rules as operations contextualised in fragments of parse structure is a promising way to ease the grammar authoring process. They could then be used directly by the parser to recognise such structures making the corpus annotation and grammar construction an in-parallel evolving process.

The data structures and operations described in this chapter can be a suitable approach to address the problem of missing realisation rules from the system networks. To do so however requires creation of a system network authoring tool (such as the one available in UAM Corpus Tool (O'Donnell 2008b)) which besides systemic network editor should contain also a graph pattern editor allowing association of graph patterns to systemic features and .

In current parser the pattern graphs are represented as compositions of Python dictionaries and lists such as the one below.

```
{
    NODES: {
        "cl": [
            {C_TYPE: 'clause',
             VOICE: ACTIVE},
            {CONFIGURATION: ['two-role-action', ['Ag', 'Ra', 'Cre']], }],
        'pred': [
            {C_TYPE: [PREDICATOR, PREDICATOR_FINITE], },
            {VERB_TYPE: "main", PROCESS_TYPE: 'two-role-action'} ],
        'subj': [
            {C_TYPE: SUBJECT, },
            {PARTICIPANT_ROLE: 'Ag'}],
        'compl1': [
            {C_TYPE: [COMPLEMENT, COMPLEMENT_DATIVE], },
            {PARTICIPANT_ROLE: 'Ra'}],
        'compl2': [
            {C_TYPE: [COMPLEMENT, COMPLEMENT_ADJUNCT, ], },
            {PARTICIPANT_ROLE: 'Cre'}],
    },
    EDGES: [
    ['cl', 'pred', None],
    ['cl', 'subj', None],
    ['cl', 'compl1', None],
    ['cl', 'compl2', None],]
}
```

This Python dictionary contains two top keys: NODES defined as with node identifiers each associated with a set of systemic features and EDGES defined as a list with three tuples of source, target and eventually a dictionary of features. The nodes contain a list of two dictionaries. The first dictionary enlists the features that the backbone structure should already carry, and against which the pattern matching is performed. The second dictionary contains the set of features that the node shall receive in case of a successful match of the entire pattern.

Writing such structures is cumbersome and requires in depth knowledge of the parser and employed system networks therefore the need for an editor is even higher. Unfortunately building such an editor is out of the scope of the current work and is among the priorities in the future developments just as switching to better technology for working with graphs such as Semantic Web suite of tools. This and other future work are described in the Section 10.4.

In the next chapter I describe the parsing pipeline and how each step is implemented starting from Stanford dependency graph all the way down to a rich constituency systemic functional parse structure.

Chapter 7

Mood parsing: the syntax

7.1 Algorithm overview

7.2 Preprocessing – canonicalization of DGs

The Stanford Parser applies various machine learning(ML) techniques to parsing. It's accuracy increased over time to $\approx 92\%$ for unlabelled attachments and $\approx 89\%$ for labeled ones (in the version 3.5.1). This section addresses known error classes of wrongly attached nodes or wrongly labelled edges and nodes.

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).

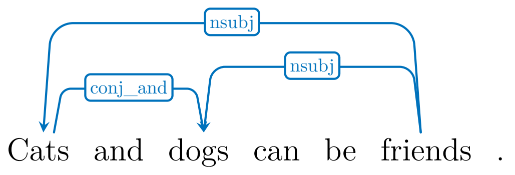
Beside stable errors, there are two other phenomena that are modified in the preprocessing phase: *copula* and *coordination*. They are not errors per se but simply an incompatibility between how Stanford parser represents them and how they need to be represented for processing by the current algorithm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

7.2.1 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 7.1 - 7.5.



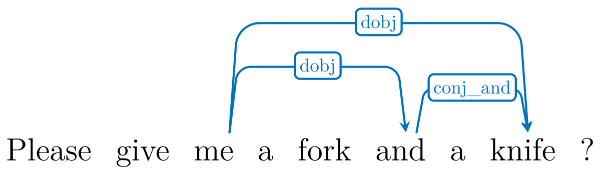
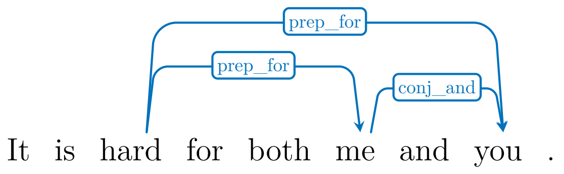


Fig. 7.2 Conjunction of noun objects



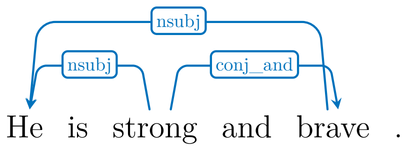


Fig. 7.4 Conjunction of copulatives sharing the subject

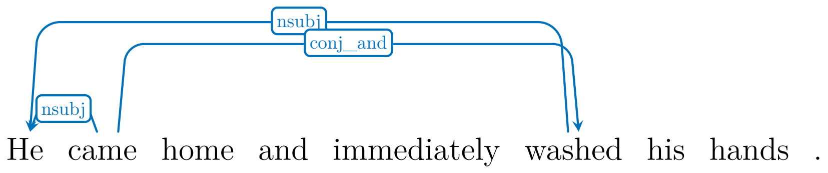


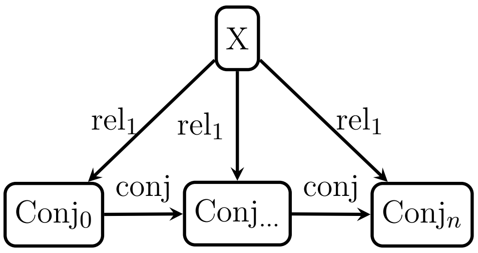
Fig. 7.5 Conjunction of verbs sharing the same subject

The main reason these extra edges need to be removed is to avoid double traversal of the same node via different paths (which will lead to creation of two constituents). For example, if multiple subject relations occur in the DG then multiple subject are going to be instantiated in CG which is not intended in the grammar. Rather only one complex unit needs to be created with the subject role composed of two noun phrases (see discussion in the Section 3.4.6).

The straight forward way to fix it this problem is removing functional edges to/from each conjunct except the first one. There are two generic patterns in figures 7.6 and

7.8 correspondingly with incoming and outgoing edges that are transformed into the forms depicted in 7.7 and 7.9.

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 7.6 and exemplified in Figures 7.1 - 7.3. In SFG terms it corresponds to cases when the functional element of a parent constituent is filled by a complex unit below.



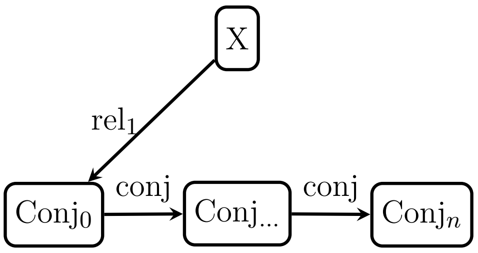
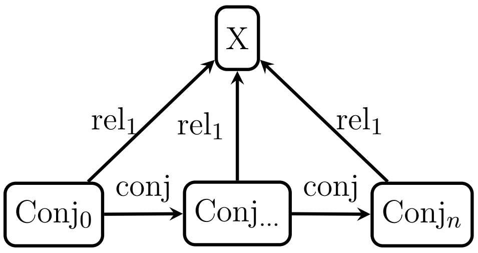


Fig. 7.7 Conjuncted elements with incoming loosely connected dependencies

The second is the pattern of conjuncts with *outgoing dependency relations* depicted in Figure 7.8. 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 5 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 7.4 - 7.5.



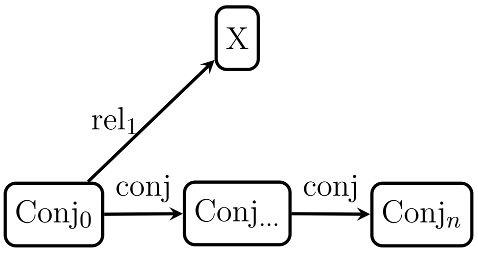


Fig. 7.9 Conjuncted elements with outgoing loosely connected dependencies

7.2.2 Transforming copulas into verb centred clauses

In Stanford dependency grammar *copular verbs* are treated as dependants of their complements (see Figures 7.10 and 7.11) 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 3.5.1 and 3.5.2.

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.

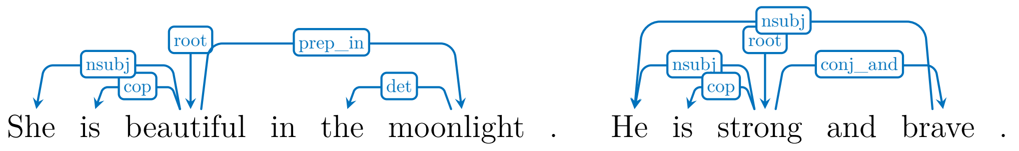


Fig. 7.11 Conjunction of copulatives sharing the subject

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 7.1 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 7.1 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 7.1, 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 7.12 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 7.13 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.

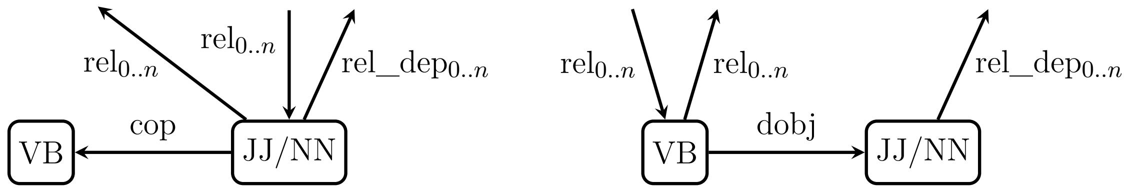


Fig. 7.13 Generic pattern for copulas after the transformation (the same as non-copular verbs).

In case of conjuncted copulas like in the example in Figure 7.11 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.

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

The Figure 7.14 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 7.14 receives a different analysis as it is represented in Table 7.2.

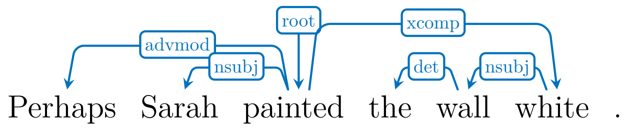


Fig. 7.14 Dependency parse for clausal complement with adjectival predicate

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

Table 7.2 SFG analysis with attributive adjectival complement

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 7.14 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.

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 7.15 to form in Figure 7.16. 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.

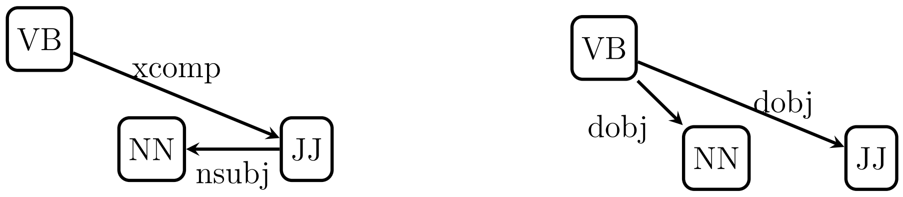


Fig. 7.16 Adjectival clausal complement as secondary direct object

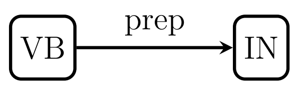
7.3 Preprocessing – error correction

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.

7.3.1 *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 7.17) into *prt* (figure 7.18) if the preposition node has no children e.g. *pobj*.



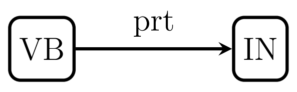
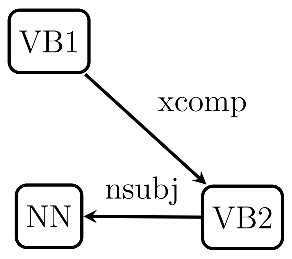


Fig. 7.18 Corrected relation to free preposition as verbal particle

7.3.2 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 7.2.3. 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 7.19 - 7.20.



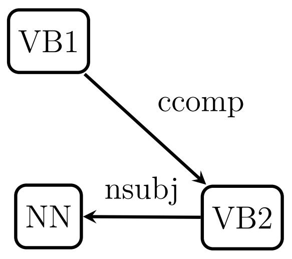


Fig. 7.20 Corrected clausal complement

7.3.3 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 7.3.

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 7.3 Mapping lexical forms of auxiliaries to their POS

7.3.4 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 7.2.2 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.

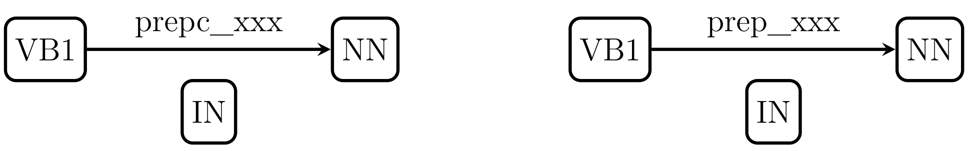


Fig. 7.22 Corrected prepositional phrase

7.3.5 Mislabelled infinitives

In English base form of the verb often coincides with present simple form (non 3sp(*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.

TO
aux
VBP
TO
aux
VB

Fig. 7.24 Correct infinitive

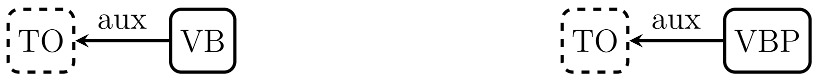


Fig. 7.26 Correct present simple

7.3.6 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 3sp(*rd*) person present simple).

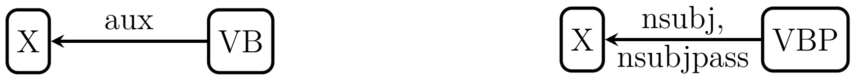


Fig. 7.28 Corrected attributive verb

7.3.7 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 106.Latter in (Marneffe et al. 2014) both relations have been merged into the *vmod*.

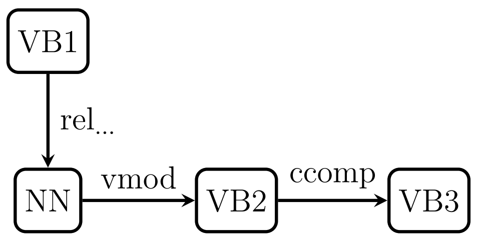
(106) Tell the boy playing the piano that he is good.

Clauses such as "(that) he is good" following immediately after the qualifier clause ("playing the piano" in the example 106) 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 106 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 7.29 which is transformed by the algorithm into the form represented in Figure 7.30

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



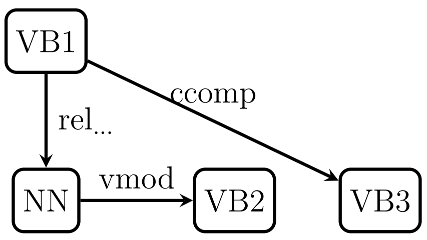


Fig. 7.30 Clausal complement attached to the main clause

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.

7.3.8 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.

(107) Bill moved those beyond the counter.

(108) Put that in our plan.

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

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

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

(112) 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 109 and 110 and cases such as 111 and 112 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.

In examples 107 and 108, attaching the prepositional phrase to the demonstratives (depicted in Figure 7.31) is incorrect. It should be attached to the verb (like in the figure 7.32) 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.

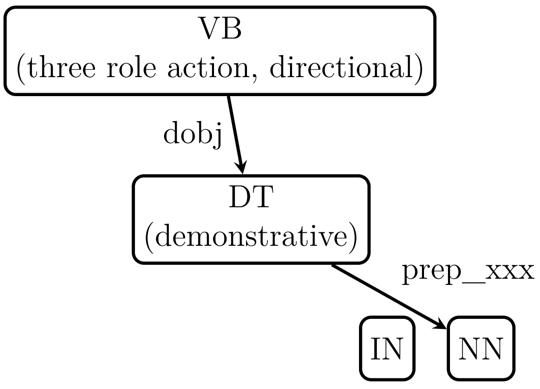
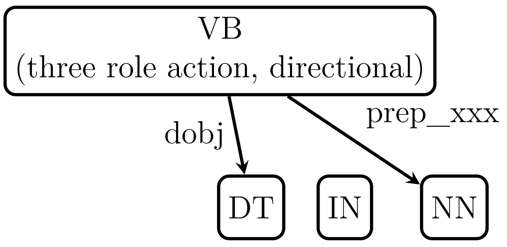


Fig. 7.32 Prepositional Phrase attached to the verb with a demonstrative pronoun in between



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

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

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.

7.3.9 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 113–118 (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 117–118.

(113) Joe(,) his name is.

(114) Relaxation(,) you call it.

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

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

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

(118) 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.

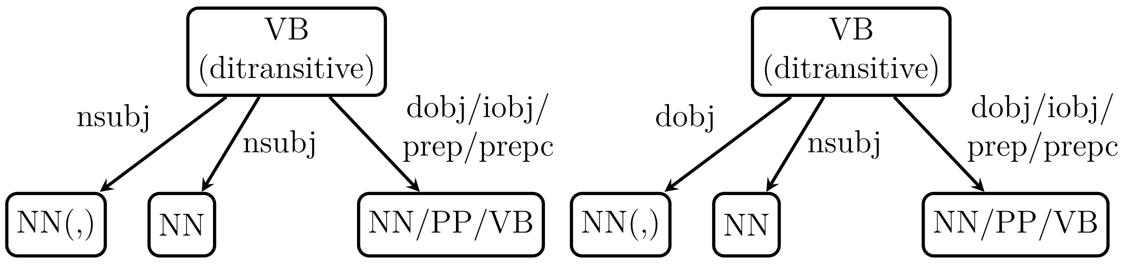


Fig. 7.34 Topicalized Direct Object – moved to pre-subject position

The algorithm is looking for the cases of multiple subjects (represented in figure 7.33) and gives priority to the one that is closest to the verb. The other one is relabelled as a complement (Figure 7.34). The rule is generalized in the algorithm for multiple subjects even if so far only cases of two subjects have been observed.

7.3.10 Misinterpreted clausal complement of the auxiliary verb in interrogative clauses

Sometimes the auxiliary verb in the interrogative clauses (examples 119 and 120) 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.

(119) Do you walk alone?.

(120) Has Jane fed the cat?.

The algorithm searched for the pattern depicted in Figure 7.35 and transforms it into the form Figure 7.36.

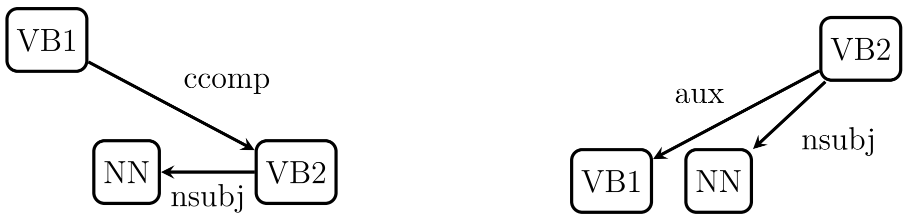


Fig. 7.36 Corrected clausal complement

7.4 Creation of systemic constituency graph from dependency graph

This section describes how the systemic constituency structure is generated from the dependency graph. Even if at first sight they appear isomorphic, DGs and CGs differ in their structure. The CG is created in two phases as presented in the Algorithm 4. The first phase generates an incomplete CG through a top-down breadth-first traversal of a DG. The second phase complements the first one ensuring creation of all CG constituents through a bottom-up DG traversal.

Algorithm 4: Constituency graph creation

```
   input  : dg (the dependency graph), rule table
   output : cg (the constituency graph)
1  begin
2      create the bootstrap 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.

7.4.1 Dependency nature and implication on head creation

As explained in Section 4.6, the nature of dependency relations is different 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 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.

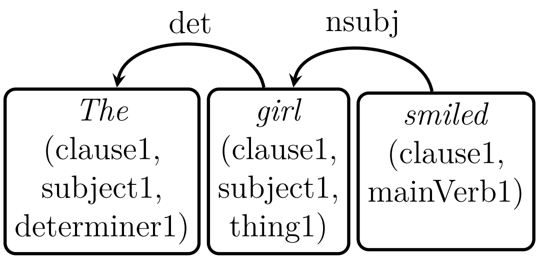
In current approach, the top-down perspective considers the DG node as representing the upper most rank. The bottom-up perspective considers the DG nodes as representing the lower most rank. Thus the generation algorithm first traverses the graph in a top-down order and generating the units as appropriate and then bottom up in order to create their heads.

The result of the top-down phase is a constituency graph without head nodes. Therefore the bottom-up phase is performed by traversing th constituency graph and not on the dependency graph. The traversal task is to locally resolve which dependency nodes form the syntactic head. The local resolution is possible because of the tight coupling between the dependecy and constituency graphs established in the top-down phase. It is explained in the section below.

7.4.2 Tight coupling of dependency and constituency graphs

At the creation stage, the CG is tightly coupled with the original DG. This allows navigating easily from one graph to the other one via references stored within the nodes of each of them. I say that a graph node is *aware* of it's 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 7.37). 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 7.38). This way the positioning information is available bidirectionally about CG an DG structures.


det
nsubj
The (clause1, subject1, determiner1)
girl (clause1, subject1, thing1)
smiled (clause1, mainVerb1)

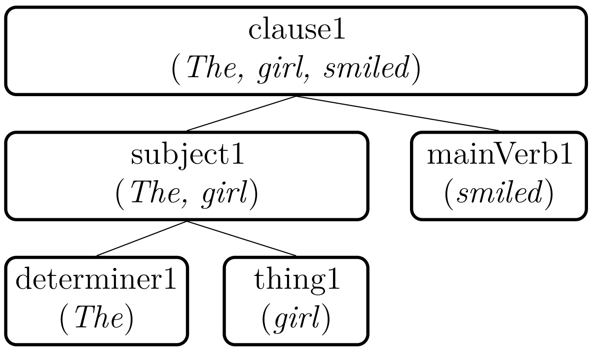


Fig. 7.38 Dependency aware CG nodes

The Figure 7.37 depicts a dependency graph. Each node has a stack of ids corresponding to constituents in the CG (Figure 7.38). Conversely, in Figure 7.38, depicts a constituency graph where the constituent nodes carry a set of tokens corresponding to DG nodes. This way the DG in Figure 7.37 and the CG in 7.38 are aware of each other.

The 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.

7.4.3 Mapping Rule Tables

Constituency Graphs are created through a top-down breadth-first walk of dependency graph. During the traversal each visited edge triggers execution of a *creational operation* on the growing CG on the side. To know what operation to execute a rule table is used where the edge type, head and tail nodes are mapped to a creational 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 7.4. It can be regarded as an attribute value matrix or a Python dictionary where the left column titled: key, contains a unique *dependency graph context* serving as a rule trigger; while the left side column named: value, contains the operation to be executed within the given context.

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 3.2 and 3.3. 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, which account for the verb semantics, these options ideally are reduced to one.

	Key	**Value**	
		Operation	***Parameter***
1	nsubj	new_constituent	Subject
2	csubj	new_clause_constituent	Subject
3	prepc	new_clause_constituent	Complement, Adjunct
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 7.4 Rule table example mapping (specific or generic) dependency context to constructive operation

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 (on 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 (on the table 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 (on the 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 (table-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.

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

7.4.4 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 table with mapping rules. The CG is created as ap parallel structure through the process of breadth-first traversal on DG edges as described in Algorithm 5.

Algorithm 5: Top-down CG creation

```
    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 the suitable rule for the current edge in the rule table
 6          operation ← get operation from the rule
 7          element type ← if any get the parameter from the rule
            // as cg and dg nodes are aware of each other we can
               navigate between them
 8          constituency stack ← the constituency stack from the tail node of the
              current dg edge
 9          cg pointer ← the top 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 the CG. Also, the root node is made aware of the root node in DG via the mechanism described in Section 7.4.2.

Then the DG is traversed on its the edges in BFS order starting from the root node. As each DG edge is visited an operation is chosen based on the edge type and nodes POS along with computation of an additional set of parameters Lines 7 - 10; after which 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. Note that the head nodes are not created in current but the next phase.

The Line 5 of the algorithm is responsible for looking up and selecting the operation from the rule table as described in Algorithm 6. It is based on two lookups based on the rule indexes: one contextualised to the edge relation and its nodes and another one generic 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.

Algorithm 6: Operation selection in the mapping rule table based on the edge type

```
input  : rule table, edge
output : rule
1  begin
2      generic key ← the simplified dependency relation of 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 ← tail POS + generic key + head POS
6      if specific key index in rule table:
7          rule ← rule indexed with specific key from rule table
8      elif generic key index in rule table:
9          rule ← rule indexed with generic key from rule table
10     else:
11         rule ← None
12     return rule
13 end
```

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 7.4.3: *extend current* and *new (sibling) constituent*. Next I present the pseudo-code for each of them. Note that these operations do not return anything because their effect is on the input cg and dg.

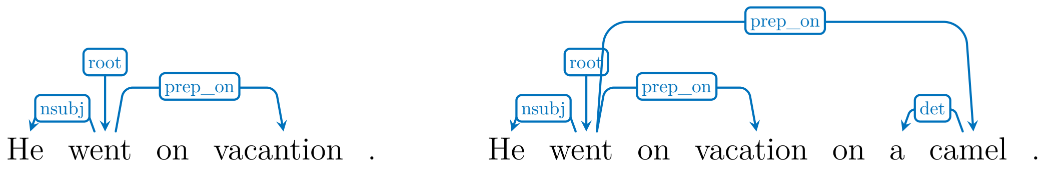


Fig. 7.39 Challenging free nodes

Extend constituent. Algorithm 7 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 concomitantly making them aware of each other.

Algorithm 7: 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 $prep \lor conj$ 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 4.5.

Figure 7.39 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 8 is a function that extends CG with a newly created constituent object (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 7 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 3.

Algorithm 8: Creating new child constituent

```
   input  : cg pointer, children, element type, edge, dg, cg
 1 begin
 2     node ← new Constituent
 3     node type ← element type
 4     add to cg the edge (cg pointer, node)
       // invoking the Algorithm 7
 5     extend cg pointer with children of edge
 6 end
```

A variation of the *create new* is *create sibling* outlined in the Algorithm 9. 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 9: 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 8
 3     create new constituent to an updated cg pointer
 4 end
```

7.4.5 Bottom up traversal phase

Chapter 3 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.

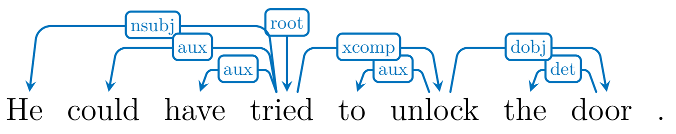


Fig. 7.40 The dependency graph before the first phase

The Figure 7.41 depicts and example CG generated in the 1sp(*st*) phase with dotted lines representing places of the missing constituents.

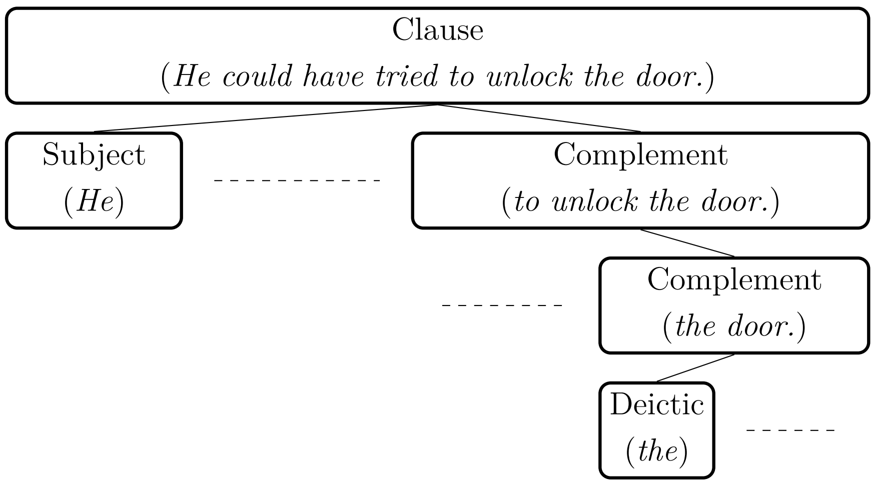


Fig. 7.41 Constituency graph after the top down traversal missing the head nodes

The missing constituents are the syntactic heads for all units. The clause, besides the Main Verb, also misses the Finite, Auxiliary elements. Determining these functions strongly depends on the place within a unit and syntagmatic order in which units occur. As the first phase is performed as graph traversal, the order dimension is not available so they have to be created in the second phase.

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 7.5. 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 disjunction 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 7.5 Constraints for unit class assignment

The Algorithm 10 traverses the CG bottom-up (not the DG) with *post-order depth-first* order (line 2) during which every visited constituent node is assigned a unit class and missing child constituents are created.

Because the CG and DG are tight coupled which means that each CG node spans over a set of DG nodes then traversing CG is equal to traversing groups of DG nodes at each step. This creates a focused mini context suitable for resolving the unit class and missing elements.

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 15 assign classes according to conditions stated in Table 7.5.

Clause classes are assigned to units started by a non-modal verb. This rule corresponds to the one main verb per clause principle discussed in Section 3.5.1. This approach however does not take into consideration elliptic clauses and they need additional resolution. This short coming should be considered in the future by an additional *ellipsis resolution mechanism*, similar to the one for *null elements* described in Chapter 5.

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 sentence token 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 section.

7.5 Feature enrichment process

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 7.6 associates a list of triggers to a list of systemic networks. Table 7.7 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 6.7 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 10: 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 ← POS of the dg entry node of node
4          element type ← currently assigned element of node
5          children ← get assigned elements to the children of node
           // assigning classes
6          if head POS in main verb POSs:
7              assign node Clause class
8          elif head POS in nominal POSs ∧ element type in prepositional element
             types ∧ Marker in children:
9              assign node Prepositional Group class
10         elif head POS in nominal POSs ∧ element type in nominal elemement
             types:
11             assign node Nominal Group class
12         elif head POS in adverbial POSs ∧ element type in adverbial element
             type:
13             assign node Adverbial Group class
14         elif head POS in adjectival POSs ∧ element type in adjectival element
             type:
15             assign node Adjectival Group class
           // creating the rest of the units
16         if node is not a leaf:
17             if node is a Clause:
18                 create the Clause elements for current node
19             else:
20                 create the head for the current node
21 end
```

The Algorithm 11 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 7.7 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 5). The line 5 executes the selection function returning a set of choices. Then the last line assigns the choices to the node in focus.

Algorithm 11: 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
```

Key	*System Activation Order*
clause	POLARITY,VOICE, AGENCY, MOOD TYPE, INDICATIVE TYPE, DEICTICITY, TENSE, MODALITY
nomi-nal	PERSON, ANIMACY, GENDER, NUMBER
deictic	DETERMINATION
pre-deictic	DETERMINATION
thing	PERSON, ANIMACY, GENDER, NUMBER
posses-sor	PERSON, ANIMACY, GENDER, NUMBER

Table 7.6 System activation table depending on unit class or element type

Currently implemented networks are outlined in Table 7.6 as associations to the node class or function. The left column represents activation triggers and the right column represents an ordered list of systems.

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 7.7 Mapping systemic networks to selection functions

Algorithm 12: 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
```

The *dictionary_lookup* function is a type of *naive backwards induction* (Algorithm 2). It checks whether the word(s) of the *mcg_node* are in a dictionary with preselected features. It is the only function that requires a parameter which is the lookup dictionary. An example of dictionary is presented in the Table 7.8

7.6 Discussion

* replace hard coded selector functions with constraint checking

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

Table 7.8 Dictionary example for the DEICTICITY system network

assign feature with a certain probability and allow assignments of mutually exclusive features; this would require the next step of more general constraint checking employing many more other sources such as semantic, phonetics, situation etc.

use of logical notations (OR, XOR, AND sets)

Chapter 8

Transitivity parsing: the semantics

Semantic Role Labeling (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.

8.1 Creation of 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.

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 5.2. 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 labeling 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 121 123 where *Subj* is a pronominal subject placeholder.

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

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

(123) 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 5.3 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.

8.1.1 The PRO and NP-Trance Subjects

The *xcomp* relation in DC can be encoded as an CG pattern graph (Figure 8.1) 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 5.2.4). They shall receive a the PRO subject constituent (governed or not) by the parent clause subject.

The generalization 5.3.2 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 8.2 and respectively 8.3. Please note that in case of Figure 8.3 the prepositional complements do not affect subject control in any way since it specifies only the nominal complements this making it complementary to 8.2 with respect to prepositional complements.

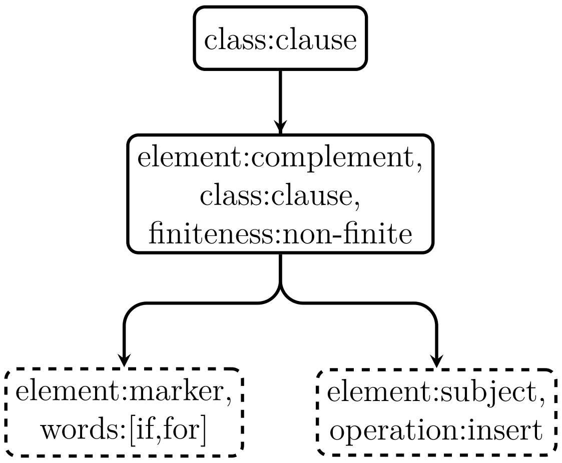


Fig. 8.1 CG pattern for detecting PRO subjects

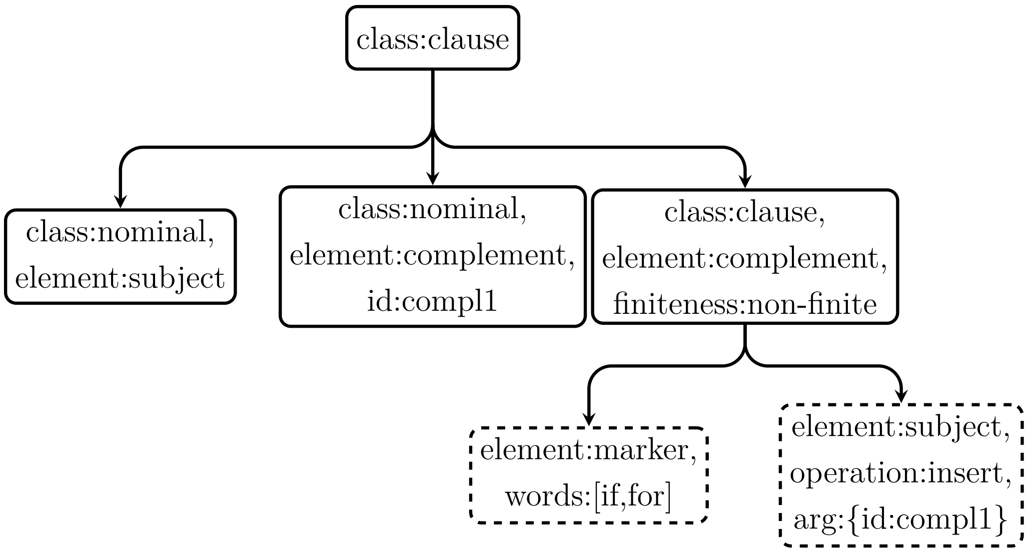


Fig. 8.2 CG pattern for obligatory object control in complement clauses

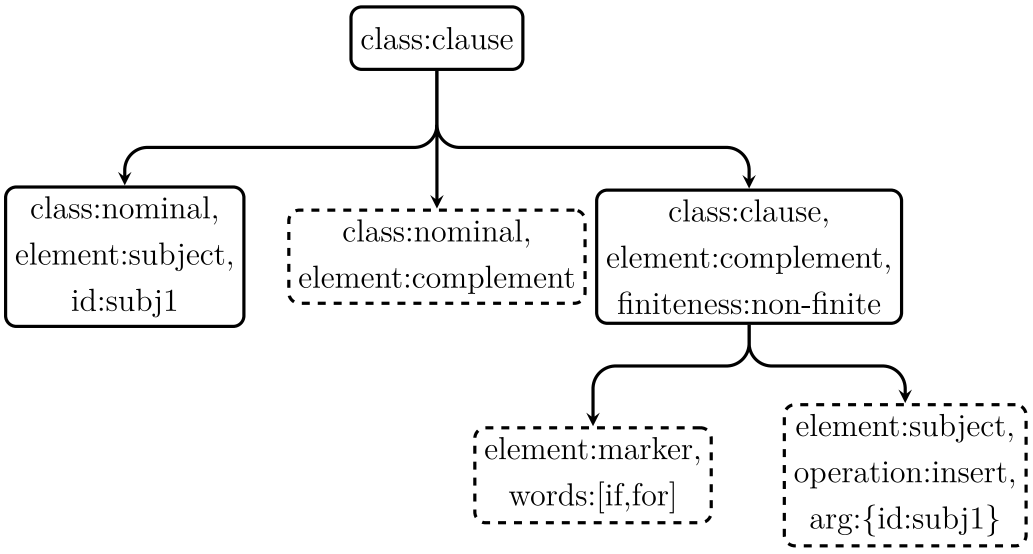

class:clause
class:nominal, element:subject, id:subj1
class:nominal, element:complement
class:clause, element:complement, finiteness:non-finite
element:marker, words:[if,for]
element:subject, operation:insert, arg:{id:subj1}

Fig. 8.3 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 8.2 and 8.3 are applicable.

According to generalization 5.2.9 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 5.2.9 is proposed the arbitrary interpretation.

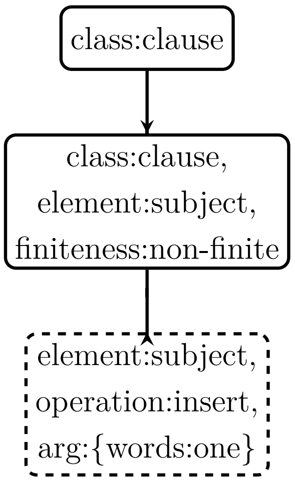

class:clause
class:clause, element:subject, finiteness:non-finite
element:subject, operation:insert, arg:{words:one}

Fig. 8.4 CG pattern for arbitrary control in subject clauses

The pattern for subject control in subject clause is represented in Figure 8.4. 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 63-66.

8.1.2 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 13: 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 14: 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 15: 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 configurationsdo
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 clausesdo
9          foreach config in clause configurationsdo
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
```

cuss the wh trace
eation algorithm

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.

8.2 Semantic enrichment stage

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 labeling*. 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 labeling is performed and why it is important to identify and create references to empty elements.

8.2.1 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 7.5.

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 8.1 presents a simplified sample of PTDB content.

verb form	**informal meaning**	**process type**	**configuration**
calcu-late	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 8.1 An example of records ins PTDB

8.2.2 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* ($1\mathsf{sp}(st)$), the Cardiff grammar *process type* ($6\mathsf{sp}(th)$) and the participant role *configuration* ($8\mathsf{sp}(th)$) columns. Note that column numbers correspond to the original PTDB structure.

After the transformation the PTDB column descriptions are as described in the Table 8.2.

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

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 8.2 The table structure of PTDB before and after the transformation

process type and the participant configuration. So they are good cues to be considered 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 8.5.

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.

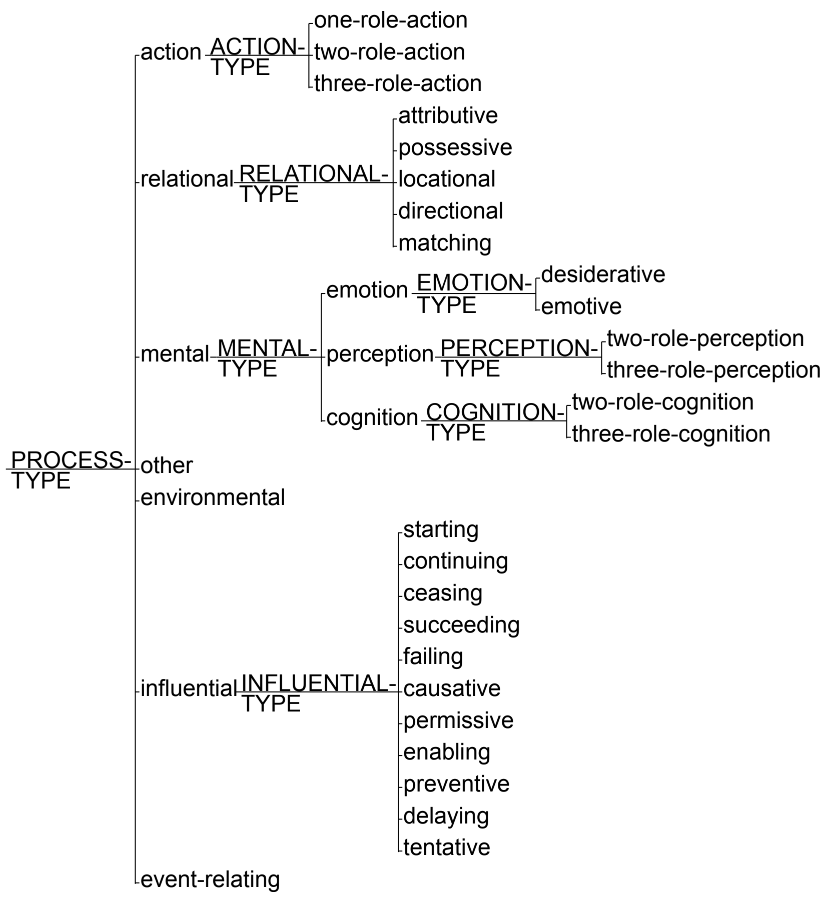


Fig. 8.5 Transitivity System in Cardiff Grammar

The Cardiff features column indicates the process type selected in the TRANSITIVITY system corresponding to one of the top levels depicted in Figure 8.5

8.2.3 Generation of the Configuration Graph Patterns

The configuration pattern graphs are used for CG enrichment executed through graph matching operation described in Section 6.6. 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 8.6 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.

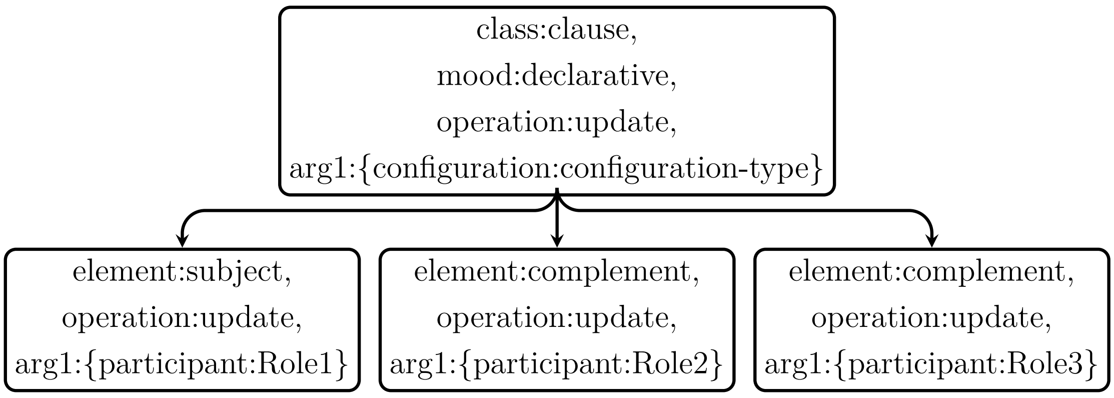


Fig. 8.6 Indicative mood and active voice configuration pattern with three participant roles

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 16 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 16: 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.

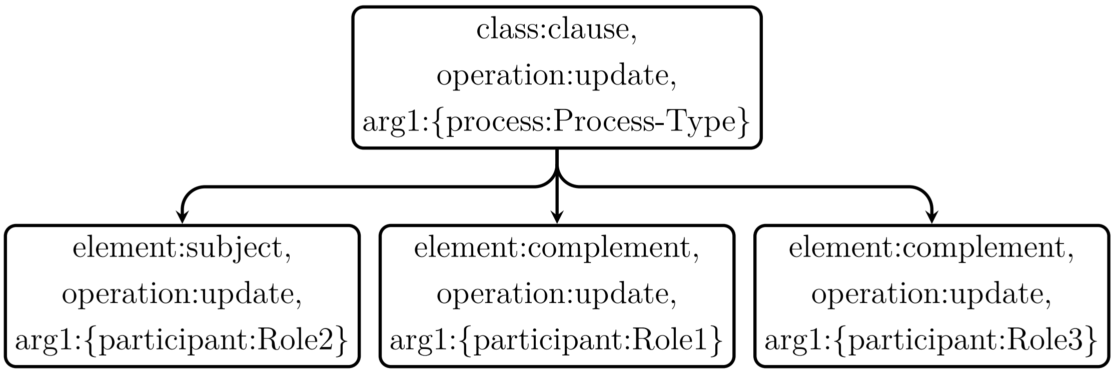


Fig. 8.7 Indicative mood and passive voice configuration pattern with three participant roles

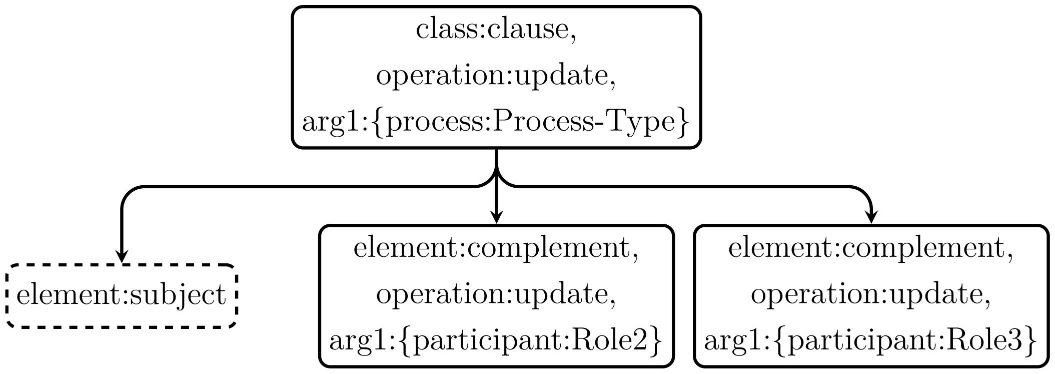


Fig. 8.8 Imperative mood configuration pattern with three participant roles

The indicative mood active voice pattern (depicted in Figure 8.6) 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 8.7. If the configuration accepts imperatives then also a subjectless pattern graph is generated with first role omitted as depicted in Figure 8.8.

Directional processes are rather a special case. Examples 124 to 126 are equally valid configurations. Example 127 is a generic representation highlighting the optionality of these participants.

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

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

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

(127) 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 8.3 below.

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

Table 8.3 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 8.1

Listing 8.1 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 8.4 Prepositional constraints on participant roles

8.2.4 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 17: Transitivity parsing

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

Algorithm 17 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 8.4) or (c) phrasal expressions such as "take a shower" which are explained in Section 8.2.1.

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 18) which were omitted in the pattern graph. These conditions may prevent the enrichment even if the pattern has been identified.

Algorithm 18: 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.

8.3 Discussion

This chapter describes how the semantic role labeling 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 labeling step employs the same pattern based enrichment mechanism as the one used in Section 7.5

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 9

The Empirical Evaluation

9.1 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 my own evaluation corpus. A part of it was created together with Ela Oren covering basic Sydney Mood constituency structure. Another part is the PhD dataset of Anke Shultz covering Cardiff Transitivity analysis. The current evaluation is based on these two datasets.

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 MCG 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 9.2) for each feature instead of single segment with multiple features (Figure 9.1).

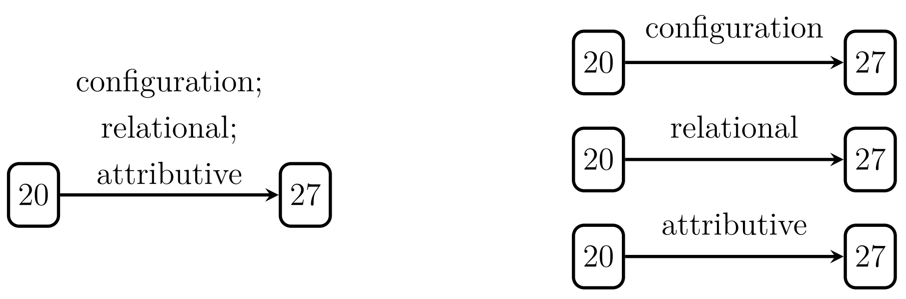


Fig. 9.2 A set of segments with single features

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.

9.2 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 9.3 they are subsumed in a cascade from the former to the latter as depicted in Figure 9.4.

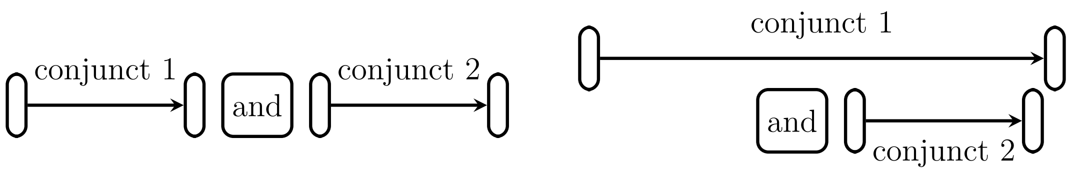


Fig. 9.4 Conjuncts annotated as subsumed segments

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 3.5.

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 ($d = \sqrt{\Delta start\mathsf{sp}(2) + \Delta end\mathsf{sp}(2)}$) to measure the difference between the segments matched with Gale-Shapley algorithm. Figure 9.5 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).

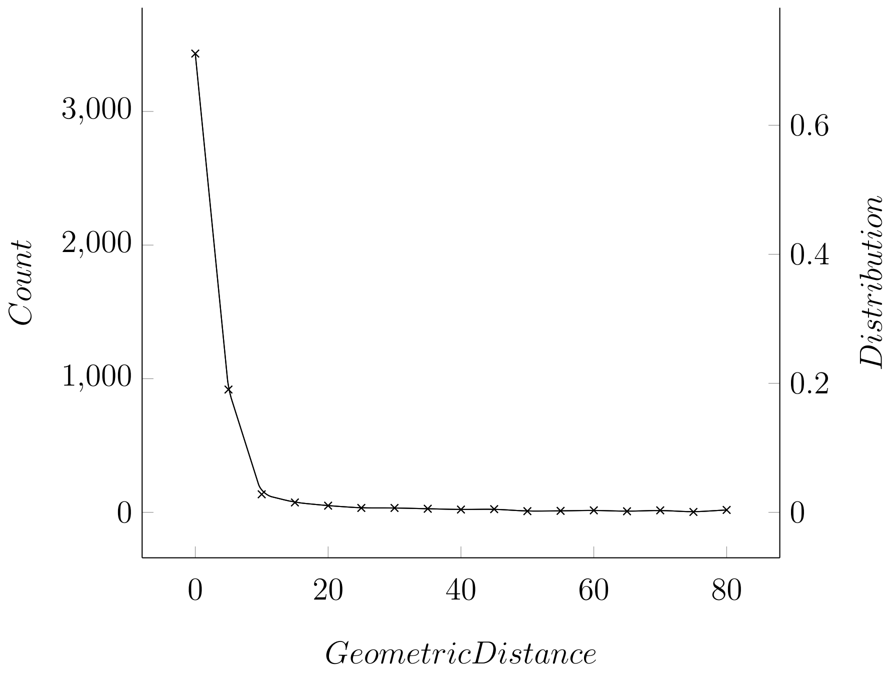


Fig. 9.5 Segment distance distribution

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 9.1 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 9.1 Rank system evaluation statistics

The clause elements are systematized in Interpersonal function system whose evaluation statistics are contained in the Table 9.2. 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 128. But when the finite is conflated with the Predicator as in Example 129 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.

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

(129) 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 9.2 Mood clause elements evaluation statistics

In Table 9.3 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 9.3 Evaluation statistics for group types

9.3 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 9.4 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 9.5 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.

9.4 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 10

Conclusions

why is it useful

make reference
to the software

10.1 The thesis summary

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 3 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 7 together with the mapping rules used in the process.

To make clear what are the basic ingredients and how the algorithms are cooked, Chapter 6 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 6. Then in Chapters 7 and 8 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 5 and then a concrete implementation described in Chapter 8.

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 9.4 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.

10.2 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.

10.3 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.

10.4 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.

10.4.1 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 3.5.1) 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 10.1 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 10.2 Cardiff sample analysis of a clause *embedded* into another

Check the sample analyses in Table 10.1 and 10.2. 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 10.1.

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 10.4.1 (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 10.4.1. 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 10.1 and 10.2 – but also of the modal assessment that includes modality, as demonstrated by the Examples 130 and 131.

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

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

Examples 130 and 131 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*.

10.4.2 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 3.3 and discussed in 3.5.

For instance, Nominal class structure proposed in Cardiff grammar (discussed in Section 3.5.3), 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 3.5.4) 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.

10.4.3 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.

10.4.4 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.

10.4.5 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.

10.4.6 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.

10.4.7 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.

10.4.8 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

10.5 Overall evaluations

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

The writer's perso
opinion on wha
has been discuss

10.6 A final word

half to one page r

[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>.

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.

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

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>.

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.

Cordella, L P, P Foggia, C Sansone & M Vento. 2001. An improved algorithm for matching large graphs. *3rd IAPRTC15 workshop on graphbased representations in pattern recognition* 219(2). 149–159. doi:10.1.1.101.5342. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.101.5342{&}rep=rep1{&}type=pdf>.

Cordella, L P, P Foggia, C Sansone & M Vento. 2004. A (sub)graph isomorphism algorithm for matching large graphs. doi:10.1109/TPAMI.2004.75. <http://www.ncbi.nlm.nih.gov/pubmed/15641723>.

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, 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. 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 (4th Edition).* Routledge 4th edn.

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

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.

Koeva, Svetla, Borislav Rizov, Ekaterina Dimitrova, Tarpomanova Tsvetana, Rositsa Dekova, Ivelina Stoyanova, Svetlozara Leseva, Hristina Kukova & Angel Genov. 2012. Bulgarian-english sentence-and clause-aligned corpus. In *Proceedings of the second workshop on annotation of corpora for research in the humanities (acrh-2)*, 51–62.

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.

Lee, Jinsoo, Wook-Shin Han, Romans Kasperovics & Jeong-Hoon Lee. 2013. An in-depth comparison of subgraph isomorphism algorithms in graph databases. *Proceedings of the 39th international conference on Very Large Data Bases* 133–144. doi:10.14778/2535568.2448946. <http://dl.acm.org/citation.cfm?id=2448936.2448946>.

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 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>.

Shang, Haichuan, Ying Zhang, Xuemin Lin & Jeffrey Xu Yu. 2008. Taming verification hardness: an efficient algorithm for testing subgraph isomorphism. *VLDB Endow* 1(1). 364–375.

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>.

Stevan Harnad. 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>.

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.

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.