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

February 2019

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

Declaration

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

Eugeniu Costetchi
February 2019

Acknowledgements

And I would like to acknowledge ...

Abstract

This is where you write your abstract ...

Table of contents

List of figures

List of tables

List of definitions

Chapter 1

Introduction

1.1 On artificial intelligence 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 (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 1950s in the context of automatic translation (Hutchins 1999) of Russian text into English and developed 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. Nevertheless there is still a long way to go through before machines excel in these narrowly scoped tasks and even longer before machines start using language in the ways human do.

1.2 Living in a technologically ubiquitous world

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 a formal, strict and rigorous manner via graphical user interfaces, command line terminals and programming languages. Advancements in *Natural Language Processing* (NLP) are a game changer in this domain. NLP starts to unlocks the information 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 as ours, where technology is ubiquitous and pervasive in almost all aspects of life, NLP becomes of great value and importance regardless of whether it materializes as a spell-checker, an intuitive recommender system, spam filters, (not so) clever machine translators, voice controlled cars, or intelligent assistants such as Siri, Alexa or Google Now.

Every time an assistant such Siri or Alexa is asked 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. The next challenge is to develop a technology that enables computers to interact with us in a language we understand rather then they.

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. In this section I present a few example application areas and use them to discuss tasks that need to be accomplished for NLP in such contexts. These examples underline the ever growing need for NLP putting into perspective the need of ever deeper and richer linguistic analysis across a broad range of domains and applications.

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 regardless of whether it is pizza or a smartphone. Popular sources for such inquiry are blogs, forums, reviews, social media, reports, news, company websites, etc. All of these 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 the sentence “The pizza was amazing but the waiter was awful!” and connect it to the following sentence “I love when it is topped with my favourite artichoke”, disambiguating the sentence so that it is clear that it is about pizza and not the waiter and so discover a topping preference.

NLP is heavily used in customer service in order to figure out what a 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 for increased responsiveness by providing immediate answers to simple needs and also decrease the load on 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 to the socio-cultural context where it occurs. Yet there is still much work to be done before the technology is capable to pertorm such complex levels of automatic analisys.

1.4 Linguistic framework

The present work is conducted under the premise that a theory of language is important and worth adopting. It is possible, in NLP, to reach considerable results even without adoption of such a framework. This is demonstrated by the advancements in (deep) machine learning. In current work the Systemic Functional (SF) theory of language is adopted because of its versatility to account for the complexity and phenomenological diversity of human language providing descriptions along multiple semiotic dimensions. This explanation is extended further in this section emphasizing SFL strengths.

Any meaningful 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 the Copenhagen School of linguistics and by European linguists from the Prague Linguistic Circle. Halliday's 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)

Embracing the *organon 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. Thus, 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 the *ideational metafunction*. Finally the two weave together into a coherent discourse flow whose mechanisms are characterised through the *textual metafunction*.

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 *syntagmatic* (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 as Butler (2003a,b) has extensively addressed. 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.

A linguistic description is then provided at various levels of granularity, that in SFL are is called *delicacy*. Just as the resolution of a digital photo defines the clarity and the amount of detail in the picture, in the same way delicacy refers to the how fine- or coarse-grained distinctions are made in the description of the language.

There is no distinction, in SFL tradition, between lexicon and grammar. And to emphasize this fact, the term *lexico-grammar* is used which means the combination of grammar and lexis into a unitary body (see Section 3.1). A deeper description of the SFL theory of language is provided below in Chapter 3.

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

Both the Cardiff and Sydney grammars have 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) consequently attempted to reuse the grammars for the purpose of syntactic parsing. I come back to these works in more detail in Section 2.

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). To what degree it is possible and what are the benefits of such descriptions still remains to be explored. Moreover it is still unexplored how much of the SFL descriptive potential needs to be employed in practice in order to achieve useful results or solve problems as those exemplified in Section 1.3. The concepts introduced above and other elements of the SFL theory will be addressed in Chapter 3 below. In order to provide a clearer picture on what the SFG analysis represents next section provides with an example.

1.5 A systemic functional analysis example

To provide with a better intuition on the current work, this section describes an analysis of a simple sentence in Example 1. It will guide us starting from a traditional "school grammar" concepts down to a detaile systemic functional description of the sentence. As stipulated in the previous section, SFL provides us with a variety of functions and features serving to express text meaning from several perspectives. Another source of the descriptive breadth is achieved through a practice of feature systematisation as mutually exclusive choices. The feature analysis provided here is partial and restricted to only two constituents (the clause and it's Subject) as this suffices to provide the reader with an intuition of what to expect from a full analysis.

(1) He gave the cake away.

School grammar teaches us how to perform 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* such as verb, noun, adjective etc. to each word; then we would focus on clustering words into constituents guided by the intuitive question "which words go together as a group". Following these actions we will arrive to a word clusterign like the one in Figure 1.1.

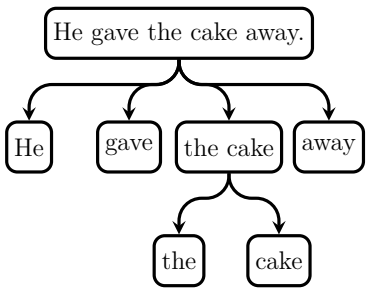


Fig. 1.1 Constituency diagram for Example 1

Figure 1.1 depicts a constituency division of Example 1.The 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. Here the sentence is formed of a single clause which has four constituting functional parts: a subject designating who is the clause about, a predicate indicating the action performed by the subject, a complement denoting what was in scope of the action and an adjunct describing its manner. Each of these functional parts is filled correspondingly by a pronoun, a verb, a nominal group and an adverb. This analysis can be seen in Figure 1.2 as a constituency tree where the nodes carry classes and functions within parent

units. In the figure, nodes have been split into three sections for clarity purposes. The first section is filled with text fragments, the second (in blue) with unit classes and the third (in red) with unit functions.

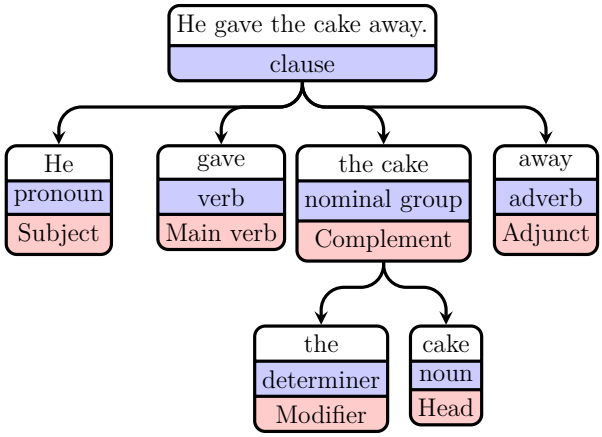


Fig. 1.2 Constituency analysis of Example 1 with unit classes and grammatical functions

Next each constituent can be assigned a set of relevant linguistic features. For example the subject "He" is a pronoun whose features, well defined in the traditional grammar, are: *singular*, *masculine*, and *3rd person*. 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 1.3) 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 1.3 below. This representation constitutes what in SFL is called a *system network* and will be formally introduce in Chapter 3.

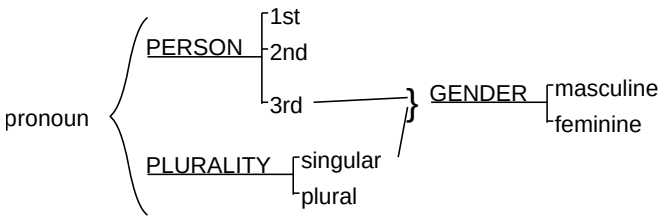


Fig. 1.3 The systematisation of three pronominal features in traditional grammar

In SFG the pronouns are systematised in the system network of Person from *Introduction to Functional Grammar* (Halliday & Matthiessen 2013: 366) that has a different structure as depicted in Figure 1.4. This systematisation reflects a semiotic

perspective where language is placed into an interactive context. The (red) rectangles from the figure represent selections that are applicable to the Subject constituent "He" in example above. These selections are the result of traversing a system network deciding at each step which branch to follow and advance to the next system in case one is available.

From the perspective of an agent generating the utterance in Example 1, to produce the pronoun "he" in the subject position it has to make a few choices in the system network. This process is called system network *traversal*. A simplified traversal for selecting the needed pronominal referent can be described as follows. For now, to make it simpler, the explanation on how the decissions are made is ommited focusing mainly on the traversal process itself. So, first the deciding agent chooses in the PERSON system whether the referent participates in the interaction or not (see Figure 1.4). In our example the referent does not participate so the *non-interactant* feature is selected and we proceeds towards the next system further distinguishing the type of *non-interactant*. It can be plural or, as in our case, singular leading to *one-referent* feature. Next, the referent needs to be differentiated on the consciousness axis which, in our example, is a *conscious* thing. And finally conscious referents need to be distinguished by gender, which in this example is masculine and therefore *male* sex type is chosen. This path of choices uniquely identifies the pronoun "He" in a system network which also defines, just like the one in Figure 1.3, the boundaries of all choice possibilities.

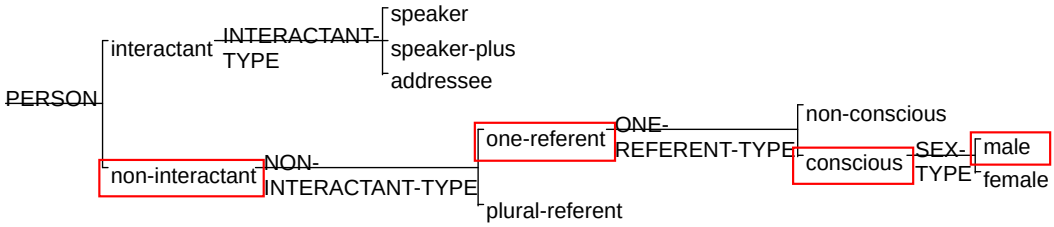


Fig. 1.4 The selections in Person system network from Halliday & Matthiessen (2013: 366) for pronoun "He"

Lets take now the clause constituent that is the root of the constituency tree (see Figure 1.2) and see how SFL features can be applied to it. If in traditional grammar the clause is usually ascribed relatively few features, e.g. as having *passive voice*, *positive polarity* and *simple past tense*; in terms of SFL grammar the corresponding 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 1.5 depicts the selections applicable to clause

constituent in Example 1 from Mood system network that is an adaptation of the Mood network proposed in Halliday & Matthiessen (2013: 162). These selections represent choices made by a natural language generation system when producing the utterance by a process similar to the one explained for the pronominal referent above. The traversal description is ommited for brevity. Organisation of the linguistic features is system networks is one of the main things that distinguishes SFL from other linguistic traditions. I will formally introduce system networks, how they are structured and how they function in Chapter 3 and 7 that follow below.

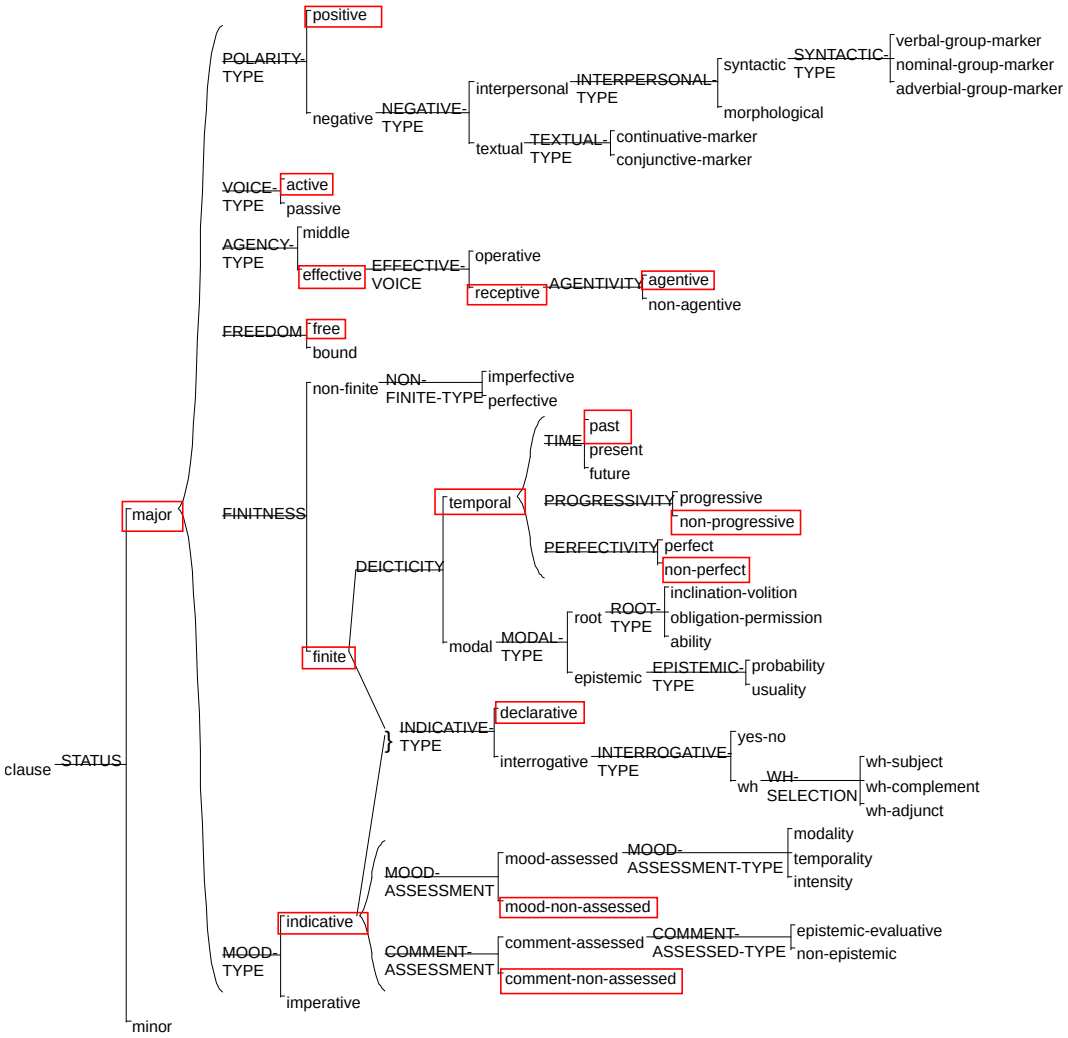


Fig. 1.5 The feature selections in the Mood system network for clause constituent in Example 1

So far we have seen constituents assigned syntactic functions such as Subject, Complement, Adjunct etc. In SFL, they are elements of the *interpersonal metafunction* which will be explained in Chapter 3. SFL provides more linguistic features and

functions depending on the kind of meaning it aims at describing. For example another view on the same clause can be provided from a perspective that in SFL is called *experiential* and corresponds to what in traditional linguistics is known as *semantics*. It is systematised, in SFL, as Transitivity which aims at providing domain independent *semantic frames* called *process configurations*. They 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.

The clause in Example 1 corresponds to a Possessive semantic frame where "He" is the Agent and Carrier while "the cake" is the Affected and Possessed thing. Example 2 provides these annotations. These configurations and participant roles correspond to the Transitivity system network proposed by Neale (2002) which I will introduce in Chapter 4.

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

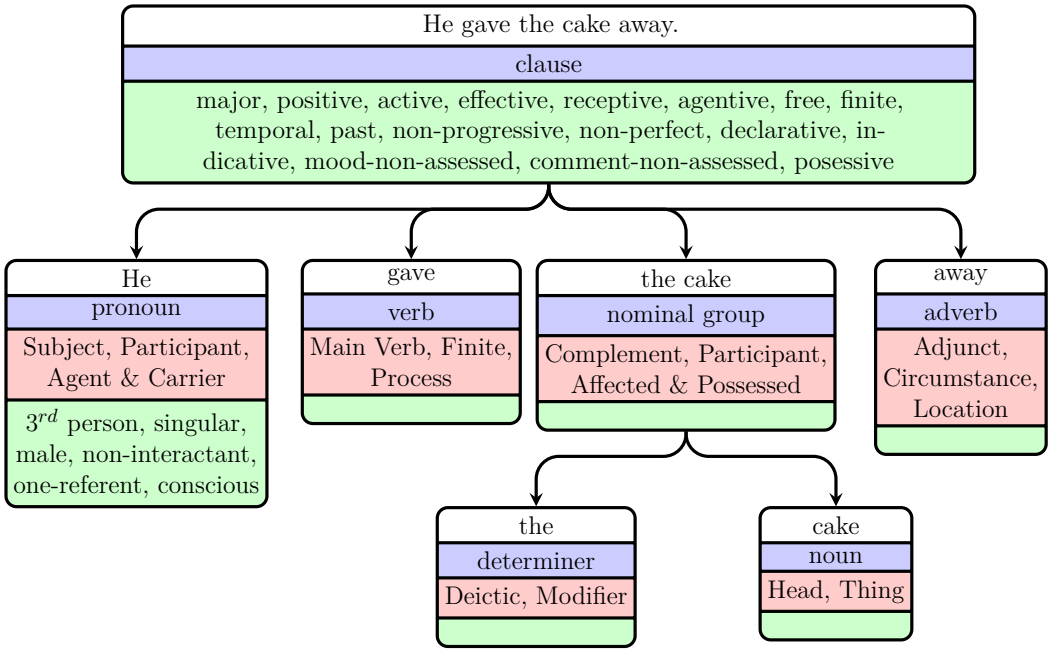


Fig. 1.6 Representation of Example 1 as feature rich constituency tree

There are more functions and features that can be assigned to the constituents in the Example 1 but this is sufficient for the current purposes of introduction. Figure 1.6 summarises everything discussed above into a partially filled constituency tree. The constituents that were not discussed are assigned only a few functions. The last (green) section of every node in the constituent tree is filled with a limited set of grammatical

features selected from system networks. In practice the feature set is much richer than those shown in the nodes in Figure 1.6; the restriction aims simply to avoid an over-crowded example and simplify the exposition. Important to underline here is the systemic functional anchoring of the features into system networks that is an SFL practice.

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 above.

1.6 Problem of parsing with SFGs

In this sections I describe the main challenges for using Systemic Functional Grammars (SFG) in computational contexts and parsing in particular. In short the first and main challenge in parsing with SFGs is that of computational complexity. Partially this problem stems from the manner grammars are structured and the fact that paradigmatic descriptions have received most of the attention at the expense of the syntagmatic ones. The second challenge is parsing with features that depart from directly observable grammatical variations towards increasingly abstract semantic features. Addressing this problem requires answers to a couple of other issues: the availability of a lexical-semantic database and a resolution mechanism for *covert constituents*. As we will see motivated below, the latter are useful constructs in providing a solution even though some linguistic theories reject the mere existence of such things. Next I describe in detail the main problems. I will start with unbalance between paradigmatic and syntagmatic accounts in SFL, then bring the computational aspects into the picture comparing the natural language generation and parson problems, after which I turn towards the problem of parsing with more abstract features and draw parallels to *semantic role labelling* problem in mainstream linguistics. Where appropriate, I will also suggesting how potential solutions may look like which will be further presented in Section 1.7.

1.6.1 The lack of suitable syntagmatic descriptions in SFG

SFL since it was established has been primarily concerned with the paradigmatic axis of language. Accounts of the syntagmatic axis of language, such as the syntactic

structure, have been put in the background. Within SFL, as we will see in Chapter 3, structure is a syntagmatic ordering in language capturing regularities and patterns which can be paraphrased as *what goes together with what*. It has been placed on the theoretical map and defined in terms of *rank*, *unit*, *class* and *function*, but afterwards it received minimal attention.

Most of the descriptive work, in SFL, is carried paradigmatically via *system networks* (Definition 3.2.10) describing *what could go instead of what* (Halliday & Matthiessen 2013: 22). Having the focus set on the paradigmatic organisation in language is in fact 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 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).

If we attend SFL literature, however, the syntagmatic dimension is implicit and present everywhere in the SFL literature, which makes the above claims sound little surprising. 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) which means that every paradigmatic description must be accompanied by precise rules how it is syntagmatically realised in text. Yet, despite these inducements, the situation could not have been more different. 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, yet quite beneficial for the text generation task.

1.6.2 Parsing is asymmetric to generation: the computational complexity issue

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 in computation (Bateman 2008: 27).

The active exchange between SFL theory and computation has been almost entirely oriented towards automatic *natural language generation*. Such systems take abstract semantic specifications as input and use grammars to produce grammatically correct and well connected texts. One of the grammars successfully used in generation tasks is the Nigel grammar developed within Penman generation project (Mann 1983a). The efficiency in generation tasks is, in part, due to decomposition of language along the paradigmatic axis using functionally motivated sets of choices between functionally motivated alternatives (McDonald 1980). The Nigel grammar 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).

The computational processes driving 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 a 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 expressing communicative goals in generation in almost ideal manner (Bateman 2008: 28).

Automatic analysis or *parsing* can be seen as a reverse problem of finding appropriate analysis within a search space of possible solutions. That is to identify, as accurate as possible, the 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 two things. First a description in terms of a formal structure of the sentence revealing the constituents plus their syntactic relations to each other. And second, a description 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 (see example in Section 1.5), 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 is that of handling a combinatorial space which 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 a typical grammar includes many disjunctions, which leads 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 calculate where would the upper limit of a grammar fall even when the configuration of relations of a particular system network is known.

For the generation task the issue of size is not a problem at all as the number of choice points is actually rather small. Such a paradigmatic organisation is, in fact, a 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). This property of gradual exposure of choices characterises the traversal of the system networks, in generation process, which starts from the root and gradually advances towards more delicate features down to a leaf.

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 to explore the 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 needs to be checked leading to evaluation of the system network as a whole. There is then no way to restrict the search space as in the case of generation (Bateman 2008: 29).

One of the grammars successfully used in generation tasks is the Nigel grammar, described above, which is a large grammar by modern standards. To parse with such a grammar would means exploring a search space of approximately 3×10^{18} feature

combinations. A more detailed break down the complexity by rank or primary class as provided in Table 1.1 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 1.1 Size of major components of the Nigel grammar expressed in terms of the number of selection expressions generated (Bateman 2008: 35)

1.6.3 The challenge of parsing with semantic features

Another difficulty in parsing with SFGs lies 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. The 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 (forthcoming) called Process Type Database (Neale 2002).

Such databases provides with 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).

1.6.4 The issue of covert elements

Besides the challenge of identifying configurations and their participants in text, the problem with semantic features goes one step further. Sometimes the semantic roles correspond to constituents that are displaced or not realised in the text called *covert* or *null elements*. This increases the challenge of identifying and assigning them correctly. Next I show how (non-)realisation in text of the semantic roles impacts possibility to interpret that text. Then I will show how frames may still be valid even when an element is realised or displaced which will bring us to the core of the problem. This is followed by a brief description of the approach taken in the current work to solve it.

For a frame to be considered correctly realised in text it needs to fill at least the mandatory roles. Let's imagine that a part of the text in Example 3 is erased. If we take the Agent-Perceiver away as in Example 4 the texts is perceived as incomplete because it is not possible to interpret its meaning. It leaves us with the questions *Who* glanced at her new watch? Similarly, if we delete the Phenomenon like in Example 5, we are unable to resolve the meaning of the text without first answering the question *what* or *who* did Jaqueline glance at?

(4) glanced at her new watch

(5) Jaqueline glanced

Consider now Example 6. It is a sentence that has three non-auxiliary verbs: seem, worry and arrive. According to the Cardiff grammar, which will be introduced in Chapter 3, this corresponds to three clauses *embedded* into each other. A constituency analysis is provided in in Table 1.2.

(6) 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 1.3. For the sake of this example the first role 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

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 1.2 SF constituency analysis in Cardiff grammar style

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 distribution**
seem$_1$	Attributive	Carrier + Attribute
worry about$_1$	Two Role Cognition	Agent & Cognizant + Phenomenon
miss$_1$	Possessive	Affected & Carrier + Possessed (thing)

Table 1.3 Semantic role configurations according to Neale (2002); Fawcett (forthcoming)

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 7 and 8 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 also an index i to highlight that the null elements correspond to the higher clause Subject "She".

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

(8) *She* missed the river boat.

(9) 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 1.4 below.

She$_i$	*seemed*	$\emptyset_i$	*to*	*worry*	*about*	$\emptyset_i$	*missing*	*the*	*river*	*boat.*
Attributive configuration										
Agent		Attribute								
		Two role cognition configuration								
		Agent & Cognizant			Phenomenon					
					Possessive configuration					
						Affected & Carrier		Possessed		

Table 1.4 Transitivity analysis in Cardiff grammar style (Neale 2002; Fawcett forthcoming)

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. In SFL, Fawcett describes these elements in the context of Cardiff grammar (Fawcett 2008: 115,135,194) but provides no means to recover them. Some 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 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.

1.6.5 The problem summary

This section has shown some of the issues related to parsing with SFG. In summary, 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.

Regarding the problem of computational complexity explained above, how could the large 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) in at least what would a possible solution look like. The lack for an answer to the second part and probably for other hidden reasons the results of parsing with SFGs so far are not usable in real world applications. This is drawn from the past 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.

Therefore to address parts of the above problems I attempt a different approach. Some linguistic frameworks, other than SFL, have been shown to work well in computational contexts solving problems similar to the ones identified above. For the purposes of this thesis I selected Dependency Grammar (DG) and GBT. And instead of attempting to find novel solutions within the SFL framework, an alternative approach, I argue in the next section, would be to establish a cross-theoretical and inter-grammatical links and to enable integration of the ready solutions.

1.7 Goals and scope of the thesis

This thesis aims at a modular method for parsing unrestricted English text into a Systemic Functional constituency structure using fragments of Systemic Functional Grammar (SFG) and dependency parse trees.

As will be described in Chapter 2, 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 1.7.4) 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.

1.7.1 On theoretical compatibility and reuse

In this thesis three linguistic frameworks are employed, 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 carry answers to above stated problems. The goal is to maximise positive properties and enable reusing the results which brings us to Research question 1.

Research question 1 (Reuse positive results)**.** To what extent resources and techniques from other areas of computational linguistics can be reused for the SFL parsing and how?

In the past decades much significant progress has 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.

In the past decade *Dependency Grammar* (Tesniere 2015) has become 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.

Research question 2 (Compatibility of DG and SFG)**.** 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?

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, as formulated in Research question 2, 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 5 and then indirectly evaluated empirically in Chapter 10 based on Stanford Dependencies parser version 3.5 (Marneffe & Manning 2008b,a; Marneffe et al. 2014).

Research question 3 (Compatibility of GBT and SFG)**.** How can Government and Binding Theory be used for detecting places of null elements in the context of SFL constituency structure?

The problem of accounting for the *null elements*, mentioned above, is not addressed either in SFL or in Dependency Grammar. It is, however, addressed in detail 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, as formulated in Research question 3, 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. Chapter 6 introduces 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 9 benefits the Transitivity analysis.

1.7.2 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 will be addressed by filling the gap of the syntagmatic account within the SFL grammar directly. This 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) who used phrase parse structures of the Chomskian style grammars. This approach 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 degree to which Stanford dependecies are suitable to serve as a syntactic backbone is one of the questiosn adressed in this thesis (Research question 4) which requires beforehand a theoretical discussion in terms of what is being transformed (expressed in Research question 2). An account of correspondence between linguistic primitives or configurations of primitives in the dependecy grammar to SFG

primitives is provided in the end of Chapter 5 along with an analisys of Stanford dependecy grammar.

Research question 4 (Compatibility of Stanford DG and SFG)**.** Is Stanford Dependency grammar suitable as a syntactic backbone for parsign with Parsimonious Vole grammar?

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 8.

1.7.3 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.

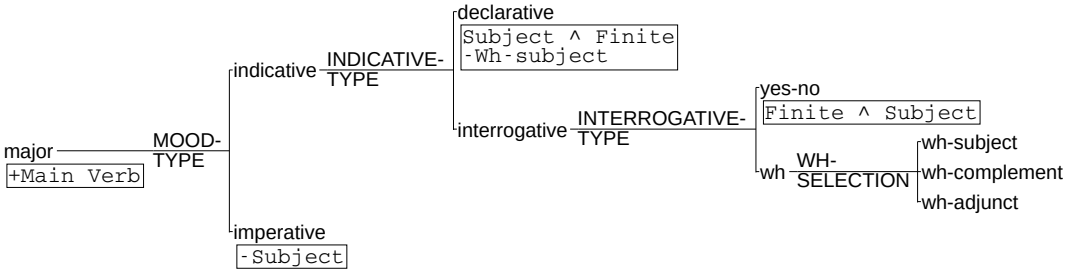


Fig. 1.7 A fragment of mood system from Halliday & Matthiessen (2013: 366)

The system network fragment in Figure 1.7 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 7.3) 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 **??**) anywhere in the constituency structure and if so then the matched nodes are enriched with the features proved in the pattern (described in Section 7.5).

The structural patterns in this work are expressed as *graph patterns* (described in Section 7.3). 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 structural configurations including 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 9.2.

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.

Research question 5 (Coverage of syntactic patterns)**.** What degree of systemic delicacy can be acheived using syntactic patterns alone wihout any lexical-semantic resources?

The pattern may comprise syntactic configuration only or it can also carry lexical-semantic specifications. The two differ in the way they are generated and maintains therefore an investigation of how many features can be expressed in terms of sole syntactic configurations is important. The two main system networks targeted in this

work are MOOD and TRANSITIVITY (described in Chapter 4). One assumption challenged by the Research question 5 is that 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.

Research question 6 (PTDB suitability)**.** How suitable is Process Type Database as a resource for SFL Transitivity parsing?

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 generation of such patterns which then are used for enrichment with TRANSITIVITY (described in Chapter 9). The appropriateness of PTDB for these tasks is inquitred by Research question 6 and addressed in Chapters 4 and 9. In the next section is presented an overview of how these processes fit together in a unitary pasing process.

1.7.4 Parsimonious Vole architecture

The current thesis is accompanied by a software implementation called the Parsimonious Vole parser. It is written in Python programming language and is available as open source distribution[1] . This section provides an overview to the construction process.

The parser follows the pipeline architecture depicted in Figure 1.8 where, starting from an input text, a rich systemic functional constituency structure is progressively built. Figure 1.8 provides three types of boxes: the rounded rectangles represent the parsing steps, the green trapezoid boxes represent input and output data while the orange double framed trapezoid boxes represent additional resources involved in the parsing step. The parsing steps linearly flow from one to the next via green trapezoid boxes on the left-hand side, which represent input-output data in between the steps. On the right-hand side are positioned double edged orange trapezoids representing fixed resources needed by some operations. 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 one, *Graph Building* (spanning the first three process steps), accomplish construction of the constituency backbone (corresponding to the syntagmatic account described in Section

[1] https://bitbucket.org/lps/parsimonious-vole

1.7.2 above) and the second phase *Graph Enrichment* (spanning the last three process steps) flashes out the backbone with features (described in Section 1.7.3).

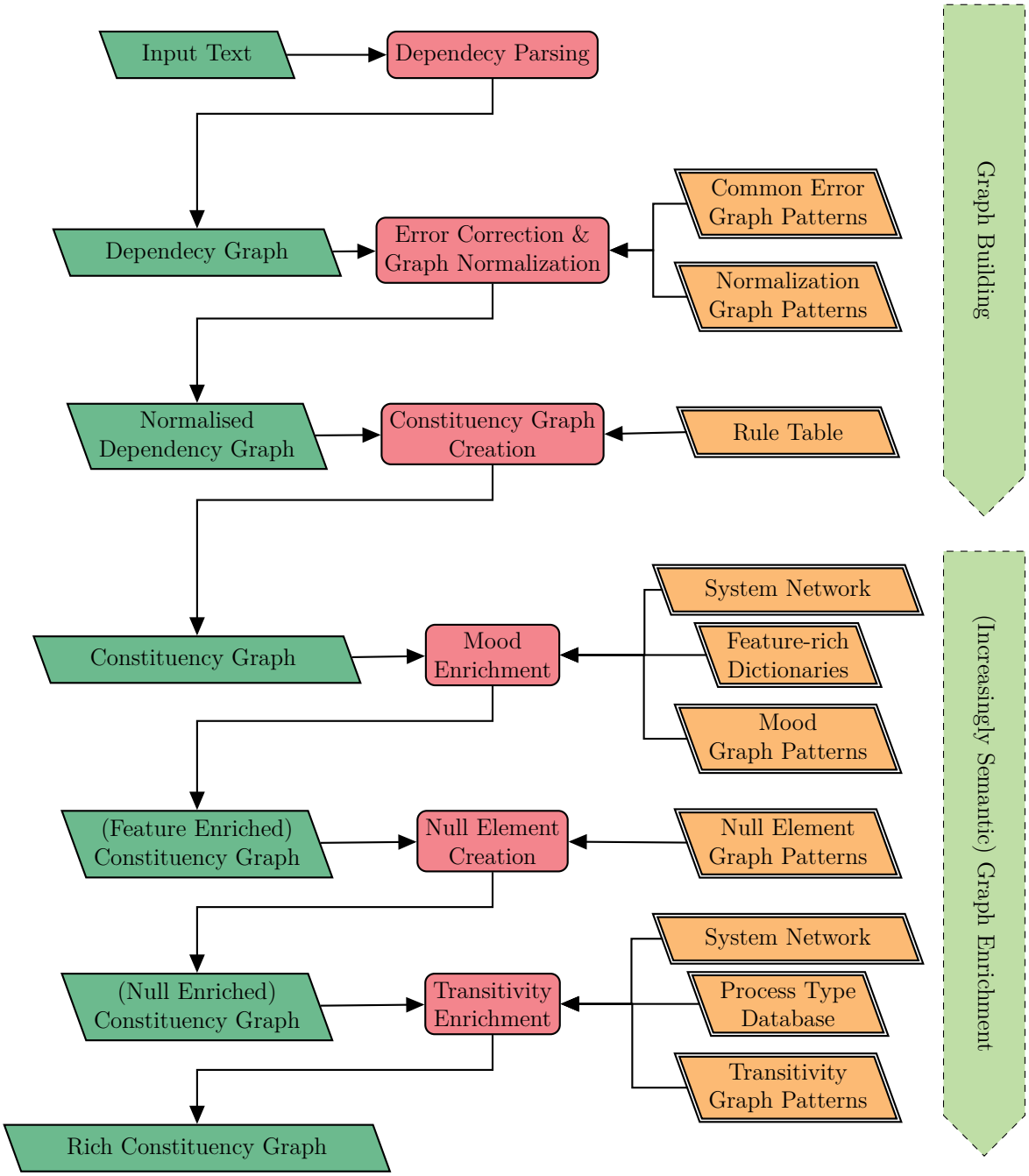


Fig. 1.8 The parsing process pipeline

One important feature of this implementation is its heavy reliance on graph pattern matching and other operations using graph patterns described in Chapter 7.

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 (Chen & Manning 2014) version 3.5[2] producing a Dependency Graph.

The dependency graphs often contain errors. Some of these errors are predictable and so easy to identify and correct. Also, 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 and Normalised* using graph pattern matching against collections of known errors and a set of normalization rules encoded as graph patterns.

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 parallelly constructed guided by a mapping *Rule Table*. The mappings indicate what operation to perform on the newly emerging graph given the visited dependency node and its incoming and outgoing relations. Even if it a distinct structure, the constituency graph is, in a way, a transformation of the dependency graph. It constitutes the syntactic backbone on which the subsequent enrichment phases are performed.

Next follows the phase where each constituent node of 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 an additional construction process adding where needed *empty constituents* that play an important role in semantic enrichment. The enrichment steps use additional resources such as *system networks*, *feature rich lexicons*, *graph patterns* and a *semantic database*. 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 6. 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 the detailed parser implementation choices and developed algorithms are presented in Chapters 8 and 9.

Next section lays out the thesis structure indicating the important contributions that every chapter provides.

[2] https://nlp.stanford.edu/software/nndep.html

1.8 Thesis overview

Chapter 1 has provided an introductions to the work described in this thesis. It has indicated the are to which it seeks to contribute, and described the motivation of work from an applied and a theoretical perspective. In Chapter 2 a list of selected works on parsing with SFG is presented and briefly discussed.

Chapter 3 provides and overview of the SFL theoretical foundations. There are two outstanding traditions in SFL each providing a theory of grammar. First is developed in Sydney by Halliday, Matthiessen, Hassan, Martin, Rose and others. The second is developed in Cardiff by Fawcett, Tucker, Tench and others. I present both schools in first two sections of the chapter and then, in the third section, I provide a comparative critical discussion on both theories of grammar motivating relaxation of the rank scale, approach to structure formation, unit classes and few other concepts relevant to current work.

In the next chapter I provide a description of the grammar implemented in the Parsimonious Vole which. It is a selection of unit classes from both Sydney and Cardiff Grammars following the theoretical motivation from the previous chapter. Here is also presented a selection of two system networks: MOOD and TRANSITIVITY that were selected to demonstrate how the current parsing method works. The former system network is tightly linked to the syntagmatic variations in the structure whereas the latter describes ideational choices of the semantic structures and, thus, is farther from the surface variations. In order to integrate this system network I use a lexical database of verb meanings called Process Type Database Neale (2002).

Chapter 5 introduces the Dependency Grammar 1959, starting with it's origins and foundations, evolution into its modern form, its applications in computational contexts particularly highlighting the Stanford grammatical model and parser. The usage of dependency grammar and dependency parse graphs is motivated in 1.7.2 as the primary input into the current parsing pipeline for creating the constituency structure. The last part of the chapter provides a set of principles and generalizations to establish cross theoretical bridge from the dependency grammar towards the systemic functional grammar which are implemented into the Parsimonious Vole parser.

Next chapter starts with an introduction of Government and Binding Theory (GBT) explaining where the empty constituents occur in sentences. These constituents were motivated in 1.6 and are a part of solution for parsing with TRANSITIVITY system network. The second section of the chapter provides an inventory of different null elements and the last section provides, just like in the previous chapter, a cross theoretical overview, this time from GBT phrase parse structures into Stanford dependency

grammar. It provides a theoretical translation of the principles from GBT into DG constituting the theoretical foundations for the technical solutions, in Section 9.3, for how to create null elements in DG and SFG graphs.

Chapter 7 provides the building blocs of the algorithms of this thesis. It makes the transition from the linguistic theoretic presentations towards the computer science foundations introducing necessary typed sets, feature structures and graphs. These concepts are employed in the chapters that follow to represent linguistic constructs described in the previous chapters. An important role, in the current work, play the pattern graphs and the operations enabled by using them presented in Sections 7.3 – 7.5. The pattern graphs, as will be presented latter constitutes a flexible and expressive method to represent systemic feature realisation rules. Also in this chapter, the system networks, are defined in a simplified form corresponding to how they are currently used along with a simple strategy for choice propagation.

The first phase of the parsing pipeline (see Figure 1.8) concerning the constituency graph building is entirely covered by the Chapter 8. It presents how the input dependency graphs are first corrected and normalised and the how they are rewritten, using a custom algorithm, into constituency graphs. The implementation of Parsimonious Vole also contaisn a full set of mapping rules between Stanford Dependency v3.5 to SFG constituency structure enumerated in Appendix .2.5.

The second phase of the pipeline (see Figure 1.8) concerning the enrichment of the constituency graph with increasingly more semantic features is described in the Chapter 9. It addresses two main system networks, that of MOOD and TRANSITIVITY introduced in Chapter 4. The MOOD features are close to syntactic variation of text and can be addressed via graph patterns alone in the first part of the chapter. The TRANSITIVITY features are semantic in nature and require additional lexical-semantic resources from which graph patterns are generated first and then applied to enrich the constituency graph. The work presented in this chapter compraises a set of syntactically grounded graph patterns covering Mood and a few other small system networks. It provides with a clean machine readable version of the PTDB along with a method to automatically transform PTDB records into semantically oriented Transitivity graph patterns. Also, graph patterns and alogithms have been developed to capture several principles and mechanisms for detecting null elements in texts.

Chapter 10 describes how the Parsimonious Vole parser was evaluated. Thsi evaluation was constituted on two corpora. One was created, by Ela Oren and I, with the purpose of evaluating the syntactic features of this parser while the otehr was provided by Anke Schultz covering Cardiff Transitivity annotations. The chapter

describes the evaluation settings and the results for syntactic and semantic parsing. Chapter 11 concludes this work by providing a thesis summary overview, indications for practical applications of this work and future directions to follow.

Chapter 2

An overview of selected works on parsing with SFG

There have been various attempts to parsing with SFGs. This chapter 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 constituency trees covering sentence Mood and Theme structure. No systemic feature selections have been assigned to any of the constituents. Also not Transitivity account was provided in their attempts. The current work is following a similar approach of converting a parse structures and in addition providing a set of feature selections from system networks. Next are presented a few selected attempts to parse with SFGs.

2.0.1 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 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

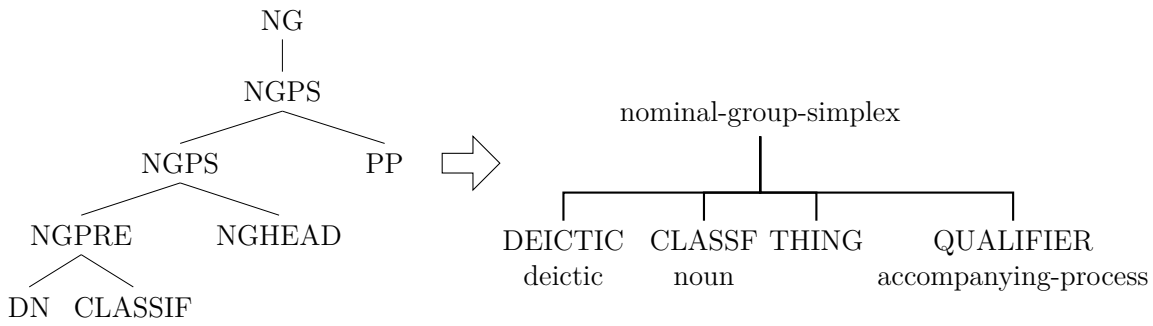


Fig. 2.1 Transformation from phrase structure into systemic constituency structure. Rule example from O'Donnell & Bateman (2005).

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.

2.2 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.

2.3 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.

2.4 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.

2.5 Discussion

In this chapter were presented several relevant attempts to parse with Systemic Functional Grammars. All of them needed to reduce the grammar size or use toy grammars in order to compute results in a reasonable amount of time. The main problem in using SFGs for parsing is that they are much more complex than post-Chomskian grammars: the grammar contains both semantic and syntagmatic aspects of language, the system networks represent large numbers of simultaneous combinations of features, and multiple layers of function structure are conflated together.

Some parsing approaches use a syntactic backbone which is then flashed out with SFG description. Other ones use a reduced set or a single layer of SFG representation the third ones use an annotated corpus as the source of a probabilistic grammar. Regardless of the approach each limits the SFG in a one way or another balancing the depth of description with language coverage: that is either deep description but a domain specific language or shallow description but broad language coverage.

Current approach is aligned with works of Honnibal, Kasper and and O'Donnell with respect to using a backbone structure and enriching it with syntactic and semantic features. Current method employs rules for graph traversal in order to build a parallel backbone constituency tree and rules for graph matching to enrich it with systemic features.

Parsing Transitivity system is a task similar to Semantic Role Labelling and requires large lexicogrammatical resource describing verb meanings in terms of their process type and participant roles. O'Donnell approaches it by providing possible process types directly to the verb by employing self constructed lexicon where each word has syntactic and semantic features. Current approach uses PTDB (Neale 2002), which provides entire process configurations (semantic frames) for each verb sense and the feature assignment is simultaneous, if matched, to the entire configuration of process and its participants.

One major advantage, as compared to Honnibal's approach is that the grammar and the program are carefully disconnected so that the code is maintainable and scalable with the respect to size of the grammar.

Next chapter will introduce the structure of Systemic Functional Grammars and draw some parallels between Sydney and Cardiff schools.

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).

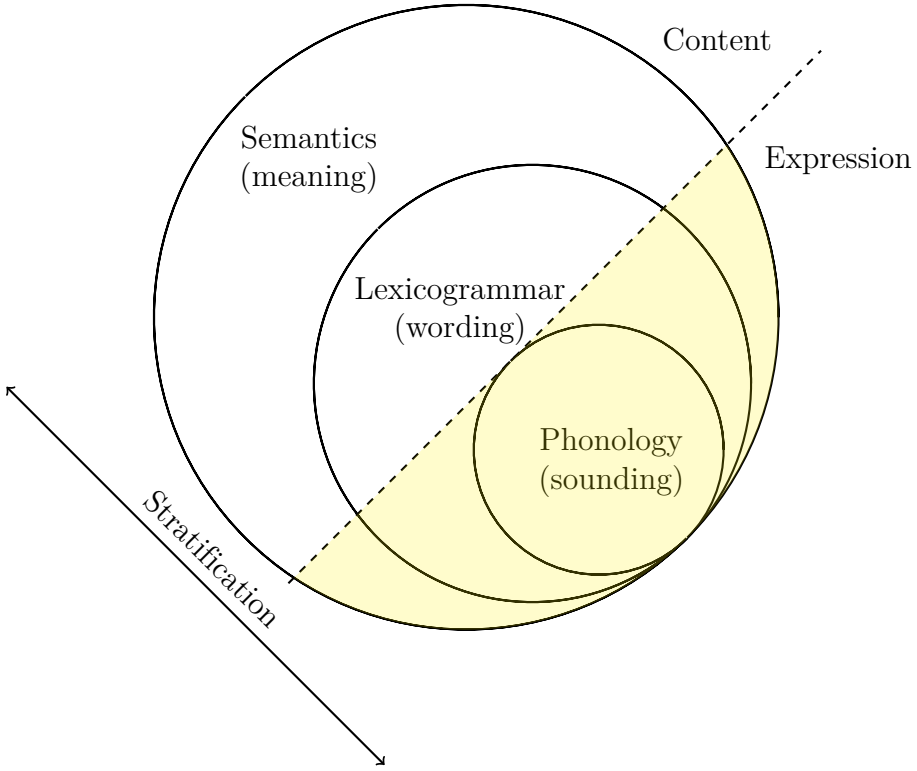


Fig. 3.1 The levels of abstraction along the realisation axis

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 10 *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 11 the prepositional phrase *Who . . . about* is gapped and has an inverted order of constituents.

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

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

(12) 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

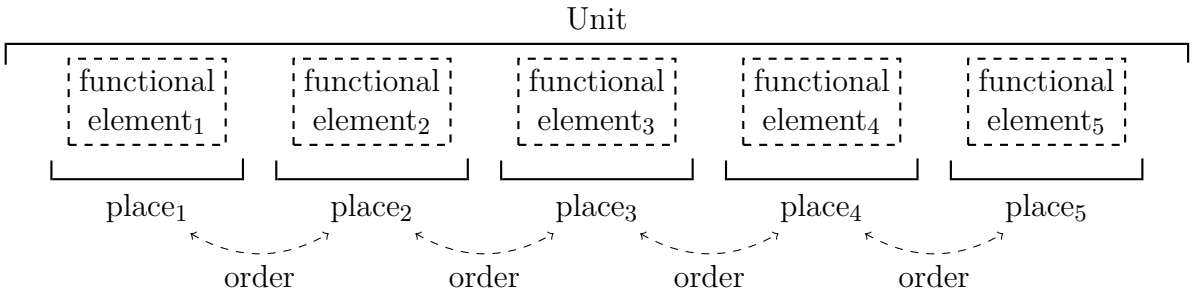


Fig. 3.2 The graphic representation of (unit) structure

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 11.3.

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.

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

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

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

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

For example 13 and 15 are instances of grammatical metaphor whereas the 14 and 16 are their non metaphorical counterparts. In Examples 13 and 14 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).

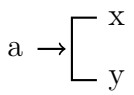


Fig. 3.3 A system with a single entry condition: if a then either x or y

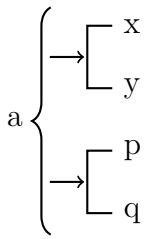


Fig. 3.4 Two systems grouped under the same entry condition: if a then both either x or y and, independently, either p or q

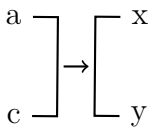


Fig. 3.5 A system network with a disjunctive entry condition: if either a or c (or both), then either x or y

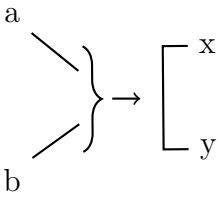


Fig. 3.6 A system with a conjunctive entry condition: if both a and b then, either x or y

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 5.6.

The elements of logical paratactic structure are notated left to right with numbers (1 2) while those of hypotactic structure with Greek letters (β α) 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 restate 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 5) or strictly speaking the *sister* or *sibling dependency* (not parent-daughter). It is suitable for describing relations between elements of structure within a unit.

(17) the man with a stick

For example the componence of nominal group in Example 17 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*.

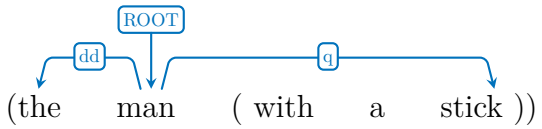


Fig. 3.7 Sibling dependency representation for “the man with a stick”

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 17, 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 18) 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).

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

(19) 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 19) 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).

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

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

In Example 20 we can see an occurrence of direct embedding where an infinite clause acts as the subject of another clause. In Example 21 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.

(22) some very small wooden ones

Consider the nominal group 22. 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 22

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 22

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 22

(23) 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	verbal group	nominal group	prepositional phrase

Table 3.6 Sydney grammar Mood analysis of Example 23

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 23 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 8. 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.

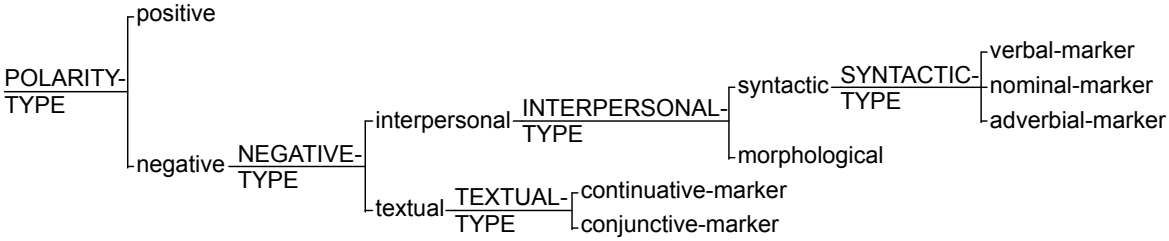


Fig. 3.8 System network of POLARITY

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.

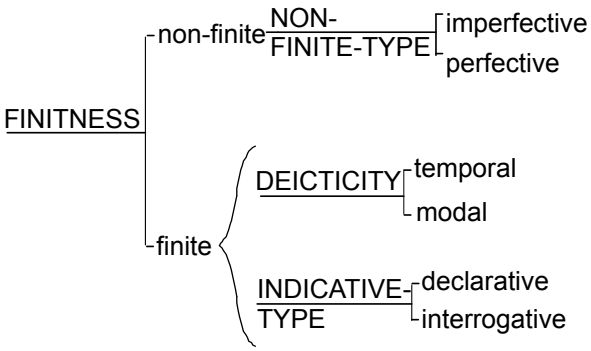


Fig. 3.9 A fraction of the finiteness system where increase of delicacy is not a "is a" relation

The distinction in the systemic relations is incorporated into the technical data structure definitions and traversal algorithms proposed in the Chapter 7.

3.4.4 Unit classes

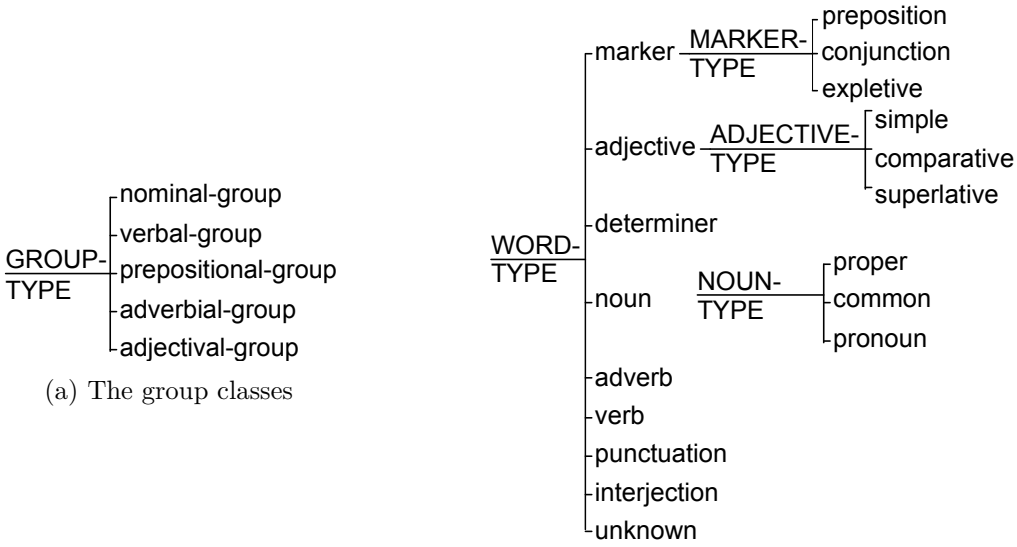
In SFL at large there is the consensus that linguistic forms and meanings are intertwined and mutually determined just like for any sign in a Saussurean approach to language. Both Halliday (quote below) and Fawcett (Definition 3.3.1) adopt this position.

> . . . 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 5, 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 4.1.3) 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 22 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.

(24) (Have) a cup of tea.

(25) The old shall pass first.

(26) I'll give you three.

Consider Example 24 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 4.1.3.

		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 24 with Sydney and Cardiff grammars: diverging semantic and syntactic heads.

In the nominal group "The old" which is Subject in Example 25, 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 26, "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 6 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 5 provides the grounds for cross-theoretical mappings and the empirical evaluation in Chapter 10 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 8.

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 **??**.

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.

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

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

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

(30) * 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 27 can be reformulated into 28 and the one from 29 into 30. 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 31 analysed in Table 3.13 where the conjuncts are a nominal group "last Monday" and a prepositional group "during the previous weekend".

(31) 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 32 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 33 is grammatically incorrect. That leads to the conclusion that the coordination complex can have categorial features which none of the constituting units has.

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

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

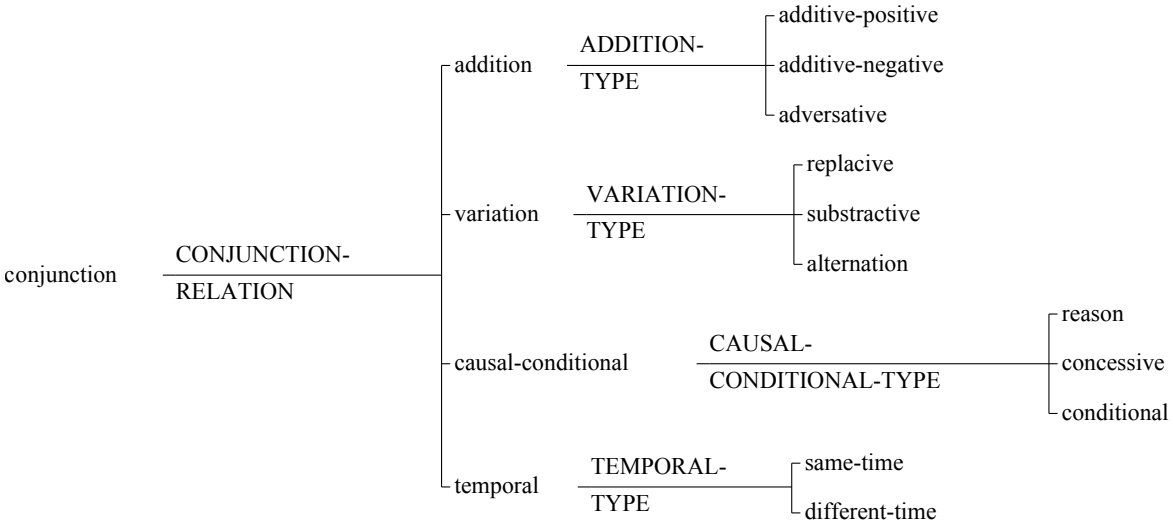


Fig. 3.11 Systemic network of coordination types

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 Concluding remarks

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.

First, in Section 3.2, I introduced the Sydney theory of grammar that originated SFL. Then, in Section 3.3, I introduce Cardiff theory of grammar. It builds on top of Sydney school but differs in several important ways from it. Finally, in Section 3.4, I conduct a critical discussion on the important aspects of both grammars such as unit, class, function, element, rank scale, unit heads, structure and other. This discussion settles my position on some elements of the theory of grammar necessary for implementation of Parsimonious Vole parser.

Chapter 4

The grammar in Parsimonious Vole

Now that the main theoretical foundations have been covered, I turn to describe the grammatical units system networks adopted in this thesis some being from Sydney and others from Cardiff grammars. They have common parts and also differ in large parts on their paradigmatic and syntagmatic descriptions. Just like in the previous chapter I base my discussion on pragmatic grounds.

First I conduct a critical discussion on the differences between the important unit structures in Sydney and Cardiff grammars: the clause, the verbal group, nominal group, the adjectival and adverbial groups. And then I describe the two important system networks: TRANSITIVITY and MOOD. The formed is from Cardiff grammar and the latter belongs to Sydney grammar.

4.1 The grammatical units for parsing

I turn now to discuss the structure of the units implemented in the Parsimonious Vole parser. I provide arguments and reasoning for choosing one over the other unit structure. The general principle for selection is that some unit structures are closer to the traditional syntactic analysis and thus easier to parse and the other ones may be a level of abstraction higher falling on the semantic grounds and thus becoming more difficult to capture in structural variance and requiring additional lexico-semantic resources.

4.1.1 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 34–36.

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.

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

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

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

In Sydney Grammar "started to investigate" (in 34) 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 36. 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 35 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.

4.1.2 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 37, none of which are covered in the present work.

(37) 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 11.3 for a complete list).

4.1.3 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 4.1 An example of a nominal group in the Sydney Grammar (Halliday & Matthiessen 2013: 264)

In Table 4.1 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 4.2 presents the mappings between the elements of nominal group and the word classes.

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,

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

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 4.3 exemplifies a noun group analysed with Cardiff Grammar covering all the possible elements. Table 4.4 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 4.3 The example of a nominal group 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

sym-bol	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 4.4 The mapping of noun group elements to classes in Cardiff grammar

than personal), numeral or adjective (mainly superlatives) then they function as Deictic, Numerative or Epithet. So, as will be described in Chapter 8, 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 38–40.

(38) (a cup) of (tea)

(39) (some) of (those youngsters)

(40) (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 4.5 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 4.5 Example analysis with syntactic and semantic heads

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

(42) 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 41 (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.

4.1.4 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.

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

(44) You're *very lucky.*

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

In the environments where nominal group functions as Attribute, typically in the attributive clauses such as 43, it can take also more contracted forms without the Thing and Deictic where the Head moves left onto the Epithet such as in example 44. One particularity of these nominal groups which here are distinguished as *adjectival group* units is that they cannot function as subject. For the example 45 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 4.6 and

4.7. 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 4.6 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 4.7 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.

4.2 Selected system networks for parsing

The previous section described the units of structure adopted in the current work. I turn now to describe a few systemic networks relevant to selected units that are fairly low in delicacy and are quite close to what is considered grammatical features in the traditional grammar.

It is close to terms and concepts of syntactic structure in traditional grammar (role and definition of Subject, Complement and other functional elements) and the .

4.2.1 MOOD

One of the first system networks presented in the Introduction to Functional Grammar (Halliday & Matthiessen 2013) is that of MOOD. This system is introduced in the discussion of interpersonal metafunction of language. In this view a clause is conceptualised as message exchanged between dialogue interactants. A range of grammatical features from traditional grammar such as *mood*, *modality*, *aspect*, *mode*, *polarity*, *tense* etc. are conveyed to this system network.

The terms in SFL literature sometimes are capitalised to distinguish system names, functions and features. Note that here the *MOOD* (all capital) refers here to the name of the system network; the *Mood* (first capital) is an element of clause structure formed of the Subject and Finite elements which, in fact, is not used in this work because of a general orientation towards Cardiff approach to structure; and the *mood* (no capital) is a type of feature carried by finite clauses (e.g. imperative, indicative, interrogative etc.)

Figure 4.1 presents the MOOD system network employed in the current implementation of the Parsimonious Vole parser. This MOOD system is to a large extent similar to the one from Halliday & Matthiessen (2013: 162). It has a few adaptations that were introduced during development the test corpus described in Chapter 10. The adaptations are adoptions of a few traditional grammar features such as *tense* and *voice* and simplification of the Hallidayan modality.

The features of this feature network apply to units of clause class only. Even if some features may be intuitive or I iterate briefly over each of them providing hints for identifying it.

The POLARITY system indicates whether the clause is affirmed or negated. The negative polarity is indicated, in English, by the presence of a clausal particle *not* or *n't*. It can also be signalled by similar negative markers in other clause elements such as the subject (None of the kids came to play.), adjunct (He is never coming back.) or complement (She loves no one.). In present work I consider the clausal negative marker only.

VOICE in the traditional grammar indicates, for transitive verbs, whether subject acts (*active* voice) or is acted upon (*passive* voice). In passive voice the subject and complement change position and the former is introduced by the preposition *by*.

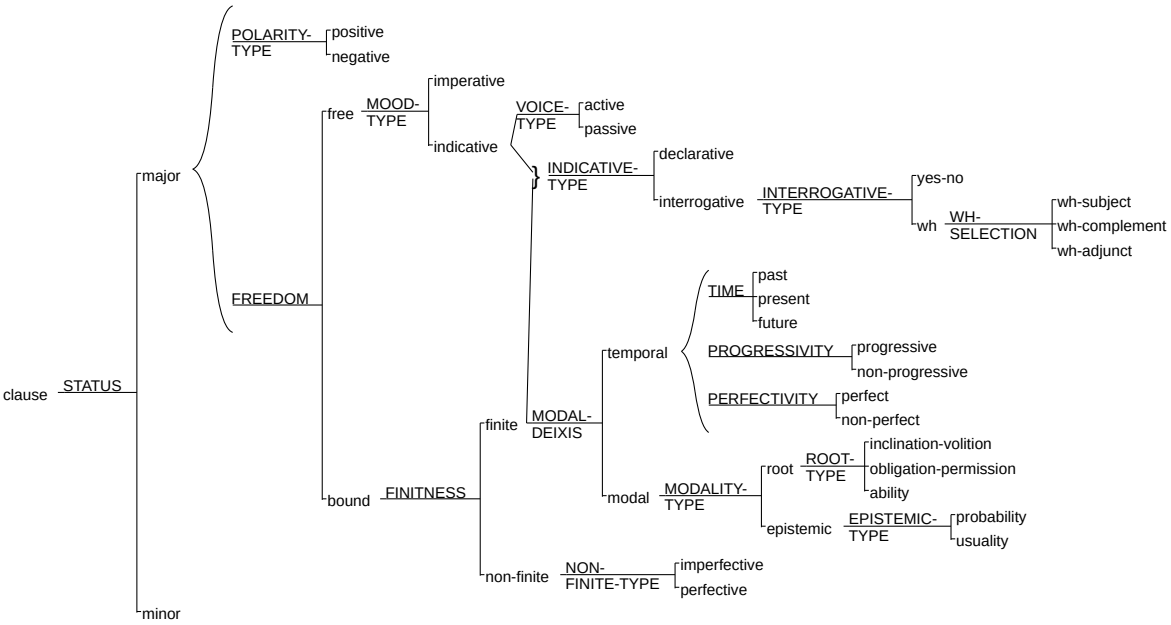


Fig. 4.1 An adaptation of the MOOD system network (Halliday & Matthiessen 2013: 162)

The semantics of FREEDOM system is to indicate whether the clause is *free* and realises a proposition or proposal and serves to develop exchange in a dialogue either by initiating or responding to a speech act. On the other hand the *bound* clauses are not open to negotiation and serve as supporting information to be taken for granted. Structurally, the bound clauses usually depend on a dominant one that is free.

FINITENESS system indicates whether the clause is *finite*, i.e. something that can be argued about. The way to make it arguable is by providing a point of reference into here and now (*temporal*) or into the speaker's judgement (*modal*). The latter two features constitute the MODAL-DEIXIS system.

The MODALITY-TYPE system is an adaptation from (Halliday & Matthiessen 2013: 689–692) that focuses on the usage of the modal verbs only and is organised into ROOT and EPISTEMIC modalities. The former one is comprised of *inclination-volition*, *obligation-permission* and *ability* while the latter of *usuality* and *probability* features.

Temporal feature indicates that the clause has a tense. For simplicity I replaced the Sydney account for tense with that of traditional grammar of English. The systematisation is on three systems, that of TIME (past, present or future), PROGRESSIVITY (progressive or non-progressive) and PERFECTIVITY (perfect or non-perfect).

In addition to MOOD system network I include into the parsing process the DEIXIS system network for the nominal groups determination depicted in Figure 4.2. The

nominal DEIXIS system network is described in detail in IFG4 at the nominal group section (Halliday & Matthiessen 2013: 364–396).

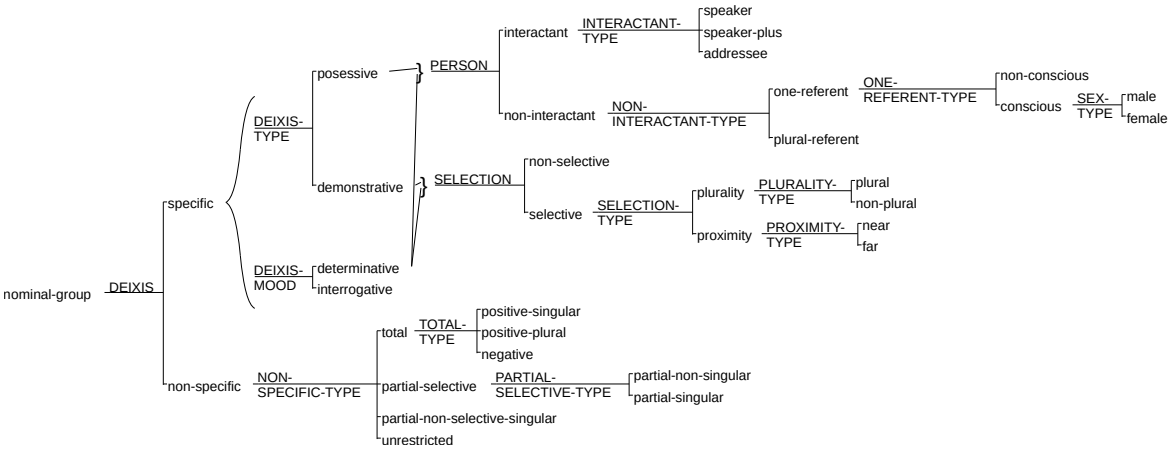


Fig. 4.2 The DEIXIS system network for the nominal group determination (Halliday & Matthiessen 2013: 366)

This system network is relevant for the current work because determining the systemic selections for the entire network can be unambiguously done based on lexical information only. In Section 9.2 we will see a dictionary lookup method in addition to graph pattern matching to determine systemic selections. Next is briefly described the system network of TRANSITIVITY.

4.2.2 TRANSITIVITY

In Section 3.2.5 I explained that SFL organises the lexicogrammar in three metafunctions conflated with each other. This section briefly reminds of the experiential metafunction and introduces the TRANSITIVITY system network that systematizes it.

In this perspective, "the clause construes a quantum of change in a flow of events as a *figure*, or a configuration of a *process*, *participants* involved in it and any attendant *circumstance*" (Halliday & Matthiessen 2013: 212).

In traditional grammar the term *transitivity* refers to the property of verbs according to which they are classified into transitive and intransitive. In SFL the term transitivity is primarily concerned with clauses. It is most insightful to refer to Halliday's TRANSITIVITY (Halliday 1967, 1968b,a) that deals with Predicate, Subject, Complement, and Adjunct all of which are elements of the clause and are usually conflated with the Process, Participants and Circumstances.

Sydney grammar makes a distinction between two types of experience: "inner" as experience inside ourselves and "outer" as experience in the world around us. The

prototypical outer experience is that of actions and events. The inner experience is more difficult to sort but it is a kind of reply of the outer, recording it, reflecting on it, reacting on it etc. The two grammatical categories realizing the these sorts of experience are the *material* process and *mental* process.

In addition to material and mental there is a third kind of process used in identifying, classifying and relating various kinds of experience. The grammatical category realizing this type of links is the *relational* process. Then using the combinations of the three main processes above, Halliday defines *behavioural*, *verbal* and *existential* processes.

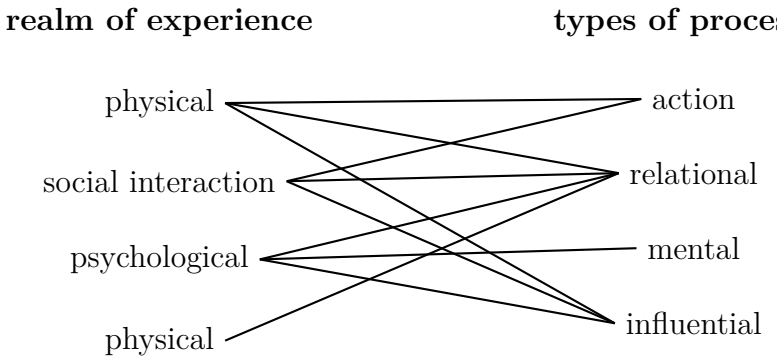


Fig. 4.3 The connections in Cardiff grammar between realms of experience and the process types

Cardiff grammar employs similar process types, that of *action*, *relational*, *mental* and *influential*. In addition it links these process types to realms of experience: *physical*, *social interaction*, *psychological* and *abstract*. Figure 4.3 provides the schematic connection between realms of experience and various process types that can realise that kind of experience (Fawcett forthcoming: 37).

Fawcett (1973, 1987, 1996) wrote the most on the TRANSITIVITY of Cardiff grammar. It is a model that evolved over time and is depicted in Figure 4.4 in its latest form. The first main process type is the *action*. It has been called "material process" in the past, but Fawcett returned to use the term "action" because there are many actions that are non material, social for instance. The second main process is the *relational* one that is subdivided into *attributive*, *posessive*, *locational*, *directional* and *matching*.

The third main distinctions is the *mental* process that if fine-grained into *emotion*, *perception* and *cognition* distinctions. *Environmental* processes, even if very rare, are recognised as another main process type. *Influential* processes are unique to Cardiff grammar and not accounted elsewhere in the TRANSITIVITY system. All these processes have a structural similarity, that of having an embedded event into the

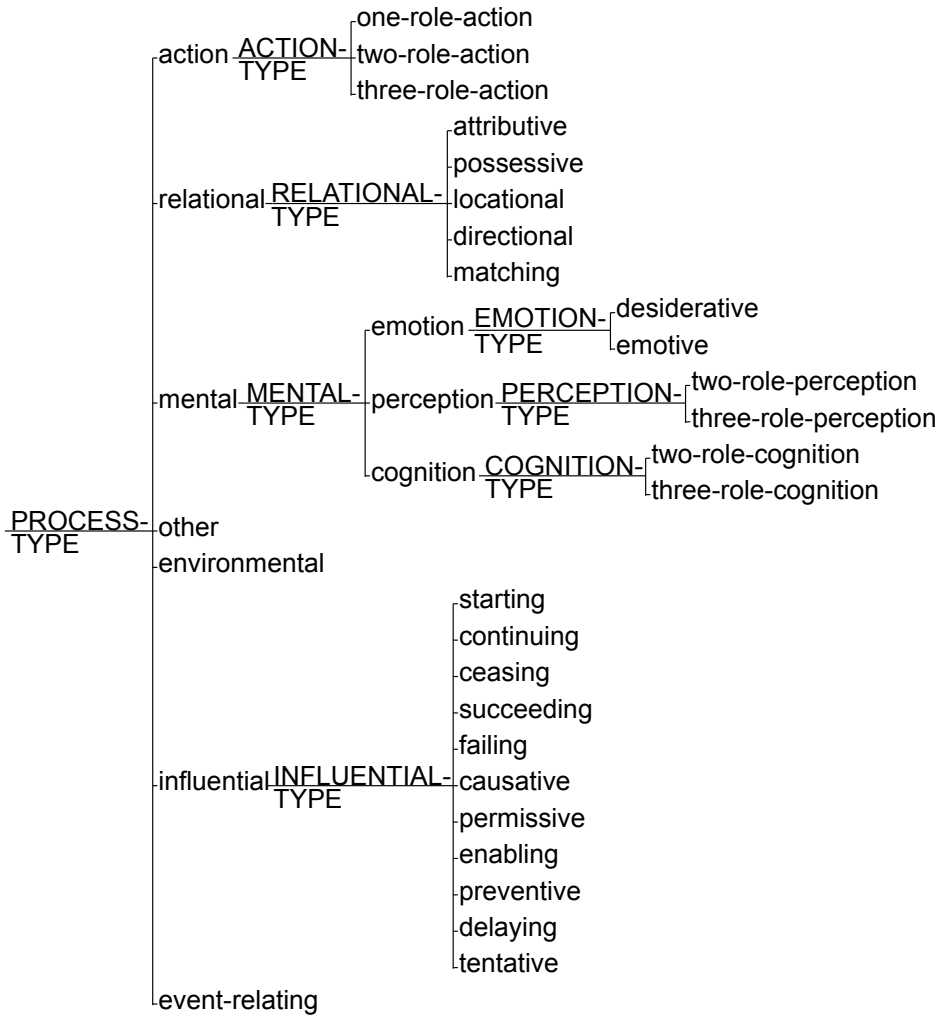


Fig. 4.4 Cardiff TRANSITIVITY system network

matrix process and which is somehow influenced. The last process type is that of *event-relating* which is also very new and specific to Cardiff grammar. All the linguistic phenomena covered by this process are treated by Halliday as grammatical metaphors, but Fawcett considered that they shall be analysed as a distinct process type.

The TRANSITIVITY system is highly dependent on the lexical semantics of the verbs. Therefore a vast account of verb senses and the participant configuration structure they command has to be provided in the grammar. Neale (2002) has pioneered such work in her thesis which I next introduce.

4.2.3 The Process Type Database

The Process Type Database (PTDB) (Neale 2002) is the key resource in the automatic Transitivity analysis addressed in Chapter 9. It is also the source for creating the graph patterns used to enrich the constituency graph as described in the same chapter. 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 4.8 presents a simplified sample of PTDB content.

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

Table 4.8 An example of records ins PTDB

4.3 Concluding remarks

This chapter has described the grammatical units and the two system networks adopted in this work. They constitute a selection from from Sydney and Cardiff grammar implemented in the Parsimonious Vole parser.

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. It is also the reason to adapt in this thesis Sydney unit structures as they are closer to traditional grammar syntax (Quirk et al. 1985).

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 5

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.

5.1 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. 5.1 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 5.1. 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 5.2 depicts the clause structure "Alfred speaks slowly" where both the subject and the object are subordinated to the central verb speaks.

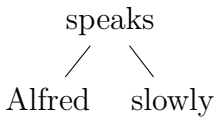


Fig. 5.2 Stemma for "Alfred speaks slowly"

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).

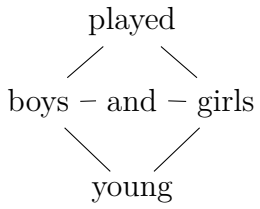
Atoms are not the only metaphor he uses and next I present another one regarding the *verbal node* that is especially important for showing the syntax-semantics interplay.

> 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).

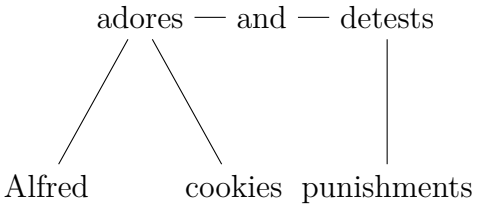
Comparison of the verb to an atom seems to emphasize connection to the syntactic aspect of valency while comparing it to a theatrical performance seems to emphasize the semantic properties of valency. Therefore his theory of valency has semantic and syntactic properties. He believed that the first actant 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 5.3 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

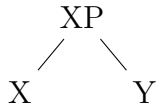
Fig. 5.3 Sample stemmas with *junction* representation

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.

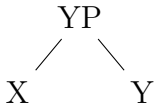
Tesniere splits the words into *function words* or *translatives* (i.e. prepositions, conjunctions, auxiliary verbs and articles) and four basic categories of *content words* (i.e. verbs (I), nouns (O), adverbs (E) and adjectives (A)). The former are empty of content marker transfer of content words 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.

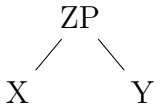
The transfer theory is controversial for the translators of the Elements. They write (Tesniere 2015: liv-lx) that while the transfer schema can not be interpreted in terms of pure dependency it is debatable whether it can be interpreted in terms of constituency. The main distinction is in the number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



(a) Headed endocentric

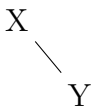


(b) Headed endocentric

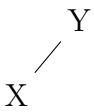


(c) Non-headed exocentric

Fig. 5.4 Constituency structure



(a) Headed endocentric



(b) Headed endocentric

(c) Non-headed exocentric

Fig. 5.5 Dependency structure

Figure 5.4 shows how a sequence of two elements X and Y can be represented in terms of constituency where the Figure 5.4a and 5.4b represent that one element

governs the other called *endocentric* structures and in Figure 5.4c a non-headed structure called *exocentric*. Dependency structure depicted below in Figure 5.5, in contrast, cannot represent non-headed structures. Hence there is no correspondent dependency representation to Figure 5.4c in Figure 5.5c.

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.

5.2 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.

5.2.1 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).

5.2.2 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.

5.2.3 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.

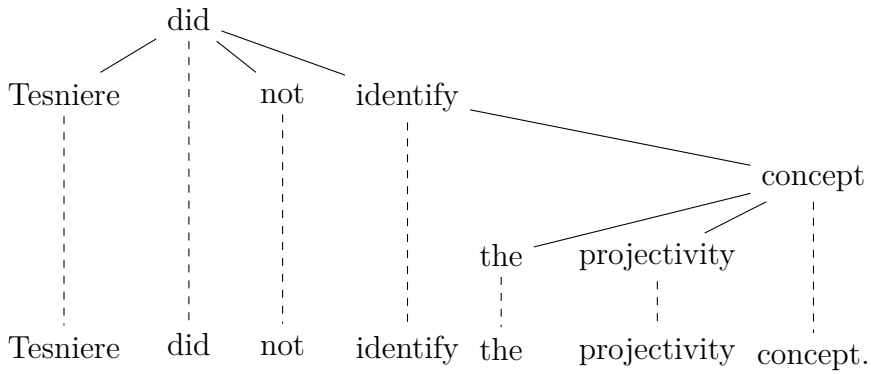


Fig. 5.6 Projective tree

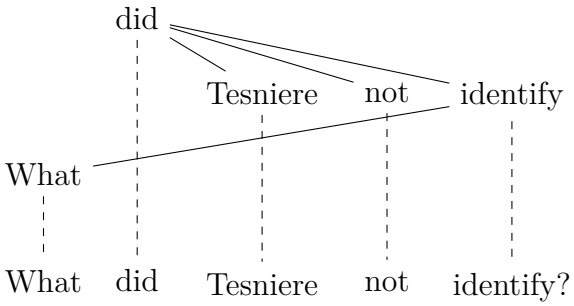


Fig. 5.7 Non-projective tree

To illustrate this principle consider Figure 5.6 where there are no crossing lines whereas Figure 5.7 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.

5.2.4 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.

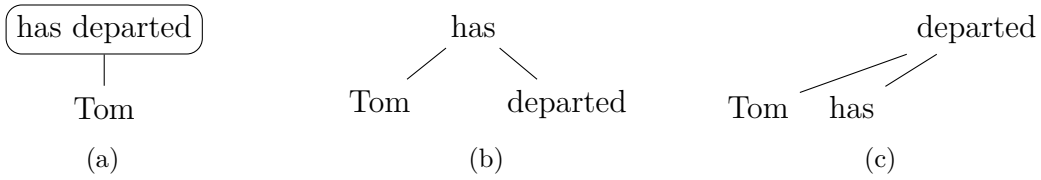


Fig. 5.8 Possible analysis representation for "Tom has departed"

Figure 5.8 represents three possible ways to analyse the word has in "Tom has departed". In Figure 5.8a 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 5.8b and 5.8c 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 5.8c. 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.

5.3 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 5.9) 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

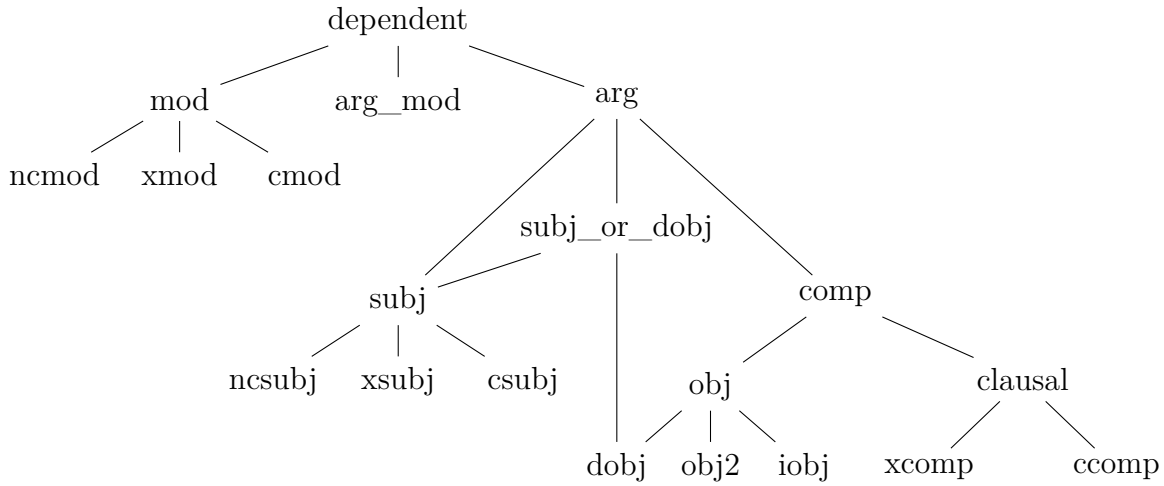


Fig. 5.9 The grammatical relations (GR) hierarchy from Carroll et al. (1999)

(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.

5.4 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 5.4.1.

Generalization 5.4.1 (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.

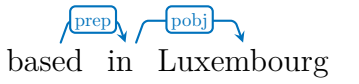
The advantage of explicit relations is that the predicate-argument relations are readily available as edge labels in the dependency structure and can be used off the shelf for real world applications which was an important goal in the schema design.

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 .2.5.

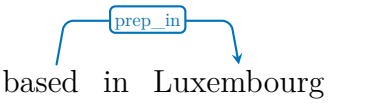
5.5 Stanford dependency representation

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 5.10a, 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 5.10b. There is a single relation between "based" and "Luxembourg" labelled "prep_in". Similar transformations are done for conjunctions.



(a) Basic (uncollapsed) preposition dependency

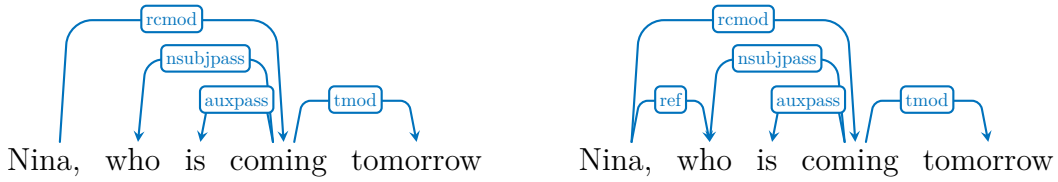


(b) Collapsed preposition dependency

Fig. 5.10 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 5.11a 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 5.11b, 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. 5.11 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 7. 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 8.

5.6 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 8.

Lets recap what dependency relations are in the Dependency theory and in Systemic Functional theory of grammar. In the Dependency theory of grammar, as we saw in Section 5.1, 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 5.13b) 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.

(46) The witness seemed quite convincing.

Consider the example 46 whose representation as dependency structure is depicted in Figure 5.12 and as Systemic Functional constituency structure in Table 5.1. In Figure 5.12 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".

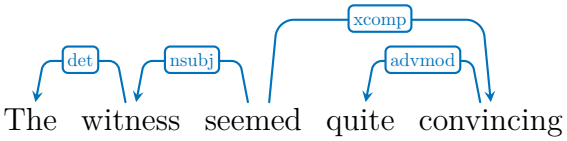


Fig. 5.12 Dependency representation

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

Table 5.1 Example of head-modifier sibling dependency

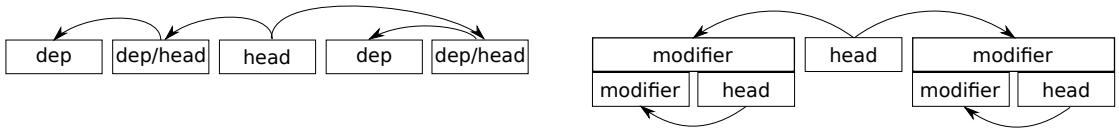
In Table 5.1, the corresponding structure, is a root clause unit composed of three elements: a Subject a Main Verb and a Complement. The Main Verb is the pivotal

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 5.6.1.

Next, in Figure 5.12, 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. 5.13 Plain dependency relations in Dependency and Systemic Functional representation

Figure 5.13 illustrates side by side the parent-daughter and sibling dependency relations in a simplified form. In Figure 5.13a dependencies are the only relations between the units of structure whereas in Figure 5.13b 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 5.13 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 5.6.1–5.6.3. 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 5.6.1 (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 5.10b) have to be accounted in the constituency structure.

Generalization 5.6.2 (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 5.6.2, 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 5.6.3 (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 5.6.3 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).

<table>
<tr><td>some</td><td>very</td><td>small</td><td>wooden</td><td>ones</td></tr>
<tr><td colspan="5">nominal group</td></tr>
<tr><td>Quantifying Determiner</td><td colspan="2">Epithet</td><td>Classifier</td><td>Head</td></tr>
<tr><td></td><td colspan="2">quality group</td><td></td><td></td></tr>
<tr><td></td><td>Temperer</td><td>Apex</td><td></td><td></td></tr>
</table>

Table 5.2 SF analysis of Example 22 (reproduced from Table 3.5)

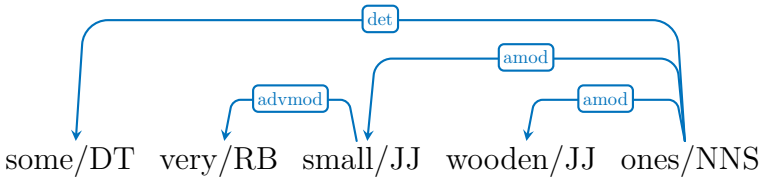


Fig. 5.14 Dependency analysis for Table 5.2

Figure 5.14 and Table 5.2 represent the analysis of a nominal group from Example 22 ("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 5.14. 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 5.13a) can be unpacked into compounding elements of a unit (Figure 5.13b) 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 5.6.1–5.6.3 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 8.3.

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 6

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 9, an account of how to perform SF Transitivity parsing resulting in a configuration of a process, participants and circumstances. For an illustration take Example 47 whose Transitivity analysis is available in Table 6.1. 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.

(47) Albert received a phone call.

(48) He asked to go home immediately.

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

Table 6.1 Transitivity analysis with Cardiff grammar of Example 47

Example 48 is slightly more complex and illustrates the main motivation behind the current chapter. It is analysed in Table 6.2, 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.

<table>
<tr><td>He</td><td>asked</td><td>[empty subject]</td><td>to</td><td>go</td><td>home</td><td>immediately</td></tr>
<tr><td colspan="7">Three Role Cognition configuration</td></tr>
<tr><td>Agent</td><td>Process</td><td colspan="5">Phenomena</td></tr>
<tr><td></td><td></td><td colspan="5">Directional Configuration</td></tr>
<tr><td></td><td></td><td colspan="2">Agent-Carrier</td><td>Process</td><td>Destination</td><td></td></tr>
</table>

Table 6.2 Transitivity analysis with Cardiff grammar of Example 48

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 6.2 briefly introduces the theoretical context of GBT and then formulates the principles and generalisations relevant for current work. Then Section 6.3 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.

6.1 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.

6.1.1 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 49 from (Haegeman 1991: 83).

(49) Poirot will abandon the investigation.

(50) $\Big[_{S}\Big[_{NP}\big[_{N}Poirot\big]\Big]\big[_{AUX}will\big]\Big[_{VP}\big[_{V}abandon\big]\Big[_{NP}\big[_{Det}the\big]\big[_{N}investigation\big]\Big]\Big]\Big]$

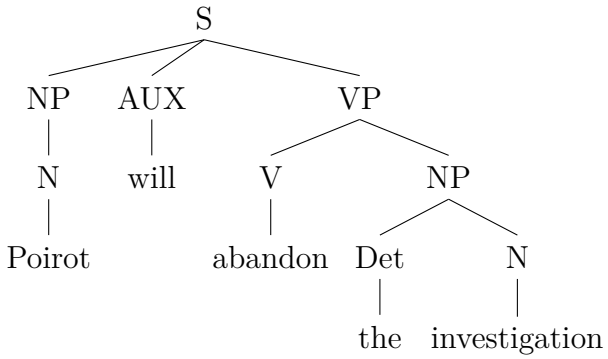


Fig. 6.1 The parse tree of Example 49 from (Haegeman 1991: 83)

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 6.1.1 (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 6.1.2 (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 6.1 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 6.1.1 (Projection principle)**.** Lexical information is syntactically represented.

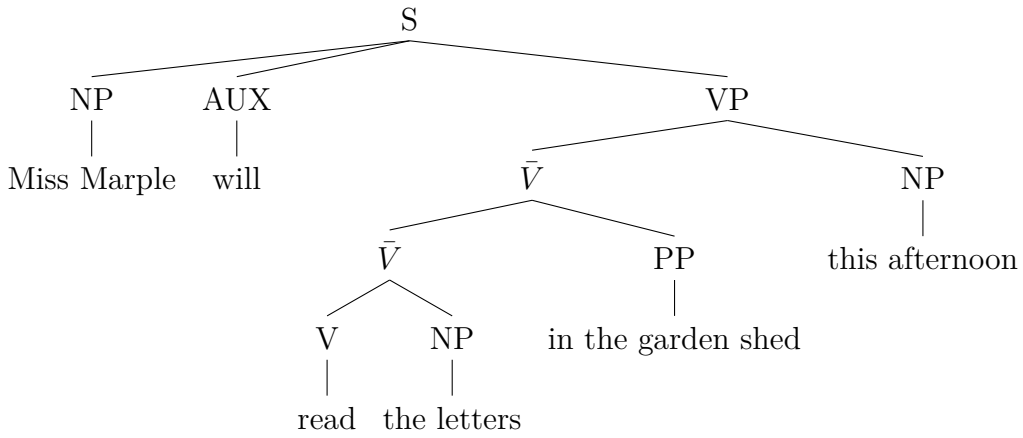


Fig. 6.2 Example of projections from (Haegeman 1991: 90)

An important principle in GBT is that of *projection* formulated in Generalisation 6.1.1. For example in Figure 6.2, 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 6.1.5 below). The role of lexicon in syntax from to GBT perspective is discussed at large in Stowell & Wehrli (1992).

6.1.2 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 5.1, 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 51 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 52 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 53. 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.

(51) P(a, b)

(52) Maigret killed Poirot.

(53) 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 6.1.2.

Generalization 6.1.2 (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)

(54) *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.

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

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

In the Example 55 Maigret has the index i and the burglar j meaning they are distinct referents. On contrary, in Example 56, "he" receives the same index as Maigret because they are interpreted as referring to the same entity. We say that the two are *coindexed*.

6.1.3 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 6.1.3 below and formally in Definition 6.1.5.

Definition 6.1.3 (government i)**.** A governs B if

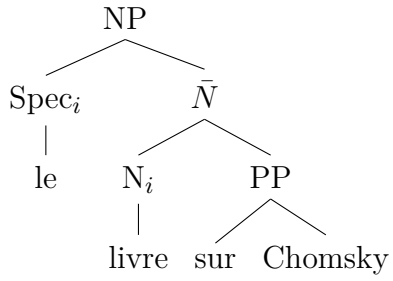
- A is a governor;
- A and B are sisters

Governors are heads (Haegeman 1991: 86).

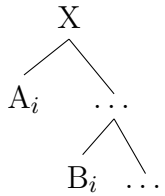
In Figure 6.1 the verb "abandon" is the head of the verb phrase (VP) and governs the direct object - nominal phrase (NP) "the investigation". V does not govern the subject NP "Poirot". All the constituents governed by a node constitute the *governing domain* of that node. In this case VP is the governing 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 6.1.4. 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 6.3b. In Figure 6.3a 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. 6.3 Agreement example and schematic representation

Definition 6.1.4 (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 6.1.5 (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 6.3 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 6.2.1 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 6.1.6. 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 6.1.7) 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 6.1.6 (Binding)**.** A binds B if and only if

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

Definition 6.1.7 (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 6.1.3, 6.1.4 and 6.1.5 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 6.1.3 (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 6.1.4 (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 6.1.5 (Principle C of binding theory)**.** An R-expression (i.e. a NP with independent reference) must be free everywhere (Haegeman 1991: 227).

6.2 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 6.2.1). 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 6.3 Four types of empty categories (adaptation from (Haegeman 1991: 436))

In Table 6.3, [+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 6.2.1 (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.

6.2.1 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 6.2.2 (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 57 the PRO is controlled, thus bound, by the subject "John" of the matrix clause (i.e. higher clause) whereas in 58 it is an arbitrary interpretation where PRO refers to "oneself" or "himself". In 59 and 60 PRO must be controlled by the subject of the higher clause and does not allow for the arbitrary interpretation.

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

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

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

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

Sometimes the controller is the subject (as in Examples 57, 58, 59) and sometimes it is the object (Example 60) 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 6.2.1. 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 6.2.2. 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 6.2.3. 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 6.2.4. 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 61 and 62 illustrate Generalisation 6.2.4

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

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

Generalization 6.2.5. PRO can be subject of complement, subject and adjunct clauses (Haegeman 1991: 278).

Generalization 6.2.6. 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 6.2.7. The object of active clause becomes subject when it is passivized and also controls the PRO element in complement clause (Haegeman 1991: 281).

Generalization 6.2.8. 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 63 and 64 illustrate such marker-free adjunct clauses.

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

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

Generalization 6.2.9. PRO in a subject clause is optionally controlled; thus by default it takes arbitrary interpretation (Haegeman 1991: 283).

(65) PRO_i smoking is bad for the health_j.

(66) PRO_i smoking is bad for your_i health_j.

(67) PRO_i smoking is bad for you_i.

(68) 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 65 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 66, (b) the complement of the higher clause - example 67 and (c) either the possessives in lower or higher clause, Example 68.

6.2.2 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 6.2.4) 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 6.2.3). 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 6.2.3 (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 6.2.4 (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 69 to 72 where I used square brackets to indicate boundaries of an embedded clause. There are two cases of expletives (69 and 71) and their non-expletive counterparts (70 and 72) 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 6.2.4) or as Postal (1974) calls it the *subject-to-subject raising*.

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

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

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

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

The subjects "It" and "Poirot" in none of the examples 69–72 receive a semantic role from the main clause. "It" is an expletive and never receives a thematic role while "Poirot" in 70 and 72 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 6.2.10.

Generalization 6.2.10. 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-word* described in the next section.

6.2.3 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.

(73) [What] will Poirot eat?

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

(75) [Whose pigs] must Wooster feed?

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

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

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

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

Haegeman (1991: 375) offers the Examples 73 – 79 of *Wh-constituents* which are any NPs or PPs that contain a *Wh-word* in their componence. A Wh-word is any of the following words or their morphological derivations (by adding suffixes *-ever*, *-soever*): *who, whom, whose, what, which, why, where, when* and *how*. Thus the wh-constituent can be a single word or a *Wh-phrase*. The Wh-phrase, in GBT, is then the NP or PP which is the maximal projection from a Wh-word. Haegeman treats each Wh-word as the head of the Wh-phrase. This is not quite the case . Below, in Section 6.3.3, 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-words with overt case differentiation. The other Wh-words *what, when, where* and *how* do not change their form based on the case.

Example 80 shows that the Accusative (*whom*) is disallowed when its trace is in Subject position since this requires Nominative (*who*). In example 81 the reverse holds and the Nominative form is disallowed as the Wh-word moved from Complement position requires Accusative case.

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

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

Another important distinction to be made among English Wh-words 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 73 – 75 or in non A-position (i.e. functioning as adjunct) such as in Example **??** and 78.

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 82) or (b) subject position of the embedded clause creating embedded questions (example 83). However regardless of the landing site, the movement principle is subject to what Haegeman describes as *that-trace filter* expressed in Generalisation 6.2.11 and the *Subjacency condition* (Generalisation 6.2.12). Note that the matrix or embedded clauses correspond to the category of inflectional phrase (IP).

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

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

Generalization 6.2.11 (That-trace filter)**.** The sequence of an overt complementizer "that" followed by a trace is ungrammatical (Haegeman 1991: 399).

The examples in 84 to 87 provided by Haegeman (1991: 398) illustrate how the above generalization applies.

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

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

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

(87) Who do you think will arrive first?

Generalization 6.2.12 (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 88 – 90 provided by Haegeman (1991: 403–406). The Wh-movement leaves (intermediate) traces successively jumping each bounding node.

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

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

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

What Generalisation 6.2.12 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 89 and 90.

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.

6.3 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 9.

In this section the grammatical accounts will often overlap. To make a clear distinction I will mark the dependency grammar relations with small caps italic. When the syntactic functions are marked with capital letter (e.g. Subject, Complement, Deictic) then they mean SFG functions and if they are in lower case (e.g. direct and indirect object) then the GBT reading applies. When the (unit/word) classes are mentioned using full word small caps (nominal group, clause,) then the SFL reading shall be applied and when they are all caps acronyms (NN, PP, WH-trace, etc.) the reader shall apply GBT reading .

6.3.1 PRO subject

Coming back to definition of PRO element in Section 6.2.1, 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 6.4.

These complements are always non-finite. Following the principles stated in Generalisation 6.2.1 and 6.2.3 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-words) 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

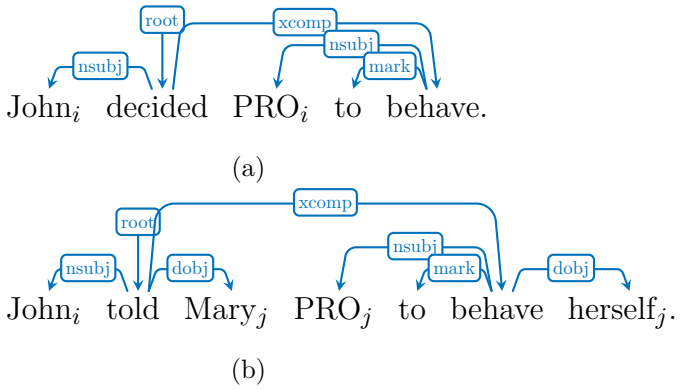


Fig. 6.4 Dependency structure with a PRO subject

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 6.4.

Generalisation 6.2.6 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 91 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 6.3.3 about the Wh-word movement.

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

Based on the above I propose Generalisation 6.3.1 enforcing obligatory control for *xcomp* clauses.

Generalization 6.3.1. 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.

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

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

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

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

The generalization 6.2.7 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 92 and it's passive form 93. 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 94 and 95 the parent clause takes two semantic arguments. Second part of principle 6.2.2 states that in case of obligatory control PRO must be *c-commanded* by an NP. In 94 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 95 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 6.2.6. 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 6.2.7 (Haegeman 1991: 281).

The above can be synthesised into Generalisation 6.3.2 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 6.3.2 presented in Chapter 9.

Generalization 6.3.2. 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 63 ("John hired Mery [PRO to fire Bill]") and 64 ("John abandoned the investigation [PRO to save money]") shall be treated exactly as the non-finite complement clauses are. Generalisation 6.2.8 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 8.1, 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 65 ("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 6.2.9 the PRO is optionally controlled. Since in this case it is not possible to bind PRO solely on syntactic grounds, the generalization 6.2.9 proposes arbitrary interpretation discussed in Section 6.2.1. Next I turn to identifying the second type of null elements (NP-traces) in the dependency structure of a sentence.

6.3.2 NP-traces

Syntactically, NP raising can occur only when there is a complement clause by moving the subject of a lower clause into 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 (explained in Section 6.2.1). In dependency grammar the complement clause without own subject called *open clausal complement* and is introduced via *xcomp* relation.

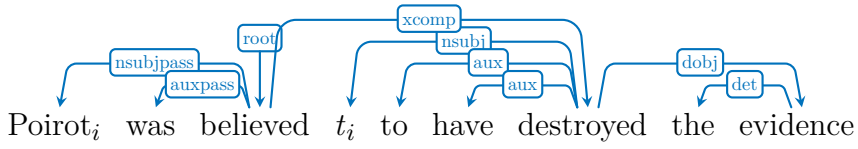


Fig. 6.5 Dependency parse for Example 70

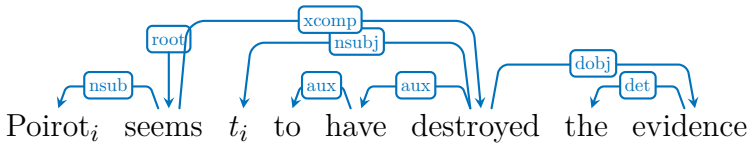


Fig. 6.6 Dependency parse for Example 72

Figures 6.5 and 6.6 represent a dependency parses for Examples 70 and 72. In these examples the subject position of the embedded clause is a NP-trace that is coindexed with the subject position in the clause immediately higher. That subject can be either another NP-trace, in which case they form a chain, or an overt subject.

There are also few exceptions to the above. First, the *xcomp* complement clause, of course, shall not have a subject of its own. Second, it shall not be introduced with the conditional marker "if" or with preposition "for". Third, the higher clause must have no nominal complement between the subject and the embedded complement clause.

Next, if the above conditions are satisfied and the embedded clause follows the *obligatory subject control* pattern, the it is important to distinguish whether the empty subject is a PRO or a NP-trace *t*. Deciding that is not possible using only pure syntactic criteria. The distinction is dictated by the distribution of semantic (participant) roles of each verb (sense) which I discuss next.

Table 6.4 represents the semantic role distribution for the verbs "believe", "seem" and "destroy" employed in the examples above. In the second column is provided the configuration type and the last column indicates the distribution of semantic roles. The role order is normalised such that in a declarative active voice clause the first role is given to the Subject, the second to the complement (direct object) and the last to the next complement (indirect object). The symbol "V" is there to indicate the position of the verb.

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 6.4 Semantic role distribution for verbs "believe" and "seem"

Example 70 is a passive clause with an embedded complement clause. The subjects and complements switch places in passive clauses and so do the semantic roles. In this example, the Cognizant role distributed by the verb believe to Subject position goes to the embedded/complement clause position. However Phenomena is the only

semantic role that can be filled by a clause, all other roles take nominal, prepositional or adjectival groups. This leads us to conclusion that the passive subject Poirot does not receive the Cognizant role from the "believe" frame and in fact, as we will see latter, it may receive semantic role(s) from "somewhere" else.

In Example 72 the verb "seem" assigns roles, as seen in Table 6.4, only to fist and second complements while the subject is left unassigned as indicated by the expletive "it". The embedded clauses, which is the only complement, can fill only the Phenomenon role. This means that the Cognizant role, just like in the previous example, is left unassigned and the Subject "Poirot" does not receive any semantic role from the verb "seem".

The embedded clause in both cases is governed by the verb "destroy" which assigns the Agent role to the Subject and Affected role to the complement. The Subject however has moved from the embedded clause into the Subject position of the higher clause leaving an NP-trace and. The distributed semantic role moves upwards on the chain as well such that the Subject in the higher clause (the antecedent) receives the Agent role from the embedded clause.

Here the problem is to decide what the type of relationship holds between the empty constituent and it's antecedent (subject in the 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.

But this distinction requires analysis of the role distribution of upper and lower clause predicates and deciding the type of relationship between the empty category and its antecedent. If a semantically informed decision shall be made at this stage of parsing or postponed to Transitivity analysis (i.e. semantic role labelling) is debatable because each approach introduces different problem.

Generalization 6.3.3 presents the conditions that need to be verified in order to distinguish the t-trace from the PRO element. I employ here the traditional grammar syntactic features and categories and semantic configurations together with participant roles from Cardiff grammar.

Generalization 6.3.3. To identify 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).

6.3.3 Wh-traces

I now turn to how Wh-movement and relative clauses are represented and behave in dependency grammar and SF grammars. Figures 6.7 and 6.8 present dependency parses for Wh-movement from Subject and Complement positions of lower clause while 6.9 from adjunct position.

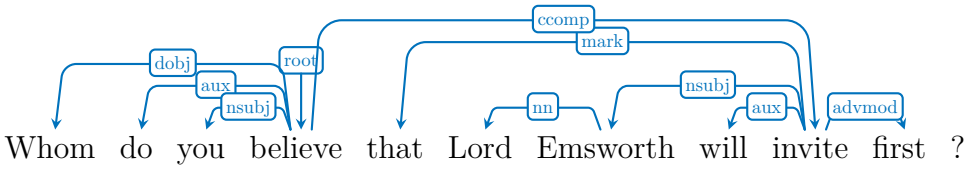


Fig. 6.7 Dependency parse for Example 81

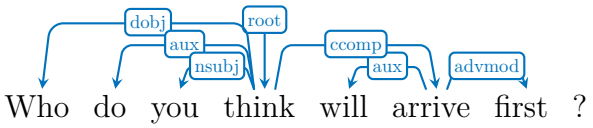


Fig. 6.8 Dependency parse for Example 80

Haegeman (1991: 375) treats each Wh-word as the head of the Wh-phrase. This is not the case in SFG. In some cases, the Wh-words, 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). From these discussions, I have systematised the functional distribution for Wh-words and Wh-groups in Table 6.5 below. This systematisation follows a systemic functional approach as it is more appropriate in discussions about

groups, constituents, their features and functions. This systematisation is useful for scoping the Wh-movement phenomena in SFL terms.

For clarity the Wh-group in Definition 6.3.1. Note that Wh-group is not a new unit class in the grammar but a constituent feature possibly assigned to any of the three unit classes.

Definition 6.3.1 (Wh-group)**.** *Wh-group* is a nominal, prepositional or adverbial group that contains Wh-word either as head or as modifier.

Features	**Clause functions of the Wh-group**		**Group functions of Wh-word**
	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 6.5 Functions and features of Wh-words and groups

Just like in cases of NP-movement, the Wh-groups move only into two and three role cognition, perception and emotion semantic configuration. As mentioned above, the NP-antecedents land in expletive or passive subject position, the Wh-antecedents, by contrast, land in subject or subject preceding position functioning as subject, complement or adjunct depending on the Wh-word (or Wh-group).

The essential features for capturing Wh-movement in dependency graphs are (a) the finite complement clause identified by *ccomp* relation between the matrix and embedded clause (b) the Wh-word/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. 81), Complement (e.g. 80) or Adjunct. *ccomp* relation is defined in Marneffe & Manning (2008a) to introduce complement clauses with an internal subject which are usually finite.

Regardless whether the syntactic function of the traces in the lower clause is Subject or Complement, in the higher clause the Wh-group takes Complement 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 involved. 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.

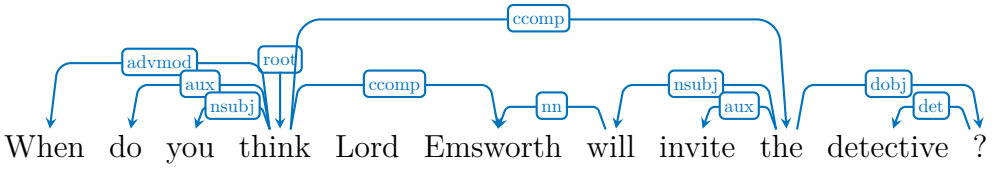


Fig. 6.9 Example dependency parse with Adjunct Wh-word

In case of WH-traces with Adjunct function in the lower clause, as shown in Figure 6.9, their antecedents also receive Adjunct functions in the higher clause. One peculiar finding about Adjunct Wh-trace is that it cannot bind, at least in English, to the clause with a (generic) present simple tense. The clause must be bound in some other tense or modality than present simple. The reader can experiment with changing tense in the above example to test this.

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

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

Not always are the Wh-groups movements from a lower clause. It is possible that the trace of the moved element resides in the higher clause (complement) or even to have a case of no movement when a Wh-word takes the Subject function in the higher clause. Examples 96 and 97 are good examples of a *short* movement (in clause) of this kind (similar to relative clause Wh-traces described in Section 6.3.4 below). However the short movement cannot be grasped in terms of dependency grammar because order is the only factor that counts and dependency grammar is order free and does not (explicitly) account for it. The functions are already assigned accordingly so short movement is not a subject of interest for the current section.

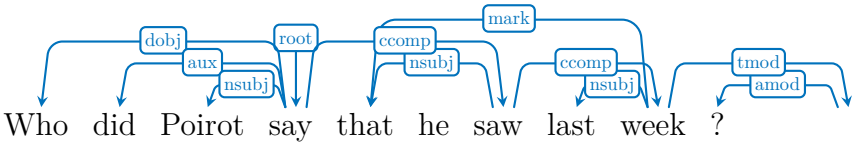


Fig. 6.10 Dependency parse for Example 90

Recall the *cyclic* and *successive* properties of Wh-movement from the previous section underlined by Example 90 and its dependency parse in Figure 6.10. GBT suggests that the Wh-movement leaves traces in all the intermediary clauses. In dependency grammar these properties are treated instrumentally for determining intermediary wh-traces in the search for the foot of the chain and none of the intermediary traces shall be created as null elements. There is no further purpose for them as they do not receive a thematic role in the intermediary clauses.

6.3.4 Wh-traces in relative clauses

In GB theory the Wh-words that form relative clauses (*who, whom, which, whose*) are considered moved. In dependency grammar such movement is redundant since the Wh-word and its trace are collapsed and take the same place and function. The Wh-words 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 an anaphoric indexing relation to a noun it refers to.

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 a nurse.
(*any of the above in prepositional group*)	person/ non-person	thing/possessor in Subject	... the boy to whom I gave advice. ... the cause for which we fight.
that	person/ non-person	Subject	... the apple that lies on the table.
Zero Wh-word	person/ non-person	Subject	... the sword sent by gods.

Table 6.6 The Wh-words introducing a relative clause.

Focusing now on the relative clauses, there are three more possible constructions that introduce them: (a) a prepositional group that contains a Wh-word, (b) "that" complementizer which behaves like a relative pronoun and (c) the *Zero Wh-word* which is a Wh-trace empty element and which functions the same way as an overt Wh-word. Table 6.6 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-words to their antecedents.

Relative clauses in dependency graphs are introduced by *rcmod* and *vmod* relations. The *rcmod* introduces relative clauses containing a Wh-group while the *vmod* introduce finite and non-finite relative clauses with Zero Wh-word. So the *vmod* dependency relation suffices to signal the empty element.

The Zero Wh-word behaves exactly like the *PRO element* in the case of non finite complement clauses discussed in Section 6.2.1. It receives thematic roles in both the higher clause and in the lower clause and is not a part of a chain like the cases of NP/Wh-movement.

6.4 Discussion

This chapter has treated the identification of the *null elements* in syntactic structures as this was identified as one of the major places where a mismatch between SFG and the structural requirements for Transitivity analysis. Section 6.2 presented how GBT theory handles null elements and then Section 6.3 showed how the same principles translate into Stanford dependency graphs involving also, for guidance, a few references to SFL constituency structure.

This chapter also contributes to establishing cross-theoretical connections that are among primary objectives of the current thesis. Specifically it provides translations of necessary principles and generalizations from GB theory into the context of dependency grammar. These results will be used directly in the parser implementation described in Section 9.3.

Identification of null elements will be important below for the semantic role labelling process described in Chapter 9 because usually the missing elements are participant roles (theta roles) shaping the semantic configuration. Therefore to increase the accuracy of semantic parsing, spotting null elements is a prerequisite.

Chapter 7

On Graphs, Feature Structures and Systemic Networks

The parsing algorithm, whose pipeline architecture we have seen in Section 1.7.4, 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 following chapters which detail on the parsing process. This chapter 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 requirements arose that can be summarised as follows:

- graphic representation
- 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) but 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 (handling uncertainty)

- conjunctive descriptions (handling multiple feature selections)
- (conditional) pattern specifications (i.e. define patterns of graphs)
- operational pattern specifications (i.e. a functional description to be executed in pattern graphs)

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

7.1 On sets, feature structures and graphs

In the field of computational linguistics trees has been taken as the de facto data representation. In Section 1.7.3 I have mentioned already that I employ graph and not tree structures.

Firstly, the trees are a special kind of graphs. Anything expressed as a tree is as well 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 Section 7.4. This expressiveness is needed when dealing with interconnection of various linguistic theories which in practice is done by mapping the nodes of one tree structure onto the nodes of another one. In addition, the structures are not always trees. There are situations when a node has more than one parent or when a node is connected to its siblings which break the tree structure.

Definition 7.1.1 (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 7.1.2 (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 $u, v \in V$)

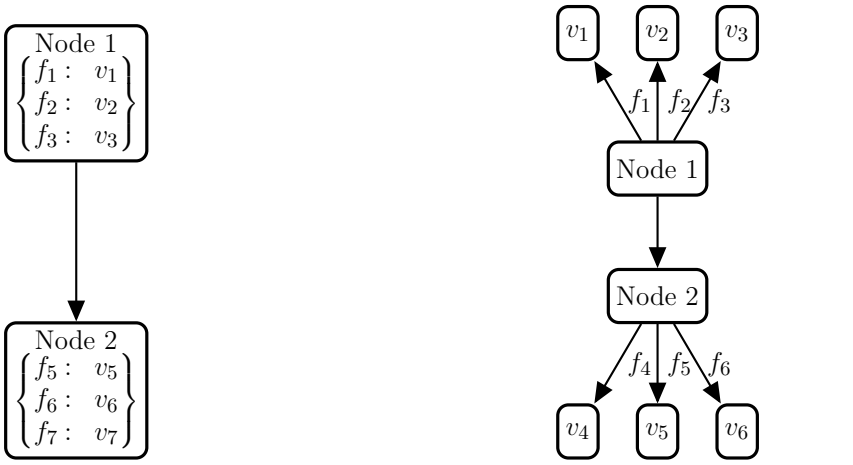
In this thesis the graph nodes are considered to be *feature structures* forming *Feature Rich Graphs* (see Definition 7.1.10). Before formally defining these graphs, I need to address first the notion of feature structure and a few kinds of sets.

In SFL the concept of *feature* takes up an important role. Also features are said to form systems of choices that are structured in relation to one another and are suitable for describing linguistic objects and phenomena.

Pollard & Sag (1987) have formally described useful concepts for grammatical representations in the context of Head-Driven Phrase Structure Grammar (HPSG). He adopts the *typed feature structure theory* and extends it in ingenious ways applicable in computational linguistics. Among others, he provides formal definitions for the concepts of *feature structure*, *hierarchy*, *logical evaluation*, *composition* and *unification*, the latter, being key operations in parsing using feature structured grammars.

In this thesis, feature structures are important but only in a simplified version serving as graph node descriptions. The main reason is the difference in approach as the main parsing operations, here, are based on graph pattern matching (introduced in the sections below).

In a broad computer science sense, including Pollard's definition, feature structures are equivalent to graph structures. So any feature structure can be expressed as a graph and any graph can be expressed as a feature structure. But in a narrow sense, as adopted in this thesis, it is useful to employ both concepts but each for a given purpose. The feature structure is reduced to an attribute-value matrix (see Definition 7.1.3) and the graphs to a network of feature structure nodes (see Definition 7.1.10) i.e. no atomic nodes.



(a) The graph with feature structure nodes

(b) The graph with atomic nodes

Fig. 7.1 Graphs with atomic nodes and feature structure nodes

The main reasons in this separation are efficiency and practicality. First, it is about handling the atomic values (strings or integers) and (ordered) arrays only as values of feature structures and never as graph nodes. Second, the graphs remain limited in

size, close to the conceptualised linguistic structures, i.e. dependency or constituency. Otherwise, the graphs would grow in complexity (a) by at least one more round of nodes for each dependency or constituency node and (b) by adoption of an additional node classification.

For example lets imagine a constituency graph fragment of two nodes *Node 1* and *Node 2* where each has three associated features as it can be seen in Figure 7.1a. If we would insist to dispose of the feature structure within the node and express the features as atomic graph nodes then the result would be a graph structure such as the one in Figure 7.1b.

Definition 7.1.3 (Feature Structure (FS))**.** A *feature structure* F is a finite set of attribute-value tuples $f_i \in F$. A *feature* $f = (a, v)$ is an association between an identifier a (a symbol) and a value v which is either an atomic value (symbol, number, string), a set or another feature structure.

The values of feature structures may be other feature structures allowing, if needed, to construct hierarchical descriptions. In the current implementation, however, the values of the feature structure are restricted to atomic values or sets of values.

For convenience I define two functions to access the identifier and value in a feature structure. The function $name : F \rightarrow symbol$ returns the feature symbol (identifier) $name(f) = a$ and the function $val : F \rightarrow \{atomic, Set, FS\}$ is a function returning the ascribed value of a feature $val(f) = v$.

Definition 7.1.3 stipulates that the value of a feature may be also a set (besides an atomic value). The sets used in this thesis need to carry additional properties required for their interpretation. Specifically, it is the order need to be addressed here and the capacity to specify that set elements stand in a certain logical relation one to another (e.g. conjunction, disjunction, negation, etc.). These two properties are covered in Definition 7.1.4 and 7.1.5. For convenience I will assume from now on that sets (see Definition 7.1.4) preserve order even when it is not really required.

Definition 7.1.4 (Set)**.** An (ordered) *set* $S = \{o_1, o_2, ..., o_n\}$ is a finite well defined collection of distinct objects o_i. A set is said to be ordered if the objects are arranged in a sequence such that $\forall o_{i-1}, o_i \in S : o_{i-1} < oi$.

Definition 7.1.5 (Conjunction Set)**.** A *conjunction set* $S_{conj} = (S, conj)$ is a 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 7.1.6 (Conjunctive set)**.** Conjunctive set (also called *AND-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 7.1.7 (Negative conjunctive set)**.** Negative conjunctive set (also called NAND-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 7.1.8 (Disjunctive set)**.** Disjunctive set (also called OR-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 7.1.9 (Exclusive disjunctive set)**.** Exclusive disjunctive set (also called XOR-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 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.

The function $\tau(S)$, defined $\tau : S \to \{S_{AND}, S_{OR}, S_{XOR}, S_{NAND}\}$, returns the type of the conjunction set and the function $size(S)$, defined $size : S \to \mathbb{N}$, returns the number of elements in the set. The size function is also denoted as $|S|$.

Now that all the necessary basic notions nave been formally defined I now define the feature rich graph and provide a couple of examples afterwards.

Definition 7.1.10 (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, for convenience, when I refer to a graph I will refer to a feature rich digraph unless otherwise stated. The parsing algorithm operates with such graph and they are further distinguished, based on purpose as: *Dependency Graphs* (DG)

(example figure 7.2), *Constituency Graphs* (CG) (Figure 7.3) and Pattern Graphs (PG) also referred as *Query Graphs* (QG).

Please note that the edges are defined to carry feature structures. This capacity will not be employed, for example, in the case of constituency graphs; only minimally employed in the case of dependency graphs, where the dependency relation is specified; and fully employed in pattern graphs. Nonetheless, treating all of them as feature rich graphs simplifies the implementation.

Definition 7.1.11 (Dependency Graph)**.** A *dependency graph* is a feature rich digraph whose nodes V correspond to words, morphemes or punctuation marks in the text and carry at least the following features: *word*, *lemma*, part of speech (*pos*) and, when appropriate, the named entity type (*net*); the edges E describe the dependency relation (*rel*).

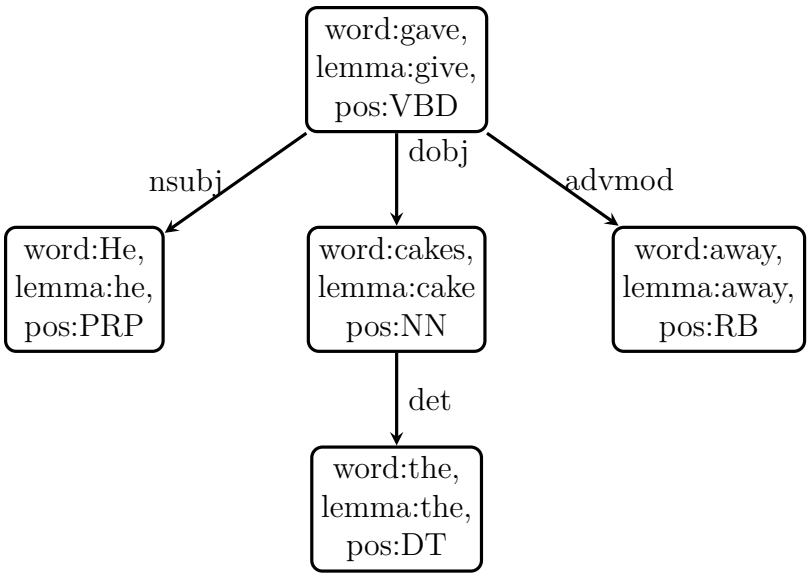


Fig. 7.2 Dependency graph example with FS nodes and edges

Definition 7.1.12 (Constituency Graph)**.** A *constituency graph* is a feature rich digraph whose nodes V correspond to SFL *units* and carry the *unit class* and the element function within the parent unit (except for the root node); while the edges E represent constituency relations between constituents.

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 (as described in the discussion of 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 not have a parent unit. The leaf nodes carry the same features as the DG nodes plus the word class feature which correspond to the traditional part of speech tags.

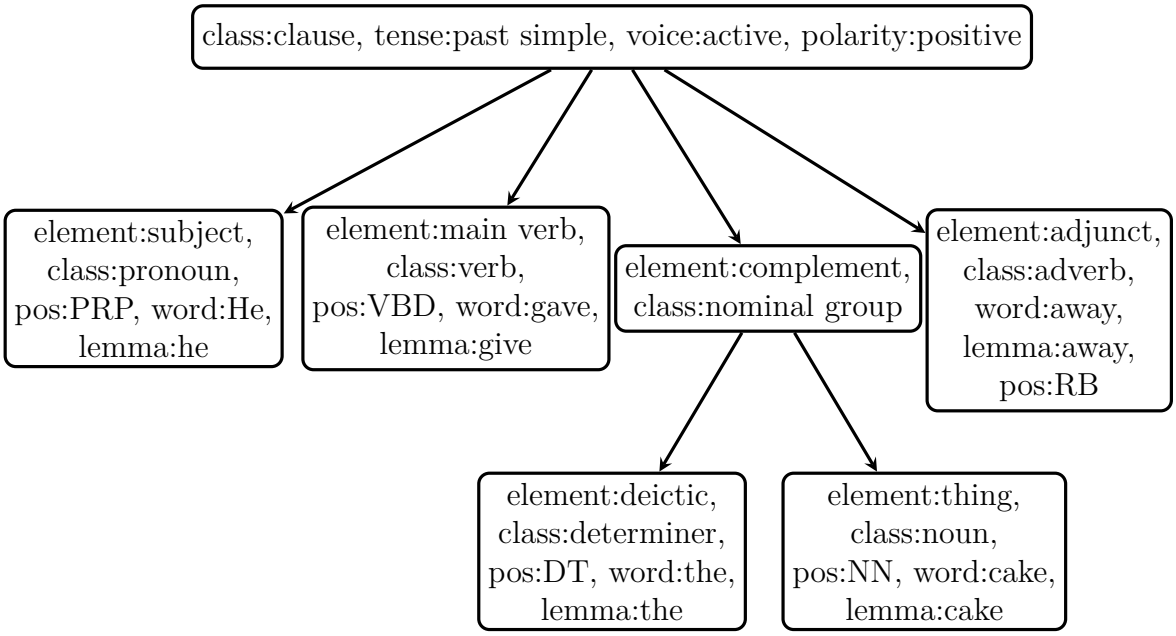


Fig. 7.3 Constituency graph example

Apart from the essential features of class and function, the CG nodes carry additional class specific features selected from the relevant system network. The features considered in this thesis are described in Chapter 4. In addition, the leaf CG nodes contain the features of dependency graph nodes enumerated in Definition 7.1.11. The way CG is enriched with features is described in the next chapter. Below in Figure 7.3, is an example CG that carries tense, modality and polarity features on the clause node in addition to class and element function.

7.2 Graph traversal

From the general set of operations on graphs defined in graph theory (Bondy et al. 1976; West et al. 2001) graph traversal in particular is of importance for the current work. It is used for the constituency graph creation step presented in the parsing pipeline from Figure 1.8 (in Section 1.7.4). The constituency graph is created throughout the traversal of a dependency graph. Traversal is used in this thesis for dependency graphs only. Next I address this operations in detail.

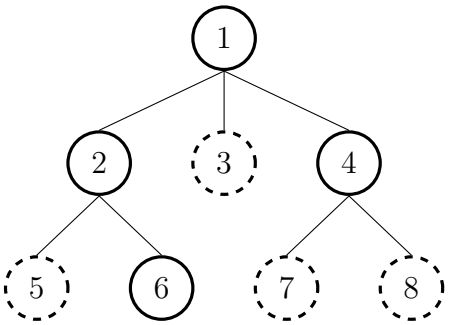


Fig. 7.4 Sample graph with numbered node of two types

The *graph traversal* defined in 7.2.1 and especially its conditional and generative extensions defined in Definition 7.2.2 and 7.2.3 are important operations in this work. They are employed in the first phase of the algorithm, for copying the dependency graph as constituency graph described in Chapter 8.

Definition 7.2.1 (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 neighbouring each other in either *breadth first* (t_{BF}) or *width first* (t_{WF}) orders.

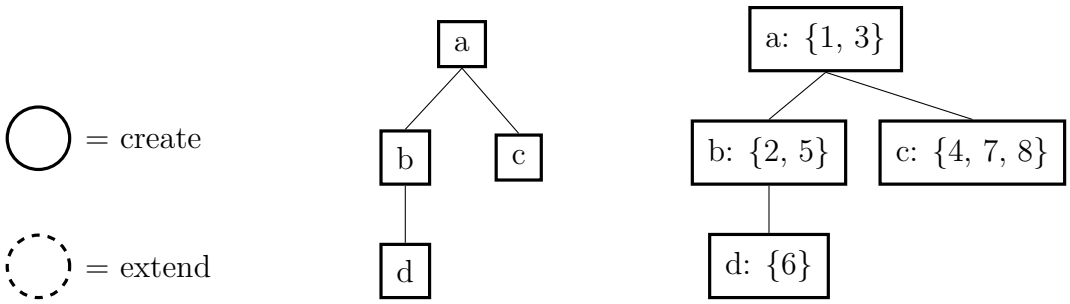
The graph traversal is employed either for searching for a node or an edge or finding a sub-graph that fulfils certain conditions on its nodes and edges if it is a conditional traversal. For example in the semantic enrichment phase (that will be described in Section 9.2), to ensure that the semantic patterns are applied iteratively to each clause, from a multi-clause CG graphs are selected all clause sub-graphs without including the embedded (dependent) clauses.

Definition 7.2.2 (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.

One of the potential complete traversals for the graph from Figure 7.4 starting from the node 1 is {1, 2, 3, 4, 5, 6, 7, 8} using breadth first strategy or {1, 2, 5, 6, 3, 4, 7, 8} for depth first strategy. On the other hand, a conditional traversal of non-dashed nodes staring from the node 1 results in {1, 2, 4, 6}, {1, 4, 2, 6} or {1, 2, 6, 4}. The first two traversals corresponding to the width first strategy and the third one to the depth first strategy.

I also use the graph traversal to execute generative operations on a parallel graph, which is a special case of *graph rewriting*. For example DG traversal is employed for bootstrapping (i.e. creating in parallel) the CG as it was previously motivated in Section 1.7.4.

Definition 7.2.3 (Generative Traversal)**.** Generative traversal $g(M,G)$ of a source graph G via a operation matrix M is an operation resulting in the creation of the target graph H by contextually applying generative operations to bring the latter into existence. The operation matrix M is a set of tuples (ctx,o,p) that link the visited source node context ctx (as features of the node, the edge and previously visited neighbour) to a certain operation o that shall is executed on the target graph H with parameters p.



(a) The rule set (b) The target (c) The explained target

Fig. 7.5 The generative traversal result for Figure 7.4 using create and extend operations

Next I provide an rough description of what happens when a generative traversal is executed and the exact algorithm will be described in detail in Section 8.3. For example lets assume that only two types of operation are needed for our task at hand. First is to create a new node on the target graph once a non-dashed node is visited on the source graph. And, second, is to pass without doing anything the dashed nodes. This is schematically represented in Figure 7.5a. Lets now apply these operations on traversing the example graph using breadth first strategy following the order provided above {1, 2, 3, 4, 5, 6, 7, 8}. The traversal graph is considered source graph and the target graph is empty at the beginning of the process. Upon visiting the node 1 a first node is created on the target graph which is labelled *a*. When traversing nodes 2 and 4 then each of them signal creation of the nodes *b* and *c* as children of *a* in the target graph and correspondingly node 6 signals creation of node *d*. The final target graph is depicted in Figure 7.5b and in Figure 7.5c the source nodes are embedded into the

target node to make explicit that the non-dashed nodes (i.e. 3, 5, 7, 8) are simply passed over without any generative operation.

Now that generative traversal is defined, 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. In this work, however, these operations are not used and therefore omitted here.

In Section 7.1 the last two definitions were for the constituency and dependency graphs. They are used in this thesis to represent grammatical analysis of of a sentence. Next we will look at a special type of graph which represents fragments of structure repeatable across multiple analysis. They represent generalisations or patterns that usually are associated with grammatical features or a set of features which I explain in Chapter 8. These graphs are called *pattern graphs* and the next section is dedicated to them.

7.3 Pattern graphs

Regardless of the type, constituency or dependency, the parsing process (which will be described in Chapter 8) relies on identifying patterns in graphs. The patterning is described as both graph structure and feature presence (or absence) in the nodes or edges. The *pattern graphs* (defined in 7.3.1) are special kinds of graphs meant to represent small (repeatable) parts of parse graphs that, in the context of the current work, are used to identify grammatical features.

The pattern graphs contrast with the *parse* (or *instance*) graphs which are either constituency or dependency graphs. The parse graphs express what is an actual analysis of a text, i.e. representing what is the case, whereas the pattern graph expresses a potential that could be the case in the instance graph. This way the pattern graphs have a prescriptive interpretation whereas the instance ones taken a factual interpretation.

Definition 7.3.1 (Pattern Graph)**.** A *pattern graph* (PG) is a feature rich graph for expressing regularities in the node and edge structure.

Next I discuss two example of pattern graphs. One example shows a pattern graph encoding the present perfect continuous tense, which traditional grammar defines as in Table 7.1. Afterwards, the second example will show how the notion of linear succession among nodes is accounted for in the pattern graphs for declarative and interrogative mood.

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

Table 7.1 Present perfect continuous tense

Examples 98–100 show variations of present perfect continuous tense in a simple clause according to declarative and interrogative mood and the "has" contraction. Of course there are more variations possible for example according to voice (active and passive) but they are omitted here because they adds combinatorially to the number of examples and the ones provided already serve their purpose here. The Figures 7.6-7.8 represent corresponding dependency parses for these examples (generated with Stanford dependency parser).

(98) He has been reading a text.

(99) He's been reading a text.

(100) Has he been reading a text?

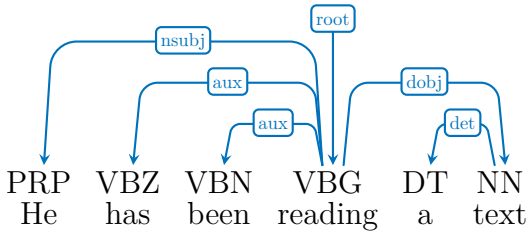


Fig. 7.6 Present perfect continuous: indicative mood, un-contracted "has"

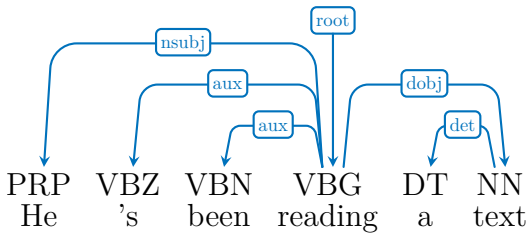


Fig. 7.7 Present perfect continuous: indicative mood, contracted "has"

The present perfect continuous tense can be formulated as a pattern graph (including voice) over the dependency structure as illustrated in Figure 7.9. In this pattern the main lexical verb is *present participle* indicated via the *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

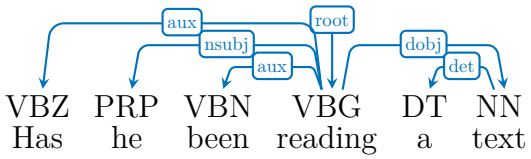


Fig. 7.8 Present perfect continuous: interrogative mood, un-contracted "has"

person. Also the *to be* can be either connected by the *aux* relation or in case of passive form by the *auxpass* relation. Note that the pattern in Figure 7.9 constraints the edge type (using an OR-set) 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*.

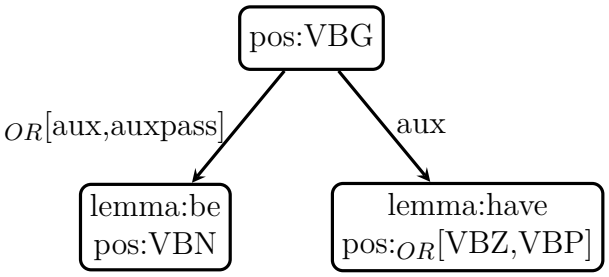


Fig. 7.9 The graph pattern capturing features of the present perfect continuous tense

One of the fundamental features of language is its sequentiality and directionality. This aspect is not inherent in graphs. In the simplest form, they just describe connections between nodes and are agnostic to any meaning or interpretation. Next, I introduce the way I deal with linear order in the pattern graphs.

Lets consider the clause mood and encode the distinction between *declarative* and *Yes/No interrogative* moods. In SFG this features is described in terms of the relative order of clause elements. If the finite is before the subject then the mood is Yes/No-interrogative, whereas when the finite succeeds subject then the mood is declarative. Example 100 contrasts in mood with 98 and 99.

Order can be specified in absolute or relative terms and partially or exhaustively. In order to cover these three kinds of constraints, I introduce three special features: the node *id*, *precede* and *position*. Node id takes a token to uniquely identify a node in the graph, the precede feature takes an ordered set to indicate the (partial) linear precedence to other node ids, and the position feature indicates the absolute position of a node.

One way to introduce order among nodes is then by marking them with an absolute position. This is a good method applicable to parse graphs. The DGs and CGs, are automatically assigned at creation time the absolute position of the node in the

sentence text via the feature *position*. This feature is present in the leaf nodes only and corresponds to the order number in which they occur in the sentence text while the non-leaf node's position is considered to be the lowest position of its constituent nodes. The absolute position description is rarely used in the PGs. The only cases are to state that the constituent is first or last position in a sentence.

Another way to specify node order is through relative precedence, for which the node id and the precedence features are introduced above. This is the preferred method to provide linear precedence dimension in pattern graphs. It is also relative so the specification can be partial. With this method a node specifies that it precedes a set of other nodes.

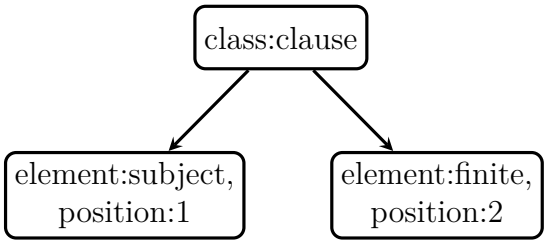


Fig. 7.10 Declarative mood pattern graph with absolute element order

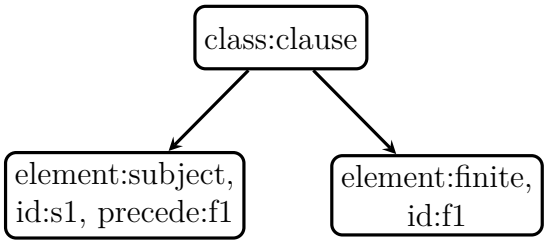


Fig. 7.11 Declarative mood pattern graph with relative element order

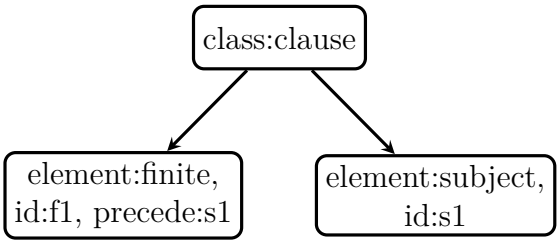


Fig. 7.12 Pattern graph for Yes/No interrogative mood

To continue the example of mood features, I illustrate in Figures 7.10 and 7.11 the use of relative and absolute node ordering constraints for declarative mood; and in

Figure 7.12, I depict the PG for the Yes/No interrogative mood. In both cases I use the relative node ordering.

Patterns like the ones explained above can be created for a wide range of grammatical features. Once the grammatical feature is encoded as a pattern graph it can be identified in parse graphs (DG or CG) via *graph pattern matching* operation described in Section 7.4. Moreover, once the pattern is identified it can act as a triggering condition for the node update operation. When the pattern is matched then additional features can be added to the nodes of the parse graph thus the mechanism of enriching the parse graph. Coming back to out tense example above, once the pattern 7.9 is identified then the clause can be marked with the tense feature. In the next section I address the graph matching operation and then show how it works using pattern graphs.

7.4 Graph matching

So far we have discussed about constituency and dependency graphs and, in the last section, I introduced pattern graphs. The intuition behind pattern graphs is that they are meant to be matched against or found in other graphs. The patterns graphs can be viewed as small reusable modules and as generalizations consisting of structural patterns. I will address next what does it mean for two graphs to be the same, i.e. *isomorphic* and how this sameness checking is used in the current work.

In *mathematics* the structure-preserving mappings $f : X \to Y$ (from one object X to the other Y) of the same type is called *morphism*. The morphism $f : X \to Y$ is called *isomorphism* if there exists an inverse morphism $g : Y \to X$ such that $f \circ g = id_X$ and $g \circ f = id_X$.

In computer science, the *graph matching* operation is known (sub-)graph isomorphism. Two graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$ are isomorphic if there exists a mapping from the nodes of graph G to the nodes of graph H, such that the edge neighbourhood is preserved. In such context the nodes are unique atomic symbols functioning as node identifiers.

Definition 7.4.1 (Graph matching)**.** For two graphs G and H, where $G \leq H$, *graph matching* is the operation of finding an isomorphism between G and H.

Definition 7.4.2 (Graph isomorphism)**.** An isomorphism of graph $G = (V_G, E_G)$ and $H = (V_H, E_H)$ is a bijective function $f : V_G \to V_H$ such that if any two nodes $u, v \in V_G$ from G are adjacent $(u, v) \in E_G$ then $f(u), f(v)$ are adjacent in H as well $(f(u), f(v)) \in E_H$.

Graphs do not always have the same number of nodes (or edges). We say that a graph is smaller than another one, denoted $G \leq H$, when its number of nodes is less than that of the other graph. In such cases, for two graphs G and H where $G < H$, the (sub-)graph matching task is redefined to establishing isomorphism(s) from G to a sub-graph of H.

Definition 7.4.3 (Sub-graph isomorphism)**.** Given two feature rich graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$, G is sub-graph isomorphic to G (denoted $G \subseteq H$) if there is an injective function $f : V_G \to V_H$ such that

- $\forall v \in V_G, f(v) \in V_H$ and
- any two nodes adjacent in G, $(u, v) \in E_G$, are also adjacent in H, $(f(u), f(v)) \in E_H$

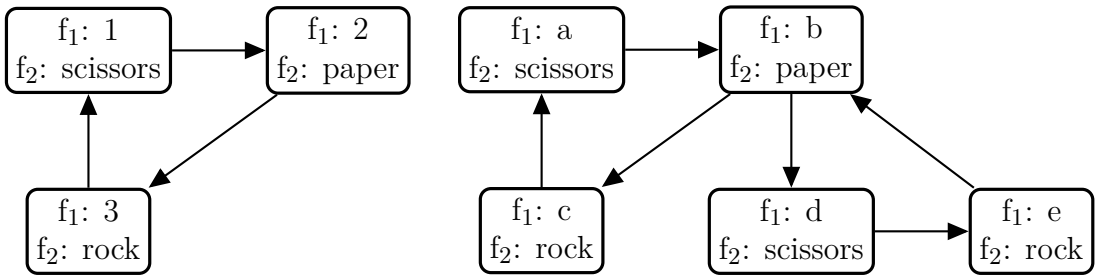
(a) The pattern graph (b) The target graph

Fig. 7.13 Sub-graph isomorphism {1=a, 2=b, 3=c}

In Figure 7.13a is depicted a labelled graph that is isomorphic to a sub-graph in Figure 7.13b. The example graphs presented in Figure 7.13 have atomic labels as nodes and the isomorphism is established as a mapping between labels. Section 7.1 above mentions that the graph considered in this thesis have feature structures as their nodes and not atomic nodes. But in case of feature rich graphs additional rules how to establish the isomorphism need to be provided because there are multiple ways to approaching it.

Lets look at Figure 7.14 where the graph nodes are feature structures using features: f_1 and f_2. One way to approach isomorphism in this scenario is by the value of one feature, for example f_1. Then we can identify two sub-graph isomorphisms: {1=a, 2=b, 3=c} and {1=b, 2=d, 3=e}. This approach, besides additional specification what values to compare, i.e. f_1s, is the same as providing a sub-graph isomorphism on the labelled graphs from Figure 7.13.

In addition to the rule above, lets add a constraint that the isomorphism is not only a mapping between the feature values (numbers to letters) but also that the mapped values are identical (strict value equality). If we consider the strict equality rule applied



(a) The pattern graph (b) The target graph

Fig. 7.14 An example of rich sub-graph isomorphism

on f_1 feature, there is no isomorphism between the two graphs because first one uses numbers {1, 2, 3} and the second uses letters{a, b, c, d}. Now if we turn to the values of f_2 and apply the same rule then there is one isomorphism possible {paper=paper, rock=rock, scissors=scissors}. The second one, even if it is a cycle, {paper=paper, rock=scissors, scissors=rock} is no longer acceptable because the "scissors" and "rock" switched places in the target graph and it would have been acceptable as a mapping, but not as strict value equality. Formally, the additional equality constraint can be expressed on the graph isomorphism f as $u = f(u)$.

This brings us to the idea that, in the feature rich (sub-)graph isomorphism, we need to introduce a *matching* operator (denoted $\succcurlyeq$) for nodes. This means that we no longer can use atomic symbol mapping but have to employ the matching operator. The node matching operation is defined on feature structures. We say that a feature structure may match once, several times or not at all another feature structure, $FS_1 \succcurlyeq FS_2$. This intuition is expressed in Definition 7.4.6. The sub-graph isomorphism over the feature rich graphs is captured in Definition 7.4.4 below.

Definition 7.4.4 (Rich sub-graph isomorphism)**.** Given two feature rich graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$ and a matching relation $\succcurlyeq$, G is sub-graph isomorphic to H if there is an injective mapping $f : V_G \to V_H$ such that

- each node in V is mapped to exactly one node in H, $\forall v \in V_G, f(v) \in V_H$ and
- each node in G matches with its correspondent in H, $\forall v \in V_G, v \succcurlyeq f(v)$ and
- any two nodes which are adjacent in G, are also adjacent in H, $\forall (u,v) \in E_G, \exists \Big(f(u), f(v)\Big) \in E_H$

We already mentioned in the introduction of this section that pattern graphs are considered as generalizations over parse graphs (constituency or dependency). The

pattern graphs are smaller and their nodes have fewer features specified. The parse graphs have more nodes and features. We say that a parse graph *instantiates* a pattern graph if there exists a rich sub-graph isomorphism. This operation is called *pattern graph matching* and is defined in Definition 7.4.6 below.

Definition 7.4.5 (Pattern graph matching)**.** Given a pattern graph G and an instance (parse) graph H (either dependency or constituency), *pattern graph matching* is the operation of finding a rich sub-graph isomorphism from G to H such that each pattern node matches its corresponded instance node each H, $\forall v \in V_G, v \succcurlyeq f(v)$.

As mentioned before, nodes of the parse and the pattern graphs are feature structures. I approach the matching between them in two steps: first, matching the feature names in Definition 7.4.6, and second, matching the feature values in Table 7.2. In order to simplify and make explanations clear, I will further refer to the feature structures that constitute nodes in the pattern graphs as *pattern feature structures* and the feature structures that constitute nodes in the instance graphs as *instance feature structure*.

Definition 7.4.6 (Feature structure matching)**.** A pattern feature structure V matches an instance feature structure U if and only if every feature in V has a corresponding feature U and their values match; that is $\forall f_V \in V, \exists f_U$ such that $name(f_V) = name(f_U)$ and $val(f_V) \succcurlyeq val(f_U)$.

According to the Definition 7.1.3, the values of feature structures can be either atomic (numbers, strings, symbols, etc.), denoted *atomic*, or one of the conjunction sets: S_{AND}, S_{OR}, S_{XOR} and S_{NAND}. For simplicity, the option of nested feature structures is excluded in the current work even though it is perfectly viable configuration. Consequently, the matching relation takes into consideration the *type* of the compared elements in addition to how they relate to each other, including comparisons between set and atomic values. Please note that this relation is *not symmetric* i.e. *not commutative* because the subsequent relations used in the definition i.e. set inclusion and set subsumption are not symmetric. This means that $A \succcurlyeq B \neq B \succcurlyeq A$.

I will remind here that, the function $\tau(S)$, defined in Section 7.1, returns the *type* of the feature value. I extend its definition here to handle also atomic data types as follows $\tau : x \rightarrow \{atomic, S_{AND}, S_{OR}, S_{XOR}, S_{NAND}\}$. The matching rules have to be defined for each possible pair of types returned by the function τ yielding 25 possibilities. Table 7.2 provides matching relations for each pair of types where the rows represent value types of the pattern features, denoted $\tau(v)$, and the columns represent value types of the instance features, denoted $\tau(u)$. Each table cell contains a comparison expression

returning a truth value. Cells with a bottom sign $\perp$ mean that there can be no match between these types no matter the values.

$\tau(v) \backslash \tau(u)$	*atomic*	S_{AND}	S_{OR}	S_{XOR}	S_{NAND}
atomic	$v = u$	$v \in u$	$\perp$	$\perp$	$v \notin u$
S_{AND}	$\perp$	$v \subseteq u$	$\perp$	$\perp$	$\perp$
S_{OR}	$v \ni u$	$v \cap u \neq \varnothing$	$v \supseteq u$	$v \supseteq u$	$\perp$
S_{XOR}	$v \ni u$	$\perp$	$\perp$	$v \supseteq u$	$\perp$
S_{NAND}	$v \not\ni u$	$v \cap u = \varnothing$	$v \cap u = \varnothing$	$v \cap u = \varnothing$	$v \subseteq u$

Table 7.2 Strict matching between pattern and instance feature values organised by value type

For example, if both values are of atomic type then in order to match they have to equal. If the $\tau(v)$ is atomic and the $\tau(u)$ is an AND set then v needs to be among the set of values constituting u; whereas if the $\tau(u)$ is an OR or XOR set then these values do not match, designated by the bottom sign $\perp$. The same reading applies to the rest of the table for each pair of value types.

A more permissive matching is defined in Table 7.3. Here, on the instance feature values, the two types of disjunction (OR and XOR) and the negated conjunction (NAND) are interpreted as possibly matching and provided with the corresponding relation whereas in the previous definition these cases were completely excluded. The permissive match is rarely used in this work but it is nevertheless useful cases of instance graphs where the feature values could not be are assigned with a certainty but as a disjunction of either one or the other.

$\tau(v) \backslash \tau(u)$	*atomic*	S_{AND}	S_{OR}	S_{XOR}	S_{NAND}
atomic	$v = u$	$v \in u$	$v \in u$	$v \in u$	$v \notin u$
S_{AND}	$\perp$	$v \subseteq u$	$v \subseteq u$	$v \subseteq u$	$v \cap u = \varnothing$
S_{OR}	$v \ni u$	$v \cap u \neq \varnothing$	$v \supseteq u$	$v \supseteq u$	$v \backslash u \neq \varnothing$
S_{XOR}	$v \ni u$	$\perp$	$\perp$	$v \supseteq u$	$\lvert v \backslash u \rvert = 1$
S_{NAND}	$v \not\ni u$	$v \cap u = \varnothing$	$v \cap u = \varnothing$	$v \cap u = \varnothing$	$v \subseteq u$

Table 7.3 Permissive matching between pattern and instance feature values organised by value type

The permissive FS matching has been developed together with the strict FS matching as it was not clear in the beginning which one is suitable for the current work. After conducting several tests it became clear that strict matching is the one yielding the better results because it yields fewer matches. This is especially visible in the cases of Transitivity enrichment when too many incorrect patterns are qualified by

the permissive operator. Now that pattern graph matching is explained, lets take a look next at how it is used to perform operation on instance graphs.

7.5 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. 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 *operational pattern graphs* (Definition 7.5.1). Next I explain how to embed the operations into the graph pattern and how they are used in the algorithm.

Definition 7.5.1 (Operational graph pattern)**.** An *operative graph pattern* is a pattern graph that has least on one node *operation* and *arg* features.

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 an 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. If a node has the feature operation then it is called *operational node*. Also, in the current implementation, the special features such as operation, arg, id, precede etc. are excluded from the pattern matching operation because they have functional role linked to the implementation and do not describe the linguistic properties of a graph node.

So far the implemented operations are *insert*, which is used for creation of empty nodes, *delete*, which is used for corrections of predictable errors in dependency graphs and *update* operation, which is the main mechanism behind graph enrichment. These operations are depicted in Figure 1.8 of the parser pipeline architecture from Section 1.7.4.

Operative patterns are enacted once they are matched to an instance graph. An operative pattern graph G is *enacted* on an instance graph H, in two steps. First, the pattern graph is strictly matched to instance graph. If an isomorphism f is found then, second, for every operational node $v \in G, \exists att(v) = operation$, the specified operation $op = val(v.operation)$ and the corresponding node of the instance graph $u \in H$, the

operation is executed on the node of the instance graph $op(u)$. If the arg feature is provided then the operation is executed with that additional argument.

7.5.1 Pattern based node update

As mentioned above the pattern based node update is used for adding onto the constituency graph new features. Lets consider Example 101 whose constituency graph is provided in Figure 7.15 and the task to assign *agent* feature to the subject node and *affected-possessed* feature to the complement. This can be done using the pattern graph matching with a feature update operation indicated on subject and complement nodes. The PG depicted in Figure 7.16 fulfils this purpose because it matches constituency graph from Figure 7.15 and has the corresponding update operations indicated.

(101) He gave the cake away.

(102) He gave her the cake.

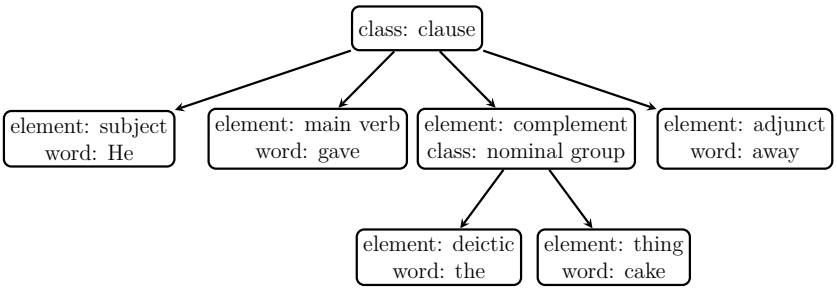


Fig. 7.15 Constituency graph corresponding to Example 101

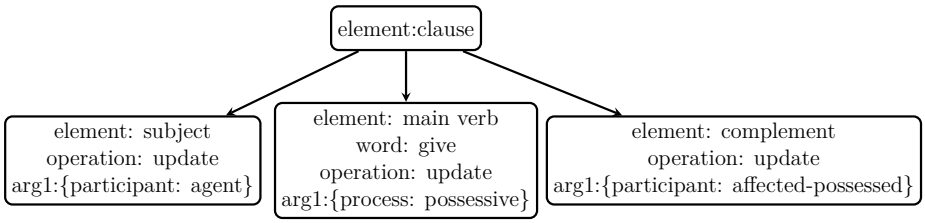


Fig. 7.16 A graph pattern for inserting agent and affected-possessed participant roles

Consider the same pattern, but applied to a sentence in the Table 7.4. The clause has two complements and they are by no means distinguished in the pattern 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

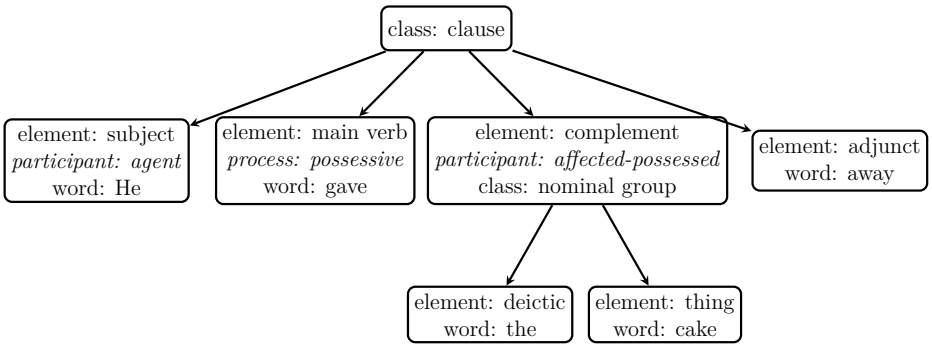


Fig. 7.17 The resulting constituency graph enriched with participant roles

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 7.18 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 7.4 CG with a di-transitive verb

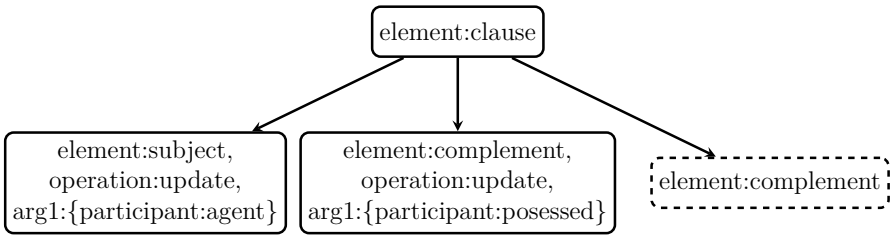


Fig. 7.18 PG for inserting agent and possessed participant roles to subject and complement nodes only if there is no second complement.

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.

7.5.2 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 6.

(103) Albert asked [∅ to go alone].

Consider the Example 103. 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 6.2. The table 7.5 provides a constituency analysis for the example and the null elements (in italic) are appended for the explicit grammatical account. In the Section 6.3 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 7.5 The constituency analysis that takes null elements into consideration

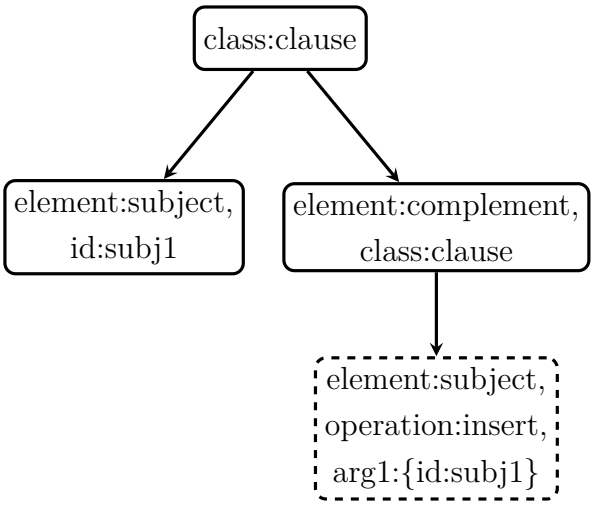


Fig. 7.19 A graph pattern to insert a reference node

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.

7.6 Systems and Systemic Networks

In the Section 3.2.4 I presented the basic definition of *system* and *system network* and the notations as formulated in the SF theory of grammar. In this work the system networks are simplified to hierarchies of disjoint features maintaining the entry conditions. This corresponds to the type logic component described in (O'Donnell 1993) where the syntagmatic organisation is restricted to a single functional layer. The reason behind this simplification is because the hierarchy of disjoint feature structures are perfectly suitable to correctly derive the complete set of parent features from a one

or several manual selections. Moreover the systemic networks are not interconnected into an uniform grammar but separate modules describing selected aspects of language. Once the complete set of features is derived then it is associated with a graph pattern. This pattern graph is then used in the parse graph enrichment process described in Section 1.7.4. Next I define the necessary concepts serving the current work.

First I would like to introduce abstract concept of *hierarchy* defined in a computer science fashion by Pollard & Sag (1987: 30) which is a formal rephrasing of Definition 3.2.1 stating that the hierarchy is a sort of logical precedence among the terms of a system.

Definition 7.6.1 (Hierarchy)**.** A hierarchy is finite bounded complete partial order $(\Delta, \prec)$.

The next concept that requires closer formalization is that of a system first established in Definition 3.2.10. For precision purposes, this has a narrower scope without considering the system networks or precondition constraints; these are introduced shortly afterwards building upon current one.

Definition 7.6.2 (System)**.** A *system* $\Sigma = (l, P, C)$ labelled l is defined by a finite disjoint set of distinct and mutually defining terms called a *choice set* C (of type S_{XOR}) and an *entry condition set* P (of type S_{OR}, S_{XOR} or S_{AND}) establishing the delicacy relations within a system network.

There is a set of functions defined that apply to systems: $label(\Sigma) = l$ 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 set of entry condition features, and the $size(\Sigma)$ return the number of elements in the system choice set.

Definition 7.6.3 (Systemic delicacy)**.** We say that a system S_1 is more delicate than S_2 denoted as $S_1 \prec S_2$ if

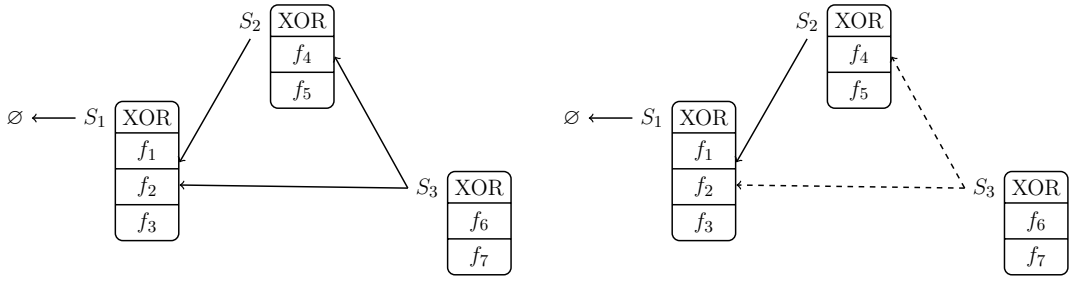
1. both system belong to the same system network: $S_1, S_2 \in SN$
2. there is at least a feature but not all of S_1 which belong to the entry condition of S_2 i.e. $choices(S_1) \in precondition(S_2)$ or
3. there is another system S_3 that has a among the entry conditions a feature of S_1 and whose features are among the entry conditions of S_2 i.e. $\exists S_1' \in SN$ such that $choices(S_1) \in precondition(S_3) \wedge choices(S_1') \in precondition(S_2)$

Systems are never used in isolation. SF grammars often are vast networks of interconnected systems defined as follows.

Definition 7.6.4 (System Network)**.** A *system network* $SN = (r, \Gamma)$ is defined as a hierarchy over a set of systems Γ where the order is that of systemic delicacy starting from a root feature r such that for any system in the network either there is another less delicate system or its entry condition is the root feature i.e. $\forall S_i \in \Gamma, \exists S_j \in \Gamma | S_j \prec S_i \vee precondition(S_i) = r$.

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 an *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.

The graphical notation of system networks has been introduced in Section 3.2.4. Considering the above definitions, the system network can be represented as a graph where each node is a system features and edges represent precondition dependencies. Figure 7.20 depicts examples of a system network with three systems. In Figure 7.20a the entry conditions are S_AND sets only and in Figure 7.20b the entry condition for S_3 is $OR(f_2, f_4)$ depicted with dashed lines. In such a graph, all system features must be unique i.e. $\forall S_1, S_2 \in SN : choices(S_1) \cap choices(S_2) = \varnothing$ and there must be no dependency loops.



(a) System Network with conjunctive entry conditions

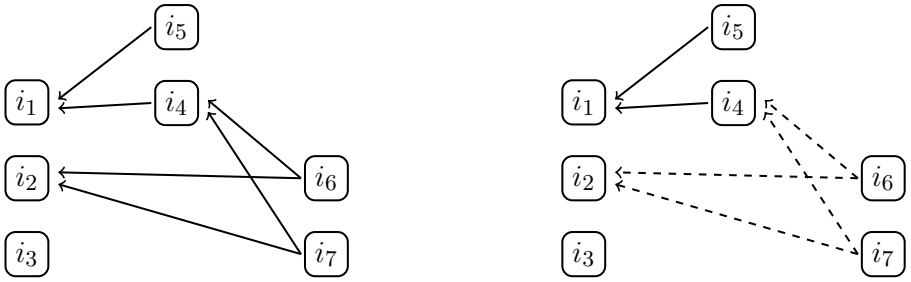
(b) System Network with disjunctive entry condition for S_3

Fig. 7.20 Example System Network presented as graphs

For a chosen feature in the system network it is possible to trace a path to the root feature by traversing systems through their preconditions. Generating such a path is equivalent to preselecting the features in all earlier systems. This is needed for assigning a complete set of feature selections to a pattern graph before it is used in the parse graph enrichment. Conversely, in the verification process it is necessary to check

whether a set of arbitrary features belong to a *consistent* and *complete selection path*. Next are addressed the concepts needed for addressing this task and how it is executed.

The system networks from Figure 7.20 can be unpacked into a *feature network* (Definition 7.6.5 which is referred to as *maximal selection graph*) interconnected by system entry conditions. In Figure 7.21 are depicted two feature networks corresponding to the system networks above. The dashed lines mean disjunction and continuous ones mean conjunction of entry features.



(a) Feature Network graph with conjunctive entry conditions only

(b) Feature Network graph with some disjunctive entry condition

Fig. 7.21 Example Feature Network graphs

The feature network can be defined in relation to the system network as follows.

Definition 7.6.5 (Feature Network)**.** For a given system network $SN = (r, \Gamma)$, a *Feature Network* $FN(N, E)$ is a directed graph whose nodes N are the union of choice sets of the systems Γ and the edges E connect choice features with the entry condition (precondition) features of the systems Γ.

Given a feature network $FN(N, E)$ and a feature the network $f \in N$ a *selection path* $SP(N, E)$ is a connected sub-graph of traversal paths between the root of the feature network and the feature f. A *complete selection path* is a selection path from one of the leaf nodes up to the network root.

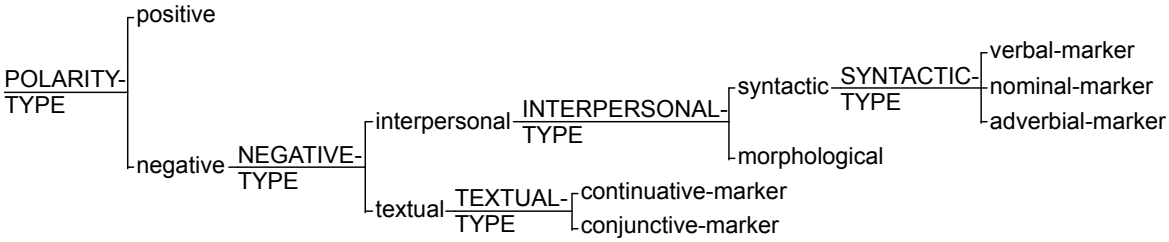


Fig. 7.22 Polarity System

The selection path is generated by traversal of the feature network from a given feature node towards the rood node. If the node has no incoming edge then the result

traversal it is a leaf node and the resulting selection path is a complete one. For example if *verbal-marker* feature is selected in the system network depicted in Figure 7.22 the traversal of the corresponding feature network towards root feature yields the selection path *negative – interpersonal – syntactic – verbal-marker*. This path is also complete with respect to the network because there are no systems younger than the SYNTACTIC-TYPE system.

When the system networks define more than one feature in the precondition it leads to split paths it means that there is more than one feature needs to or can be preselected. If the edge is a continuous line (depicted in Figure 7.21a) the both variants are part of the same selection path. In case the edge is a dashed line (depicted in Figure 7.21b) the paths are considered alternative and further decision making mechanism shall be employed to reduce the disjunction to a single variant. In the current work no such mechanism is employed and the parse result is presented with both alternatives. In most cases the entry condition is constituted of a single feature, when there are multiple ones, usually they are conjuncted and only in a small minority of cases entry condition is a disjunction of features.

Algorithm 1: Naive backwards induction of a selection path

```
   input  : feature, system network
1  begin
2      add feature to empty selection path
3      for system in traversal path to the root of system network:
4          get entry condition features of system
5          add entry condition features to the selection path
6      return selection path
7  end
```

The pseudo code for creating a selection path as described above is outlined in Algorithm 1. Given a system network a random non root feature belonging to the network, it traverses the networks, one by one, towards the root and collects the entry conditions of each system into a selection path.

7.7 On realisation rules

The previous section explains that in the current work systemic networks are simplified to a taxonomy of features and no realisation rules are considered. This section explains how the pattern graphs are a substitute to the realisation rules and how they relate to each other.

The *realisation rule* of a systemic feature specifies how that feature is realised as a syntagmatic structure down to a text form using operations such as insert or expand constituent, order two constituent, preselect another feature, lexicalise etc. They are the essential ingredient binding the paradigmatic description into a syntagmatic structure. Robin Fawcett recurrently emphasises the role of realization rules in the composition of system networks. He often stresses "no system networks without realization rules". 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. Not to mention that writing a good grammar is already a very difficult task. 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 which are not really available yet.

In this work the graph patterns can be considered as bundles of realisation rules and feature selections and therefore an approach to replace the realisation rules. This idea is implicitly covered in Section 7.3 and here I explicitly describe it through an example. Consider the fragment of the Mood system network from Halliday & Matthiessen (2013: 162) depicted in Figure 7.23. This example aims at three feature selections: major, indicative and declarative. The root feature in the system network is realised through a constituent *clause*, any of the selected fratures is ascribed directly to it, and the selection of any subsequent features impacts the elements below through the associated realisation rules. The pattern graph that captures selection of major feature is depicted in Figure 7.24.

In Figure 7.25 is depicted the pattern graph corresponding to the selection of indicative feature. Here the clause constituent receives the whole selection path up to the root feature *clause – major – free – indicative* and in addition two more constituents are required: Subject and Finite. Almost same pattern graph is valid in case of selecting bound feature instead of indicative.

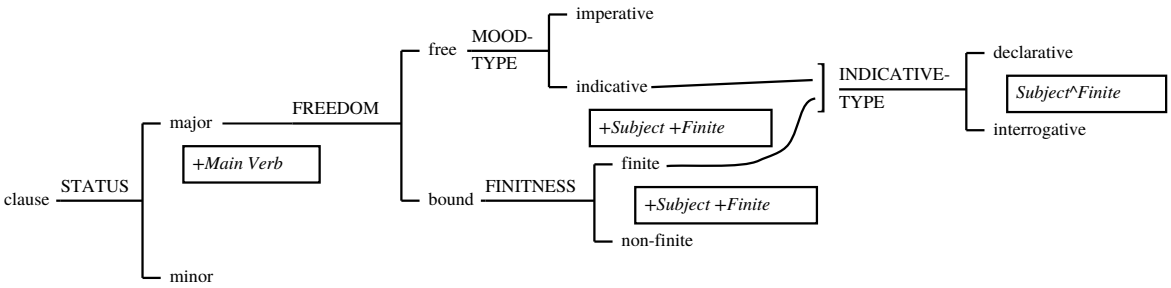


Fig. 7.23 An adapted fragment of a Mood system from (Halliday & Matthiessen 2013: 162)

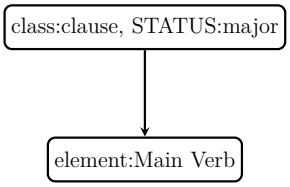


Fig. 7.24 A graph pattern for *major* feature selection in Figure 7.23

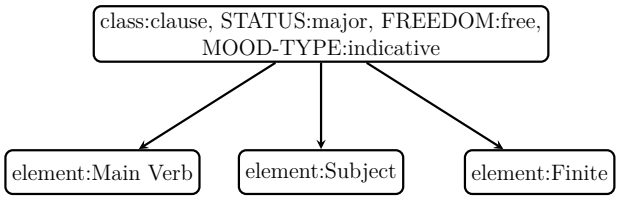


Fig. 7.25 A graph pattern for *indicative* feature selection in Figure 7.23

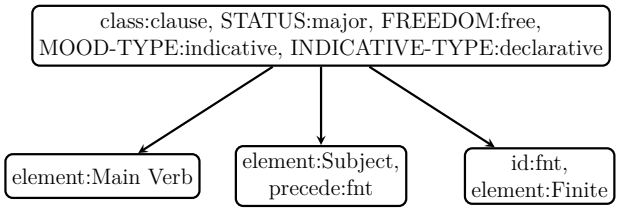


Fig. 7.26 A graph pattern for *declarative* feature selection in Figure 7.23

In Figure 7.26 the selection is taken one step further to the declarative feature. The associated realisation rule is ordering the Subject and Finite elements. This is captured via the "precede:fnt" where "fnt" is the id of the Finite constituent.

The pattern does not need to refer to the complete selection path but can be limited to the context of a few related features. For example to represent selection of indicative and declarative features in isolation is depicted in Figure 7.27. In this case the class of the parent constituent (that in the above cases is the clause) is no longer specified because the restricted selection path and thus the root of the network is not reached. Also none of the preselected features major and free are specified either.

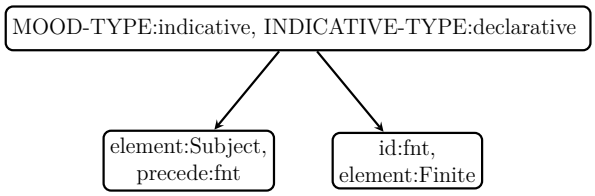


Fig. 7.27 A graph pattern for the selection if *indicative* and *declarative* features in Figure 7.23

One limitation, however, of the graph patterns is dealing with conflation phenomena. For example, the cases of the Main Verb conflated with Finite elements have to be accounted by with pattern graphs where one constituent has two functions instead of pattern graphs with two separate constituent each with the corresponding function. This limitation can be addressed in the future by manipulating the definition of the feature structure matcher described in Section 7.4.

7.8 Discussion

This chapter described from a computer science perspective the SFL concepts introduced in Chapter 3 and prepares the ground for the following chapters that address the implementation details of the Parsimonious Vole parser.

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 an excellent candidate for expressing *systemic realizations* which have not been considered in the current work as being associated to the systemic features.

In Section 7.7 I mentioned that authoring realisation rules is a difficult task and requires a proper tool support. The same is the case for graph patterns and even more so when they need to be related to systemic networks or network parts. The system network authoring tool, such as the one available in UAM Corpus Tool (O'Donnell 2008b), should provide also a graph pattern editor allowing association of graph patterns to systemic features. Unfortunately building such an editor is out of the scope of the current work and is among the priorities in the future developments. Also employing a mature specialised technology for manipulating large amounts of graph data available in the Semantic Web suite of tools is another direction for the future described in the Section 11.2.

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 8

Creating the systemic functional constituency structure

The previous chapter introduced the building blocs for the parser pipeline algorithm depicted in Figure 1.8. This chapter covers the entire first phase of the of the algorithm called "graph building".

The considered input into the parsing pipeline are Stanford dependency parse graphs. The dependency graphs are sometimes erroneous or treat certain linguistic phenomena in an incompatible way with current approach. Therefore a preprocessing stage is needed to correct and canonicalise the dependency graphs which is covered in Section 8.1 and 8.2. Then these graphs are rewritten into systemic constituency graphs. The process by which this happens is covered in Section 8.3.

8.1 Canonicalisation of dependency graphs

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

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

transformation types: (a) *relabelling of edge relations*, (b) relabelling node POS, and (c) reattachment of nodes to a different parent.

8.1.1 Loosening conjunction edges

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

Some common patterns occurring between noun, verb and adjective conjuncts are depicted below in figures 8.1 - 8.5.

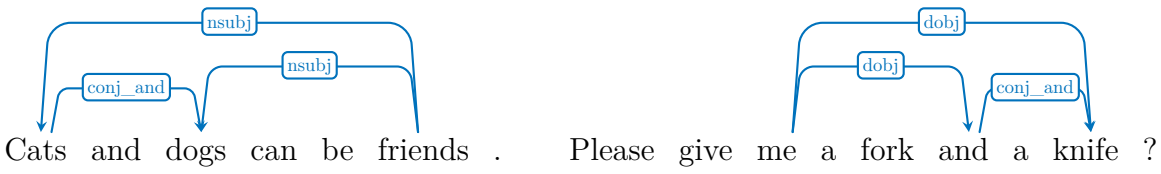


Fig. 8.2 Conjunction of noun objects

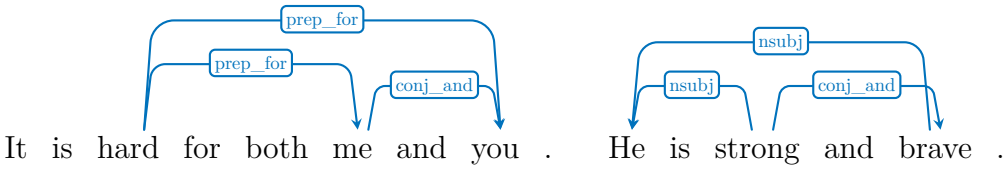


Fig. 8.4 Conjunction of copulatives sharing the subject

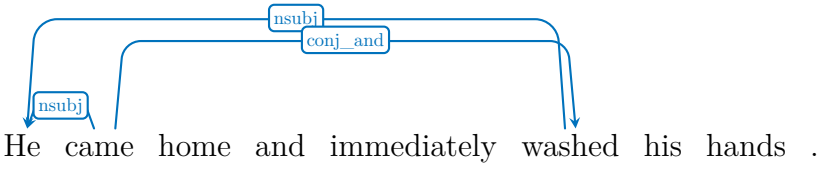


Fig. 8.5 Conjunction of verbs sharing the same subject

The main reason these extra edges need to be removed is to avoid traversal of the same node via different paths. This, in the current algorithm, has as consequence execution multiple times of the same operation such as for example creation of multiple constituents from the same DG node, which of course is undesirable. For example,

if multiple subject relations occur in the DG then multiple subject are going to be instantiated in CG which is not covered in the grammar. Rather only one complex unit needs to be created with the subject role composed of two noun phrases (see discussion of this case in the Section 3.4.6).

The way I fix this problem is by removing functional edges to/from each conjunct except the first one. There are two generic patterns in figures 8.6 and 8.8 correspondingly with incoming and outgoing edges that are transformed into the forms depicted in 8.7 and 8.9.

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 8.6 and exemplified in Figures 8.1 - 8.3. In SFG terms it corresponds to cases when the functional element of a parent constituent is filled by a complex unit below.

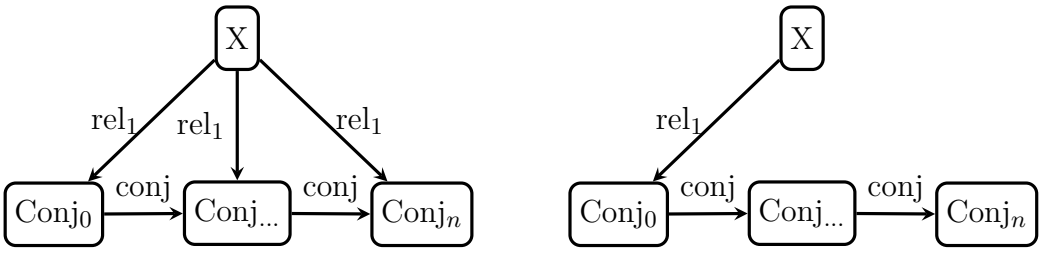


Fig. 8.7 Conjuncted elements with incoming loosely connected dependencies

The second is the pattern of conjuncts with *outgoing dependency relations* depicted in Figure 8.8. 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 6 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 8.4 - 8.5.

8.1.2 Transforming copulas into verb centred clauses

In Stanford dependency grammar *copular verbs* are treated as dependants of their complements (see Figures 8.10 and 8.11) 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 4.1.1 and 4.1.2.

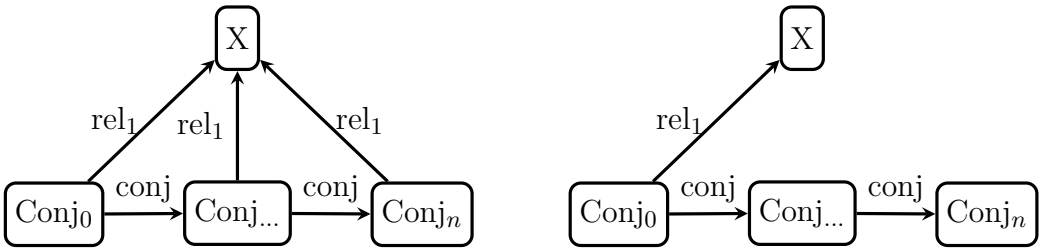


Fig. 8.9 Conjuncted elements with outgoing loosely connected dependencies

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. 8.11 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 8.1 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 8.1 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 8.1, 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 8.12 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 8.13 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.

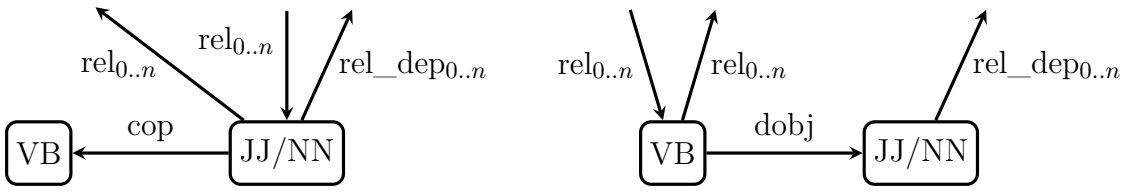


Fig. 8.13 Generic pattern for copulas after the transformation (the same as non-copular verbs).

In case of conjuncted copulas like in the example in Figure 8.11 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.

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

The Figure 8.14 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 8.14 receives a different analysis as it is represented in Table 8.2.

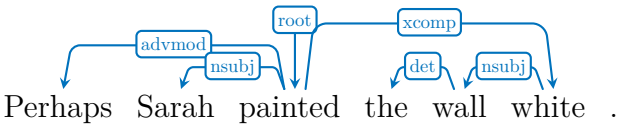


Fig. 8.14 Dependency parse for clausal complement with adjectival predicate

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

Table 8.2 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 8.14 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 8.15 to form in Figure 8.16. 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. 8.16 Adjectival clausal complement as secondary direct object

8.2 Correction of errors in dependency graphs

The Stanford Parser applies various machine learning(ML) techniques to parsing. It's accuracy increased over time to $\approx 92\%$ for unlabelled attachments and $\approx 89\%$ for labelled ones (in the version 3.5.1). This section addresses known error classes of wrongly attached nodes or wrongly labelled edges and nodes. These errors have been

discovered during the development of Parsimonious Vole parser and are corrected to increase the result accuracy.

As the Stanford parser evolved, some error classes changed from one version to another (v2.0.3 – v3.2.0 – 3.5.1). Also the set of dependency labels for English initially described in (Marneffe & Manning 2008a,b) changed to a cross-linguistic one (starting from v3.3.0) described in (Marneffe et al. 2014).

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

8.2.1 *prep* relations from verb to free preposition

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

This is not always the case especially with phrasal verbs, the *prt* relations are mislabelled as *prep*. The correction consist in changing the *prep* (figure 8.17) into *prt* (figure 8.18) if the preposition node has no children e.g. *pobj*.

Fig. 8.18 Corrected relation to free preposition as verbal particle

8.2.2 Non-finite clausal complements with internal subjects

The *xcomp* relation stands for open clausal complements of either a verb(VB) or adjective (JJ/ADJP). The latter is actually transformed as discussed in Section 8.1.3. The open clausal complement defined in Lexical Functional Grammar (Bresnan 2001: p270–275) is always non-finite and does not have its own subject. However sometimes *xcomp* relation appears either (a) with finite verbs or (b) with own local subjects and both cases correspond to definition of *ccomp* relation.

To fix this I transform all the instances of *xcomp* relation to *ccomp* if the dependent verb has a local subject (nsubj) or a finite verb as depicted in Figures 8.19 - 8.20.

Fig. 8.20 Corrected clausal complement

8.2.3 The first auxiliary with non-finite POS

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

The algorithm is thus checking that the POS of the first auxiliary is according to the mapping defined in the Table 8.3.

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 8.3 Mapping lexical forms of auxiliaries to their POS

8.2.4 Prepositional phrases as false prepositional clauses

prepc is a relation that introduces, via a preposition, a clausal modifier for a verb, adjective or noun. Assuming that the copulas had been changed as described in subsection 8.1.2 then the head and the tail of the relation can only be a verb. However when the relation head is not a verb (only nouns encountered so far) then the relation needs to be corrected from *prepc* to *prep* introducing a prepositional phrase rather than a subordinate clause.

Fig. 8.22 Corrected prepositional phrase

8.2.5 Mislabelled infinitives

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

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

Fig. 8.24 Correct infinitive

Fig. 8.26 Correct present simple

8.2.6 Attributive verbs mislabelled as adjectives

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

Fig. 8.28 Corrected attributive verb

8.2.7 Non-finite verbal modifiers with clausal complements

The early version of Stanford Dependencies (Marneffe & Manning 2008a) proposes two relations for non-finite vernal modifiers *partmod* for participial and *infmod* for infinitival forms exemplified in 104.Latter in (Marneffe et al. 2014) both relations have been merged into the *vmod*.

(104) 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 104) 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 104 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 8.29 which is transformed by the algorithm into the form represented in Figure 8.30

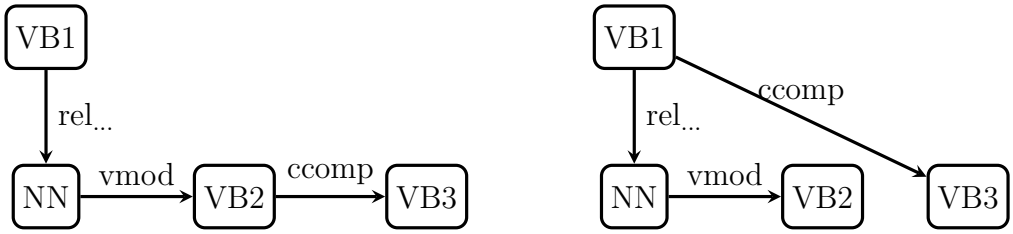


Fig. 8.30 Clausal complement attached to the main clause

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

8.2.8 Demonstratives with a qualifier

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

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

(105) Bill moved those beyond the counter.

(106) Put that in our plan.

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

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

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

(110) 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 107 and 108 and cases such as 109 and 110 when the post-modifier pertains to teh clause.

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

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

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

In examples 105 and 106, attaching the prepositional phrase to the demonstratives (depicted in Figure 8.31) is incorrect. It should be attached to the verb (like in the figure 8.32) 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.

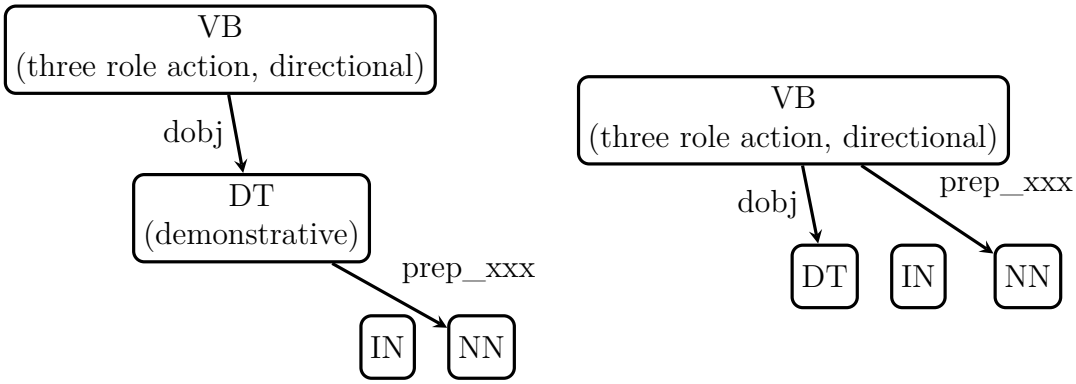


Fig. 8.32 Prepositional Phrase attached to the verb with a demonstrative pronoun in between

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

8.2.9 Topicalized complements labelled as second subjects

In generative grammars the topicalization (or thematic fronting) of complements is described in *Trace Theory* as *WH/NP/PP-movement*. Examples 111–116 (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 115–116.

(111) Joe(,) his name is.

(112) Relaxation(,) you call it.

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

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

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

(116) 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.

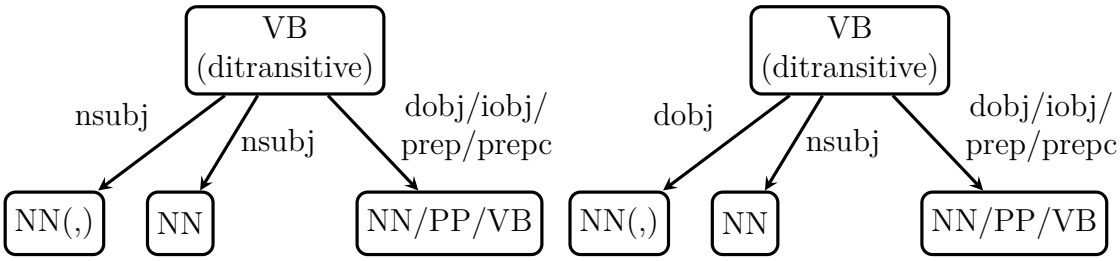


Fig. 8.34 Topicalized Direct Object – moved to pre-subject position

The algorithm is looking for the cases of multiple subjects (represented in figure 8.33) and gives priority to the one that is closest to the verb. The other one is relabelled as a complement (Figure 8.34). The rule is generalized in the algorithm for multiple subjects even if so far only cases of two subjects have been observed.

8.2.10 Misinterpreted clausal complement of the auxiliary verb in interrogative clauses

Sometimes the auxiliary verb in the interrogative clauses (examples 117 and 118) 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.

(117) Do you walk alone?.

(118) Has Jane fed the cat?.

The algorithm searched for the pattern depicted in Figure 8.35 and transforms it into the form Figure 8.36.

Fig. 8.36 Corrected clausal complement

8.3 Creation of systemic constituency graph from dependency graph

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

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

The currently implemented process consists of DG traversal and execution of generative operations on a parallel structure, progressively building the CG. The choice of the generative operation is based on a rule table described in Section 8.3.3 The DGs and CGs resemble each other but they are not isomorphic. The CG is created in two phases presented in the Algorithm 2. The first phase, through a top-down breadth-first traversal of a DG, generates, using a rule-set, an incomplete CG that misses the heads and few other elements for each CG unit. Also no unit classes are specified, with except if that unit is a clause. The second phase, through a bottom-up DG traversal, complements the first one ensuring creation of all CG constituents, corresponding to missing unit elements and assigns the unit class.

Algorithm 2: Creation of the constituency graph

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

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

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

8.3.1 Dependency nature and implication on head creation

In Section 5.6 is explained the different dependency relation nature in dependency graphs and in systemic functional constituency graphs. The DG uses a *parent-daughter dependency* while in Constituency Graphs there is a *sibling dependency*. This difference implies that, when mapped into CG, a DG node, stands for both a unit and that unit's head. In other words a DG node corresponds to two functions and unit classes at different rank scales. For example the root verb in DG corresponds to the clause node and the lexical item which fills the Main Verb of the clause.

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

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

8.3.2 Tight coupling of dependency and constituency graphs

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

Through a stack of CG nodes, the DG nodes are made aware of which CG nodes and in which order they subsume them (Figure 8.37). 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 8.38). This way the positioning information is available bidirectionally about CG an DG structures.

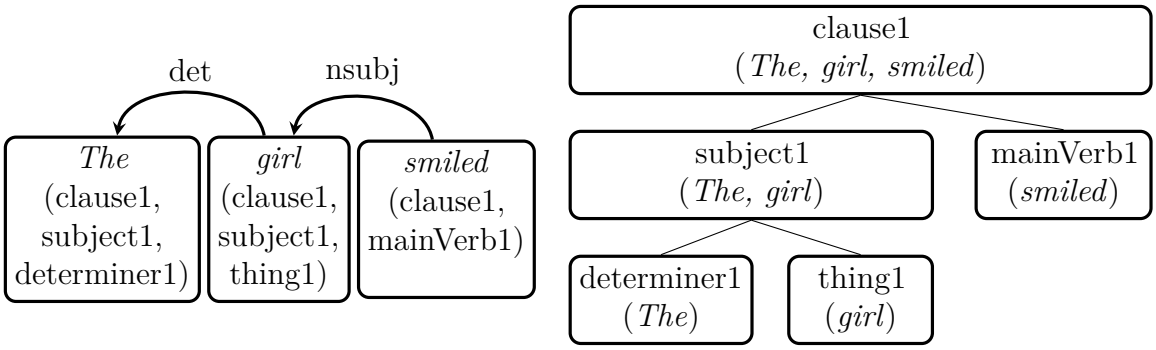


Fig. 8.38 Dependency aware CG nodes

The Figure 8.37 depicts a dependency graph. Each node has a stack of ids corresponding to constituents in the CG (Figure 8.38). Conversely, in Figure 8.38, depicts a constituency graph where the constituent nodes carry a set of tokens corresponding to DG nodes. This way the DG in Figure 8.37 and the CG in 8.38 are aware of each other.

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

8.3.3 Rule tables

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

A simplified example of the rule table is presented in Table 8.4. The full table is provided in the Appendix .2.5. It can be regarded as an attribute value matrix

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

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

Table 8.4 Example of rule table mapping specific and generic dependency context to generative operations

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

The parameter is used only for the operations (a) and (b) and specifies which element the new constituent is filling as described in Section 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, when the verb semantic configurations are verified, these options are reduced to one.

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

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

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

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

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

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

Algorithm 3: Operation selection in the rule table based on the edge type

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

The Algorithm 3 is based on two dictionary lookups: one for specific key (contextualised by the edge relation and its nodes) and another one for generic key based on the edge relation alone. The rule table is conceived as a Python dictionary with string keys and and two-tuple containing the operation and the element type parameter. If the key is found (either specific or generic) in the rule table then the operation and parameter are returned otherwise None is returned.

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

8.3.4 Top down traversal phase

The goal of this first phase is to bootstrap a partial constituency graph starting from a given dependency graph and a rule table. The CG is created as a parallel graph structure through the process of breadth-first traversal on DG edges as described in Algorithm 4. The rewriting of the DG graph into a partial CG is performed as follows. DG nodes, starting from the root, are traversed top-down breadth first order and for each visited node apply the creation or extension operation as provided in the rule table.

Algorithm 4: Partial constituency graph creation by top down traversal

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

First the CG is instantiated and an empty root node is created within. Also, the root node is made aware of the root node in DG as described in Section 8.3.2. Then

the DG is traversed in breadth first order (BFS) starting from the root node and as each DG edge is visited an operation is chosen based on the edge type and POS of the connected nodes (Lines 7 - 10). The Line 5 of the algorithm is responsible for looking up and selecting the operation from the rule table as described in Algorithm 3. Then the creative operation is executed with the established parameters: dependency successors (children), a constituent node parenting the newly created one (cg pointer), element type which is chosen from the rule-table together with the operation.

In Python function are first class objects allowing objects to be called (executed) if they are of callable typed. This duality allows storing the functions directly in the rule table and then, upon lookup they are returned as objects but because they are also callable these objects are executed with the expected set of parameters based on their function aspect. The possible operation have been already explained in Section 8.3.3: *extend current* and *new (sibling) constituent*. Next I present the pseudo-code for each of the operations. Note that these operations do not return anything because their effect is on the input cg and dg.

Extend constituent. Algorithm 5 outlines the functionality for extending current cg pointer. It does two main things. It increases the span of an already existing CG node over more DG nodes and makes them, concomitantly, aware of each other.

Algorithm 5: Extend a constituent with DG nodes

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

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

This exceptional treatment is due to the fact that *prep* and *conj* relations are always specialised by the preposition or conjunction in place. For details on this aspect of Stanford Dependency Grammar please refer to Section 5.5.

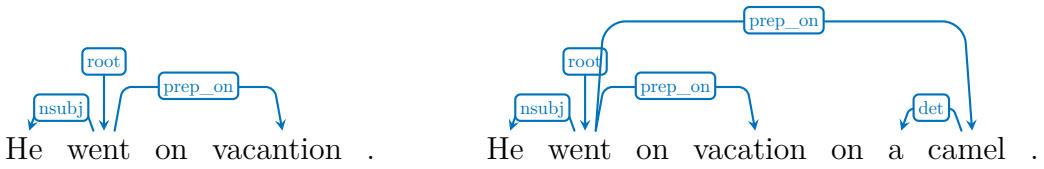


Fig. 8.39 Challenging free nodes

Figure 8.39 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 6 is a operation that inserts into CG a new node/constituent (line 4). The new constituent is created as a child of a pointed CG node with the element type extracted from the rule table together with the operation. Once the cg node is created, it is extended with the children nodes as described in Algorithm 5 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.

A variation of the *create new* is *create sibling* outlined in the Algorithm 7. 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 6: Creating new child constituent

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

Algorithm 7: 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 6
3  |  create new constituent to an updated cg pointer
4  end
```

8.3.5 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.

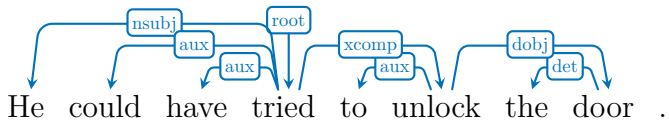


Fig. 8.40 The dependency graph before the first phase

The Figure 8.41 depicts and example CG generated in the 1^{st} phase with dotted lines representing places of the missing constituents.

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

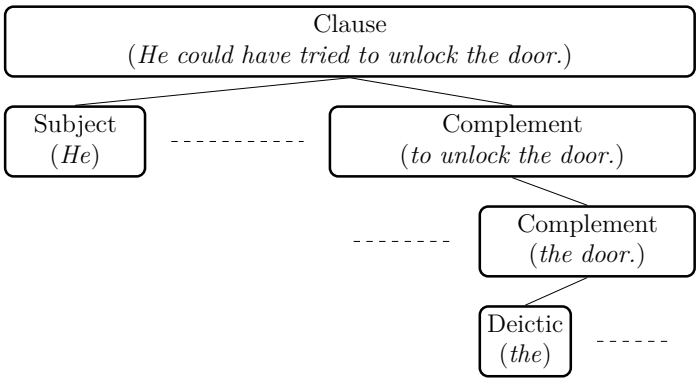


Fig. 8.41 Constituency graph after the top down traversal missing the head nodes

The class membership of constituent units is decided based on three informations available within each constituent: (a) part o speech of the head dependency node (b) element type of the constituent and (c) presence or absence of child constituents and their element types. The corresponding constraints are listed in Table 8.5. The first column carries the unit class (to be assigned), the second and third columns enumerate part of speech and element types that constituents might fill. The Second and third column enumerations are exclusive disjunctive sets (S_{XOR}) because only one may be selected at a time while the list in last column is an open disjunction (S_{OR}) because any of child elements may be present. The last column is an enumeration of what child constituents might the current one have. The *n/a* means that information is unavailable.

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

Table 8.5 Constraints for unit class assignment

The Algorithm 8 traverses the CG (not the DG) bottom-up with *post-order depth-first* order (line 2) in order to assign to every visited constituent node a unit class and create the missing child constituents.

Algorithm 8: Creating the head units and assigning classes

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

As mentioned above in Section 8.3.2, CG and DG are tight coupled which means that each CG node spans over a set of DG nodes. In this stage, traversal over the CG nodes is equal to traversing groups of DG nodes at each step. This creates focused mini context suitable for resolving the unit class and the missing elements out of the DG chunks.

When assigning unit class the following informations are considered: (a) part of speech of the head DG node that triggered node creation in the first phase (head POS), (b) the assigned element type (element type) and (c) element type of each direct child (children of node). Lines 6 to 7 assign classes according to conditions provided in Table 8.5.

The CG nodes corresponding to non-modal verb DG nodes are assigned clause class. This rule corresponds to the one main verb per clause principle discussed in Section 4.1.1. This approach however does not take into consideration elliptic clauses and they need additional resolution that is considered for the future works and can be overcome by an *ellipsis resolution mechanism*, similar to the one for *null elements* described in Chapter 6.

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

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

8.4 Discussion

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

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

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

Chapter 9

Enrichment of the constituency graph with systemic features

The previous chapter describes how to create the systemic functional constituency structure which is the syntagmatic organization aimed at in this work. This chapter presents the mechanisms by which the paradigmatic account is provided through selection of systemic features at the level of each constituent. I present how constituency graph is enriched with features from two main system networks MOOD, introduced in Section 4.2.1 and TRANSITIVITY, introduced in Section 4.2.2.

In the parsing process pipeline depicted in Figure 1.8 there is an phase called *increasingly semantic graph enrichment*. The present chapter covers this phase entirely starting from Mood enrichment, to Null Element creation and finally Transitivity enrichment.

The main method of enrichment is execution of update graph patterns (presented in Section 7.5) on the constituency graph. The MOOD graph patterns have been manually designed across various levels of delicacy. The patterns usually cover from one to four systemic selection from sibling or chained systems. Then the graph patterns are employed in the MOOD enrichment process provided in Section 9.2.

In SFL, Transitivity analysis roughly corresponds to what is known in mainstream computational linguistics as semantic analysis of text or *semantic role labelling* (SRL), a well established task in the mainstream computational linguistics (Carreras & Màrquez 2005; Pradhan et al. 2007). In this task the clause is assigned a semantic frame in which the predicate functions as the process and the participant constituents take frame dependent roles (or functions). The nodes that do not receive any participant role are adjuncts which act as circumstances and currently outside the scope of this thesis.

Sometimes not all constituents are realised in the clause. Usually these constituents play semantic roles in the process configuration realised by the clause. This happens for one of two reasons: the participant is implicit and resolvable from the discourse structure or it is implicit and resolvable by a lookup outside the clause borders within the same sentence. The resolution from the discourse structure is outside the scope of the current work, the second resolution type, however, is implemented as it is based only on the sentence structure. In this work the account for empty constituents is implemented as described in Government and Binding Theory (GBT) (Haegeman 1991) introduced in Chapter 6.

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) introduced in Section 4.2.3. 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 roles.

The majority of implementations for the SRL task use probabilistic models trained on an annotated corpus whose outcome is a single most probable assignment of semantic roles and the selection is based on the maximum likelihood. In the current work I employ a lexical data base containing sets of configurations for each verb sense. These configurations are interpreted as what may be the case, i.e. what are the possible semantic roles, and the result is a small set of possible semantic roles rather than the best single guess. In order to reduces the number of possible assignments taking the analysis close to the goal of a single "correct" configuration I use the preparatory step of identifying covert constituents (i.e. Null Elements).

In the following sections I will explain the practical steps of enriching the CG with TRANSITIVITY features starting from how the PTDB has been normalised and made machine friendly, then how the graph patterns have been generated from the PTDB and finally how the patterns are matched onto the structural backbone enriching it with systemic features.

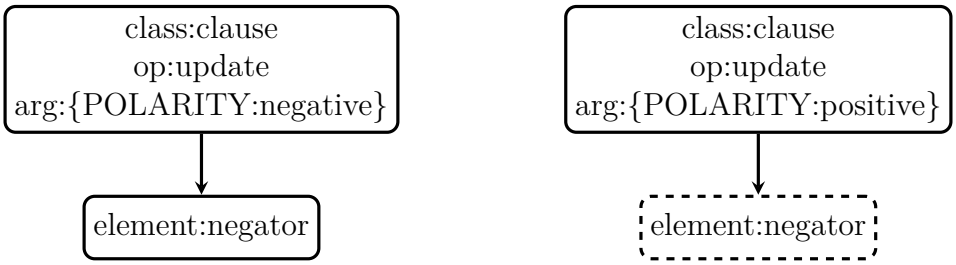
9.1 Creation the MOOD graph patterns

In this section I describe a set of graph patterns that have been manually created and included into the parsing process of MOOD system. All of the MOOD features can be recovered from the constituency and dependency elements. Therefore the graph

patterns provided below are relying on constituency information i.e. class and element and on dependency information such as POS, lemma, incoming and outgoing relation for the anchor node, order, etc.

This section describes how the patterns look like and how they were created. An extended representation of all the patterns is provided in the Annexe .2.5.

The first two patterns depicted in Figure 9.1 are used to determine POLARITY choices. The clause polarity in this work is considered to depend solely on the presence of a negation particle (this limitation is addressed in Section 4.2.1), which, in DG is represented by the *neg* edge label and in CG is signalled by the presence of a negator element. The pattern in Figure 9.1a specifies that the root node is a clause and has a negator constituent. If this pattern is identified then the negative POLARITY feature selection is added to the clause node. Conversely, if the pattern from Figure 9.1b, where the negator element is marked to be missing, is successfully matched then the clause node is updated with positive POLARITY.

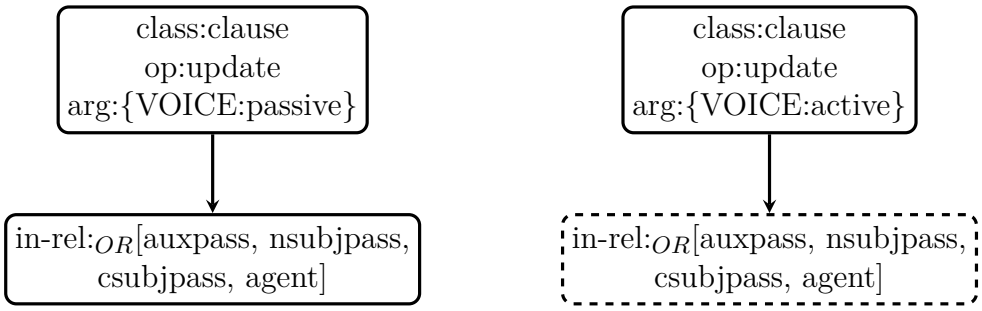


(a) Negative POLARITY (b) Positive POLARITY

Fig. 9.1 POLARITY detection graph patterns

A similar case holds for VOICE. Graph pattern in Figure 9.2 corresponds to the selection of passive voice. In DG it is captured by any of the four relations outgoing from a verb to another node. The dependency relations are: *auxpass*, *nsubjpass*, *csubjpass*, *agent* introducing either a passive auxiliary verb, a nominal subject, clausal subject or the agent in the complement position. If the pattern is matched in a dependency graph then it reflects passive voice otherwise, the voice is selected active. As the CG nodes have full awareness (described in Section 8.3.2) of DG nodes and edges the patterns can be written using the incoming relation (in-rel) special feature that represents all the incoming edges to the DG node corresponding to current constituent.

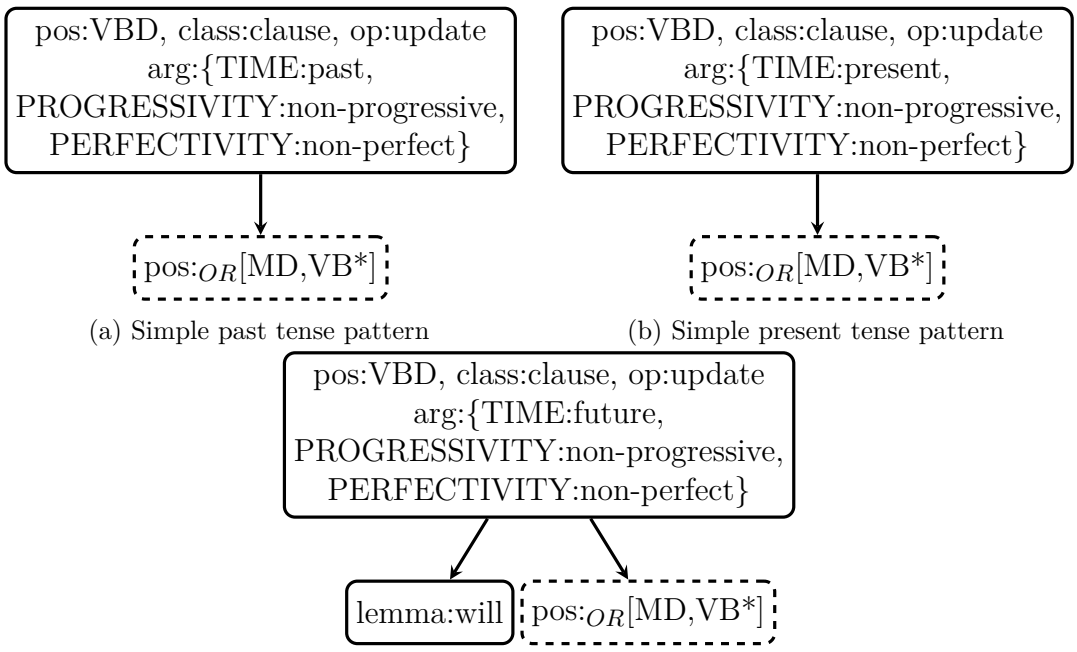
The story is very similar for the rest of the patterns. There is a mixture of positive and negated nodes (in dashed boxes) described by constituency or (still) dependency special features. The reason for using GD features is that they already cover a rich



(a) Passive voice (b) Active voice

Fig. 9.2 Voice detection graph patterns

syntactic variations that shall be exploited. As we move away towards patterns with higher in semantic information the DG features will no longer be used.



(a) Simple past tense pattern (b) Simple present tense pattern

(c) Simple future tense pattern

Fig. 9.3 Simple past, present and future tense patterns

In Figure 9.3 is depicted a set of patterns for enriching the simple present, past and future tenses. They are presented as an additional example of graph patterns. They are mutually exclusive by design, but in principle no one can prevent a mistake to happen and more than one graph pattern to yield a positive outcome. This is, in fact, one of the limitations of the current parsing method. So for example if the negated described by pos:$_{OR}$[MD,VB*] is omitted from the pattern, then it will yield matched

with situations when there is an auxiliary modal verb involved which is not correct because the intention is to capture clauses with a single verb.

9.2 Enrichment with MOOD features

In this stage the CG nodes are assigned system network features from graph patterns and a lexical-semantic dictionary. This is achieved by visiting each CG node in a bottom-up order and executing the enrichment functions.

The enrichment functions are provided via two rule tables. The first rule table (activation table) offers a pairing between element class and/or element (also called here the trigger) and an ordered set of system names. The second rule table is an association between a system name, an enrichment function (either matching or dictionary lookup) and a parameter that is either a set of graph patterns or a dictionary. Both tables have been manually created following the MOOD system network but also a few simple systems for nominal phrases.The graph patterns and dictionaries have been also manually complied following either traditional grammar sources, mainly Quirk et al. (1985), or SFL sources, mainly IFG4 (Halliday & Matthiessen 2013).

The systems are provided in Table 9.1,in the right column, as ordered associations to the node class or function, in the left column. The list of systems from the MOOD network is partial because, as we will see next, the set of associated patterns, usually cover several systems at once, therefore the least delicate one serves as a marker for a portion of the system network.

Key	*System activation order*
clause	POLARITY, FINITNESS, MOOD TYPE, VOICE, TENSE, MODALITY TYPE
nominal	PERSON, ANIMACY, GENDER, NUMBER
deictic, pre-deictic	DETERMINATION
thing, possessor	PERSON, ANIMACY, GENDER, NUMBER

Table 9.1 System activation table by unit class or element type

Table 9.2 associates systems with the enrichment functions and a parameter. Currently there are two enrichment functions: one base on matching graph patterns and the other one based on a dictionary of lexical items paired to a set of features. Most of these associations have been first implemented as hard-coded Python functions and latter transformed into the graph patterns with update operations (see Section 7.5 describing graph pattern based operations).

System Name	*Function*	*Parameter*
POLARITY	match all patterns	polarity set
VOICE	match all patterns	voice set
FINITNESS	match all patterns	finitness set
MOOD TYPE	match all patterns	mood set
TENSE	match all patterns	tense set
MODALITY TYPE	match all patterns	modality set
DETERMINATION	dictionary lookup	determination dictionary
PERSON	dictionary lookup	person dictionary
ANIMACY	dictionary lookup	animacy dictionary
GENDER	dictionary lookup	gender dictionary
NUMBER	match all patterns	plurality set

Table 9.2 Association of systemic networks to functions

Algorithm 9: Enriching CG with systemic features

```
   input  : cg
1  begin
2      for node in list of cg nodes in DFS postorder:
3          for network in activated networks for the node class or element:
4              selector function ← get associated function and parameter
5              run selector function knowing node, cg and parameter
6  end
```

Algorithm 9 presents the pseudocode for enriching CG nodes with systemic choices using enrichment functions based on the two rule tables presented above. There is one loop nested into another. The outer one is a bottom-up iteration over the CG nodes in depth first (DFS) post-order. The inner loop iterates over the provided systems for the focus node. Then a lookup in the Table 9.2 returns the enrichment function to be executed for that system network. This technique of using a mapping table from system networks to functions is similar to the one employed in CG creation phase (Algorithm 4) presented in previous chapter.

Algorithm 10: Match-all-patterns enrichment function

```
   input  : node, cg, pattern set
1  begin
2      for pattern in pattern set:
3          execute pattern based node update in cg given node context
4  end
```

Algorithm 10 takes a focus node, the cg it belongs to and a pattern set and executes for each pattern the pattern based node update operation as described in Section 7.5. If multiple patterns match then multiple updates are applied one after the other. The second enrichment function is based on associations between lexical items and feature selections described below.

Algorithm 11: Dictionary-lookup enrichment function

```
   input  : node, cg, dictionary
1  begin
2      lexical item ← get lexical item from the node
3      find lexical item in dictionary
4      systemic choices ← get the associated features + implied preselected features
5      add systemic choices to the node
6  end
```

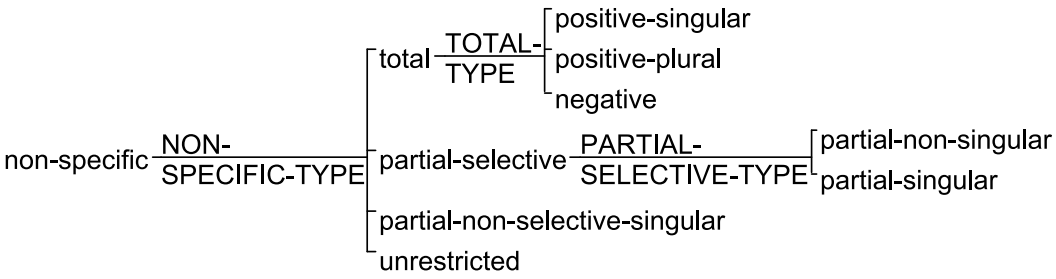


Fig. 9.4 The NON-SPECIFIC DETERMINATION system network

The dictionary lookup function outlines in Algorithm 11 checks whether the word(s) of the CG node are in a dictionary. If so then select the associated feature and all the features that need to be preselected for the current one. This works as a naive backwards induction of features from a delicate one up to the least delicate one at the root (see Algorithm 1 in Section 7.6). An example of dictionary is presented in the Table 9.3 and the system network of NON-SPECIFIC determination in Figure 9.4.

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

Table 9.3 Dictionary example for the NON-SPECIFIC DETERMINATION system network

This section completed explanation of how the MOOD features are assigned to constituency graph. In the reminder of this chapter is addressed the task of assigning

TRANSITIVITY process types and participant roles to the constituent units (Costetchi 2013).

9.3 Creation of the empty elements

In the current work, when the participants are missing and are syntactically recoverable in the sentence then they are inferred from the structure and reference nodes are created for the missing elements. This phenomena is described in GB theory specifically the *Control and Binding* of empty elements (Haegeman 1991) introduced in Section 6.2. The *reference constituents* are important for increasing the completeness and accuracy of semantic analysis by making the participants and their afferent label explicit.

This stage is particularly important for semantic role labelling because usually the missing elements are participant roles (theta roles) shaping the semantic configuration. The most frequent are the cases of *control* where the understood subject of a clause is in the parent clause like in Example 119–121 where *Subj* is a generic subject placeholder introduced in Chapter 6 ad PRO, pro, t-trace and wh-trace.

(119) Poirot is considering whether [*Subj* to abandon the investigation].

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

(121) 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 Section 6.3 which relates GBT to Dependency Grammar and create placeholder constituents for them which are in the next step enriched with semantic roles. Currently the NP traces and PRO subjects are created with a set of graph patterns while the Wh traces are created with an algorithm.

9.3.1 The PRO and NP-Trance Subjects

The *xcomp* relation in DC can be encoded as an CG pattern graph (Figure 9.5) 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 6.2.4). They shall receive a the PRO subject constituent (governed or not) by the parent clause subject.

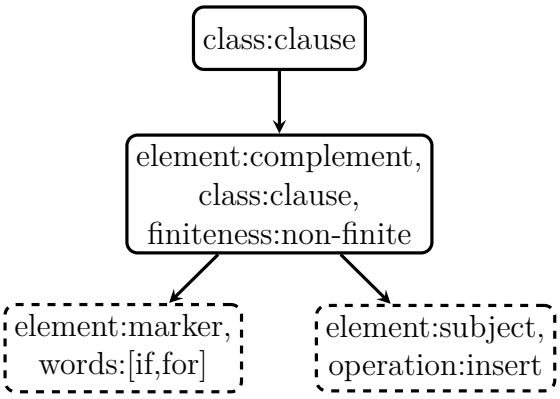


Fig. 9.5 CG pattern for detecting PRO subjects

The generalization 6.3.2 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 9.6 and respectively 9.7. In the case of Figure 9.7 the prepositional complements do not affect subject control in any way since it specifies only the nominal complements this making it complementary to 9.6 with respect to prepositional complements.

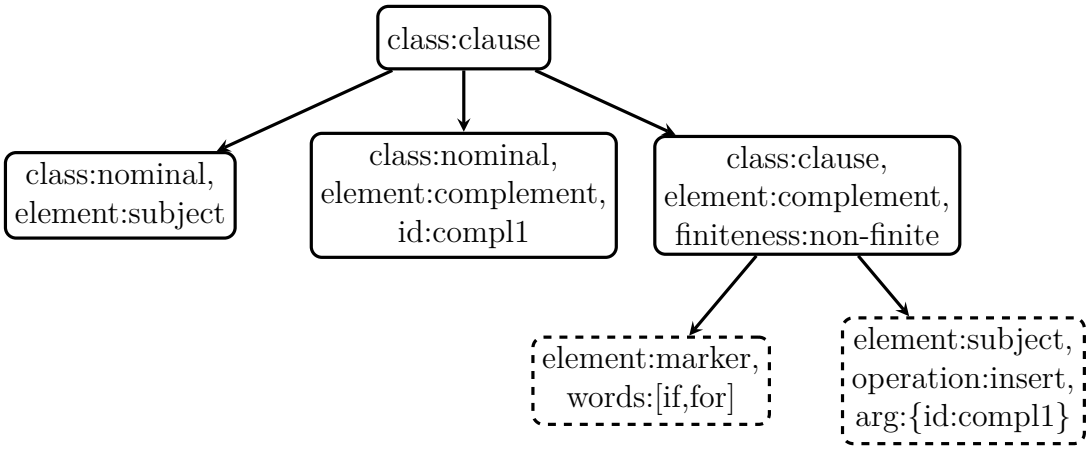


Fig. 9.6 CG pattern for obligatory object control in complement clauses

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 9.6 and 9.7 are applicable.

Before discussing each approach I would like to state a technical detail. When the empty constituent is being created it requires two important details: (a) the antecedent constituent it is bound to and (b) the type of relationship to its antecedent constituent or if none is available the type of empty element: t-trace or PRO. Now identifying the

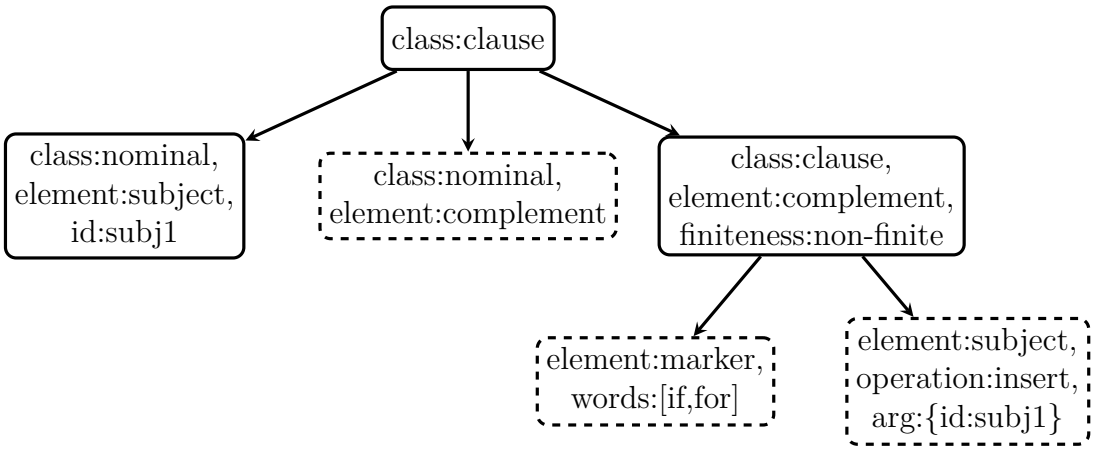


Fig. 9.7 CG pattern for obligatory subject control in complement clauses

antecedent is quite easy and can be provided at the creation time but since the empty element type may not always be available then it may have to be marked as partially defined.

The first solution is to create the empty subject constituents based only on syntactic criteria, ignoring for now element type (either PRO or *t*-trace) and hence postponing it to semantic parsing phase. The advantage of doing so is a clear separation of syntactic and semantic analysis. The empty subject constituents are created in the places where they should be and so this leaves aside the semantic concern of how the thematic roles are distributed. The disadvantage is leaving the created constituents incomplete or under-defined. Moreover the thematic role distribution must be done within the clause limits but because of raising, this process must be broadened to a larger scope beyond clause boundaries. This of course is an unwise approach as it might lead to unbounded dependencies and so unbounded complexity that needs to be addressed.

The second solution is to decide the element type before Transitivity analysis 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).

In the current work, however, the semantic roles are addressed within the clause borders following the principle of one main verb per clause thus avoiding the above mentioned risk of unbounded complexity. Also, the transitivity analysis is done based on pattern matching. This means a tremendous rise in complexity as the scope of a graph pattern is extended to two or more clauses. Instead, a desirable solution is

iteration over all the clauses in the CG (a sentence) and matching semantic patterns within the clause boundaries one at the time.

The solution adopted here is mix of the two described above and addresses the issues of: (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 9.7), (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 9.6).

If conditions from generalization 6.3.3 (from Section 6.3.2) 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 9.3.1 below.

Generalization 9.3.1. Constituents receiving thematic roles are marked with "thematic-role" label, those that do not receive a thematic role are marked with "non-thematic-role" and those that might receive thematic role with "unknown-thematic-role". So in each clause:

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

According to Generalization 6.2.9 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 6.2.9 is proposed the arbitrary interpretation.

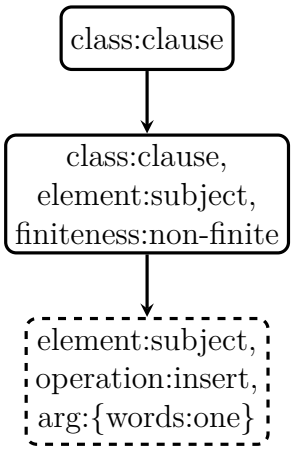


Fig. 9.8 CG pattern for arbitrary control in subject clauses

The pattern for subject control in subject clause is represented in Figure 9.8. 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 65-68.

9.3.2 Wh-trances

Creating constituents for 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 work this shall be expressed as graph patterns in order to be consistent with the general approach of the thesis. Algorithm 12 shows how it is currently done.

The first line of the algorithm checks if there are more than one clauses in the CG, otherwise it cannot continue. Then all the Wh-elements, their functions within the group and the clause shall be identified. Then iterate, from the lower to higher, over clauses and create the constituent corresponding to Wh-trace in the lower clause linked to the constituent in the parent clause. The pseudocode for creating the Adjunct or Theta Wh-traces are different and have been omitted in this section but are included in the Annexe .2.5.

After the null elements are created the constituency graph is ready for the semantic enrichment stage semantic configuration are assigned to each clause. The next sections introduce first how the PTDB has been normalised and how the graph patterns are created from it and then proceed with the semantic enrichment algorithm.

Algorithm 12: Creating the Wh-traces

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

9.4 Cleaning up the PTDB

The original version of the PTDB available on Neale's personal page[1] is not usable for computational purposes as such. It contains records applying a couple of different notations and sometimes informal, comments for human understanding, which from a machine standpoint is noise and cannot be processed as such. In this section I explain now how the original PTDB was transformed into a machine friendly form in order to be used as a lexical database.

The internal structure of the PTDB is detailed in Neale's PhD thesis (Neale 2002: 193–231). Here I focus on three columns which are of interest for the parsing purpose: the *verb form* (1^{st}), the Cardiff grammar *process type* (6^{th}) and the participant role *configuration* (8^{th}). Note that column numbers correspond to the original PTDB structure. After the transformation the PTDB column descriptions are as described in the Table 9.4. The field names in italic are the ones of interest and have been modified.

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

[1] see http://www.itri.brighton.ac.uk/~Amy.Neale/

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 9.4 The table structure of PTDB before and after the transformation

process type and the participant configuration. So they are important cues to consider 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 from the PROCESS-TYPE systemic network depicted in Figure **??**.

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* described in Fawcett (forthcoming). 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, in canonical form, 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. Sometimes, for directional or locational process types

the second or third participants may be functioning as Adjuncts, which currently complicates the matching process.

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 "?". In addition, "," (comma) and "&" (and) sign are use inconsistently with various meaning in all columns. Also, comments such as "not in Cob" were encountered in 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 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 used 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", "emotion plus xxx" became "emotive", "cognition xxx" became "two-role-cognition"). Annexe .2.5 provides the mapping across the process type versions.

Configuration column is one of the most important ones in PTDB. Checking its 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 prepending 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.

The Cardiff features column indicates the process type selected in the TRANSITIVITY system corresponding to one of the top levels depicted in Figure 4.4 provided in Chapter 4. Next follows a description of how the graph patterns are generated from the PTDB.

9.5 Generation of the TRANSITIVITY graph patterns

The configuration pattern graphs are used for CG enrichment executed through graph matching operation described in Section **??**. In previous section we saw how 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 9.9 depicts the prototypical template for generating three role CGP for canonic participant order that is active VOICE declarative MOOD. This pattern matches declarative clause with a subject and two complements and in case of success updates the constituents with process type and participant roles. When the configuration has only one or two participants the graph pattern indicates one or two negative complement nodes as depicted in Figure 9.10. In this section I do not aim at providing an exhaustive account of all the pattern forms but explain the principles by which they are automatically generated.

Besides the canonic CGP a set of variations are generated for each configurations corresponding to yes-no and Wh-Subj/Wh-obj/Wh-adj interrogative forms, imperative form and to passive voice form. When each of the variants are supported by the process type and participant configuration the flowing CGP are generated: (1) the declarative

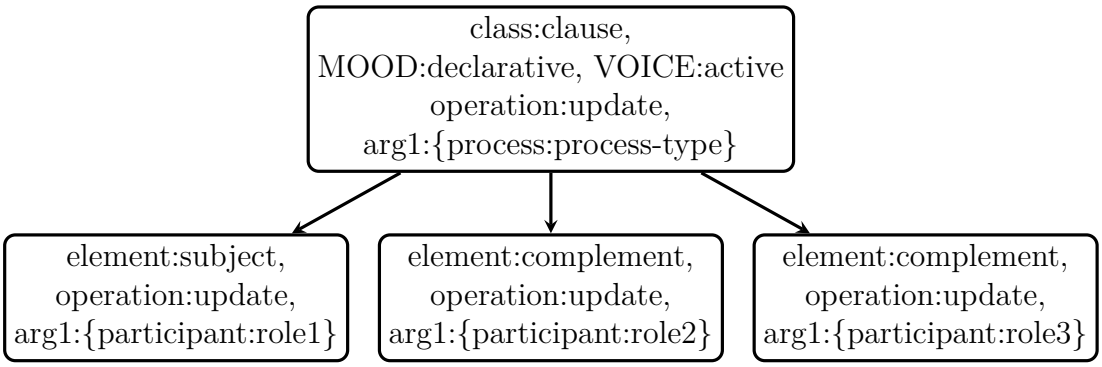


Fig. 9.9 Declarative MOOD and active VOICE graph pattern with three participant roles

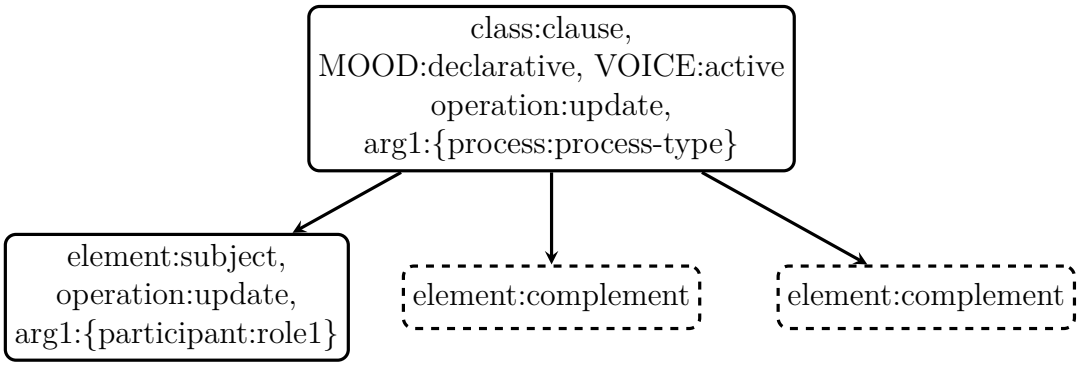


Fig. 9.10 Declarative MOOD and active VOICE graph pattern with one participant role

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 participant in the configuration is not the expletive "there" or the pleonastic "it" and the last role is not Agent role then both active and passive voice CPG are generated. Otherwise the passive form is not possible.

The imperative form CGP is generated if the first role of the configuration implies an active animate entity. Thus the nominal features of the subject must be already provided. Roles that accept imperative form are: Agent, Emoter, Cognizant, Perceiver and their compositional derivations e.g. Agent-Carrier, Agent-Cognizant, Affected-Emoter etc. Clausal subjects are excluded.

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

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

The process types are also grouped by the process type and number of participants which reduces the number of patterns be match per clause.

Algorithm 13: Generating the CPGs from the PTDB

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

The declarative MOOD active VOICE pattern (depicted in Figure 9.9) 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 9.11. Note that the role2 is assigned to the subject and the role 1 to the first complement. If the configuration accepts

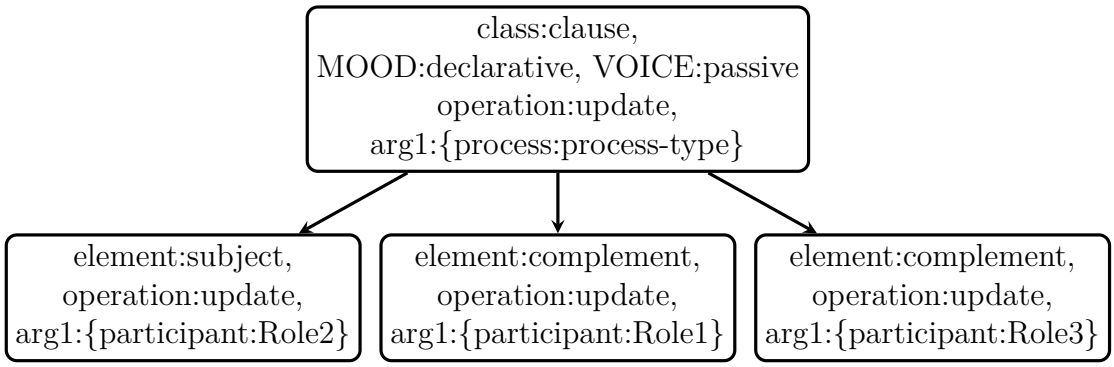


Fig. 9.11 Declarative MOOD and passive VOICE graph pattern with three participant roles

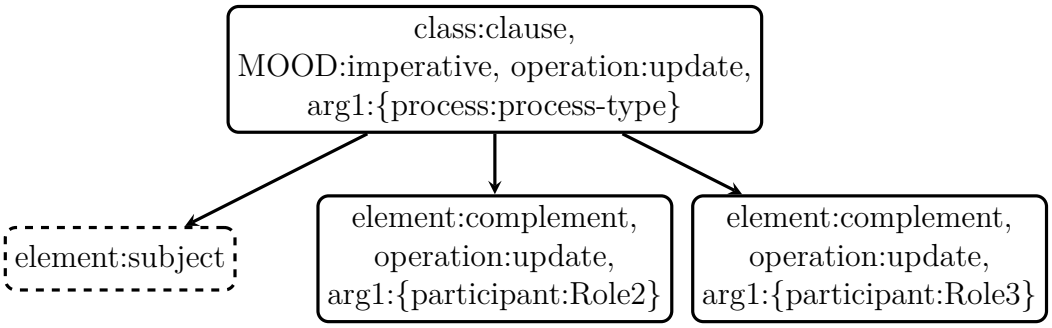


Fig. 9.12 Imperative MOOD graph pattern with three participant roles

imperatives then also a subject-less pattern graph is generated with first role omitted as depicted in Figure 9.12.

Directional processes are a special case. Examples 122 to 124 are equally valid configurations. Example 125 is a generic representation highlighting the optionality of these participants.

(122) The parcel travelled from London[So].

(123) The parcel travelled via Poland[Pa].

(124) The parcel travelled to Moscow[Des].

(125) The parcel travelled (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 9.5 below.

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

Table 9.5 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 patterns are generated in advance and su at the runtime they are ready for execution. This decreases the execution time. Next follows the description how the generated configuration graph patterns are used in TRANSITIVITY enrichment phase.

9.6 Enrichment with TRANSITIVITY features

Once the constituency graph has been enriched with MOOD and nominal DEIXIS features and the null elements created it is ready to be enriched with TRANSITIVITY features. The algorithm identifies, in each clause, the main verb and the potential participant constituents and searches in the PTDB the lexical item matching the main verb and any of its extensions and filters (based on the clause constituency) the possible configurations for the verb (comprising all the verb meanings). Then all the graph patterns corresponding to the possible configurations are executed on the clause, which, if matched will enrich the clause CG.

Algorithm 14 outlines the semantic parsing process implemented in the current parser which is a cascade of three loops. The first loop iterates through clauses in the mood constituency graph and for each the candidate process types are identified by considering: (a) the main verb, (b) the prepositions connected to it (either prepositional particles, or prepositions introducing a complement or adjunct prepositional phrase listed in Table 9.6) or (c) phrasal expressions such as "take a shower" which are explained in Section 4.2.3.

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.

Algorithm 14: 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
```

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 9.6 Prepositional constraints on participant roles

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

The Transitivity parsing results in multiple process configurations being assigned to clauses decreasing, in a way, precision and or introducing uncertainty. Because the final result contains few possibilities without knowing which one is the case. This uncertainty, decreased with several heuristics, is not dealt with by the current parser and shall be addressed in the future work.

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

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

9.7 Discussion

This chapter describes how the constituency backbone is enriched with syntactic and semantic features from MOOD and TRANSITIVITY system networks using graph patterns. Besides the core sections describing the enrichment algorithms this chapter provides multiple explanations about the graph pattern creation, CG extension with null elements and the clean-up of PTDB

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

The configuration patterns have been generated from the PTDB - a verb database accounting for Transitivity patterns in Cardiff grammar. In order to generate these patterns the PTDB had to be first cleaned up, unified and aligned to the present Transitivity system. Then for each configuration were generated a set of possible patterns varying based on mood, voice, process type and participants. Finally the semantic role labelling step employs the same pattern based enrichment mechanism as the one used in Section 9.2

The Algorithms can be improved 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 be towards higher abstraction and separation

between the grammatical constraints (in SFL represented as systemic realization rules) from the algorithm.

Chapter 10

The Empirical Evaluation

The present parser is evaluated by comparing the text analysis it outputs with manually analysed text. The evaluation seeks to measure the parser *precision* i.e. how many segments have been produced by the parser that are also found in manual analysis giving us; and the parser *recall* i.e. how many correct segments have been produced by the parser relative to the total number of produced segments.

In order to count correct and incorrect number of segments they need to be matched first. This task is not a trivial because of the annotation mistakes, some differences in grammar and methodology of treating certain phenomena described below. In this work slight divergences are tolerated provided that the segment label is the same and there is other segment that constitutes a better match. This is known, in computer science as the Stable Marriage problem defined in Section 10.4. The next two sections define segments and provide details on how and why they diverge. Section 10.3 describe how the segment labels differ and how they are mapped to extend the scope of the evaluation.

To establish the matches between manual and automatic segments I have implemented a variation of the Gale-Shapley algorithm (Gale & Shapley 1962) which is described in Section 10.4 below. Then, the section that follows, are finally described the empirical findings of the current evaluation.

In the current evaluation I use two sets of annotated texts. Table 10.1 summarises the corpora used for evaluation in this work.. The first dataset (OCD) was created by Ela Oren and I and is focused on syntactic constituency structure and clause Mood features. The texts represent blog articles of people diagnosed with Obsessive Compulsive Disorder (OCD) who self-report on the challenge of overcoming OCD. The second dataset (OE1) is the PhD work of Anke Shultz covering Cardiff Transitivity analysis. It comprises 31 files spanning over 1503 clauses and 20864 words. In addition

She provided another similar smaller corpus of 157 clauses that si also included into the evaluation.

Corpus	Elements	System Network	# characters (thousands)	# clauses	Annotator(s)
OCD	Syntax	Mood	16.2	147	Ela Oren & Eugeniu Costetchi
OE1	Semantics	Transitivity	51.8	1503	Anke Schultz
BTC	Semantics	Transitivity	5.6	157	Anke Schultz

Table 10.1 Evaluation corpus summary

The OE1 and BTC datasets have not been developed for the purpose of evaluating the current parser however they are compatible and can be adapted to evaluate (a) the boundaries of a constituent segment, (b) the constituent semantic function and (c) some Transitivity features. To enable each of these evaluations the annotation data and parser output need to be represented in such a way that they are comparable to each other. Specifically the feature names had to be harmonised with the ones from PTDB. The adjustments were the same as described in Section 9.4.

10.1 Segment definition

To compare the segment boundaries we need to understand how they are represented in each output and how they can be brought to a common for comparison.

Listing 10.1 Segment example in UAM corpus tool

```
<segment numbersid="4" numbersstart="20" numbersend="27"
numbersfeatures="configuration;relational;attributive"
numbersstate="active"/>
```

Both datasets were created with UAM Corpus Tool (O'Donnell 2008a,b) version 2.4. The annotations, in this software, are recorded as segments spanning from a start to an end positions in the text file together with the set of features (selected from a systemic network) attributed to that segment. There are no constituency or dependency relations between segments. In Listing 10.1 is the XML representation for an example annotation segment where the *id* attribute indicates the unique identification number within the annotation dataset, the *start* and *end* attributes define the segment between two character offsets relative to the beginning of the text file.

Listing 10.2 Raw text example in annotation data

```
0 0Red riding hood excerpt24
1 25"What have you in that basket,  Little Red Riding Hood?"82
```

```
2 83
3 84"Eggs and butter and cake, Mr. Wolf."111
```

Listing 10.2 presents an example raw text from the annotation dataset containing an initial line and two sentences separated by an empty line. The greyed numbers in the beginning and end of each line indicate character offset. In some files, the first line plays the role of a header containing a title the file name. In this example it is a title. Either way, this, first line, is neither considered for annotation nor for parsing. Some files contain one sentence per line, others separate sentences by an extra blank line and in other files, the text is one continuous block. The text may contain sporadically tabs and blank spaces such as in the line 1 between the comma and the Little Red Riding Hood.

Parsimonious Vole, parses sentences one by one and not whole text document at once. Before parsing, the document is chunked into sentences guided by the punctuation delimiters. In addition, the Stanford dependency parser normalises the input text by trimming spaces and tabs, adjusting character encoding and replaces some special characters such as quotes, brackets, punctuation, etc.

Listing 10.3 presents the preprocessed text before it is parsed. The title line is cropped from the file. Each line contains a separate sentence. The text is tokenised such that words and punctuation marks are separated by a single space.

Listing 10.3 Parser preprocessed text example

```
0 0" What have you in that basket , Little Red Riding Hood ? "60
1 61" Eggs and butter and cake , Mr. Wolf . "102
```

After the parser preprocessing process, the input and output texts are no longer identical. The sentence and word offsets are no longer reliably findable with respect to the original text and the segments need to be matched. There are two fundamental kinds of segment misalignments: the segment offset shifting and the segment resizing (either shrinking or enlarging). Sometimes it can also be a combination of the two. Allen interval algebra (Allen 1983) systematizes the basic thirteen relations to position segments relative to each other. Nonetheless in the current work we are not interested to investigate how the segment offset shifts but rather what would be an efficient way to match them independent of their position.

To make manual and automatic annotations comparable, the constituency graphs need to be reduced to a set of segments corresponding to each constituent. The first task of the evaluation is to find matches or near-matches between two segment bundles that is addressed in the next section.

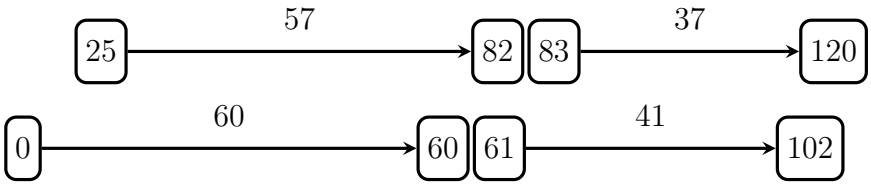


Fig. 10.1 Graphic representation of the sentence segment misalignment between Listing 10.2 and Listing 10.3

10.2 Reducing the CG to a set of segments

If we ignore the set of associated labels (systemic features), which we will consider in the next section, then two segments are we identical when (a) their start and end indexes and (b) the text in between is identical.

To address this issue in the current evaluation, the text processed by the parser is re-indexed back into the original raw text at the level of words (tokens), constituents and sentences. The high level perspective for generating segments for the constituency graphs indexed on the original text file is provided in Algorithm 16.

Algorithm 16: Algorithm to generate segments for the CG bundle indexed on the raw text

```
   input  : CG bundle, text
1  begin
2  |  offset ← 0
3  |  for cg in CG bundle:
4  |  |  generate segments for cg indexed on text given the offset
5  |  |  offset ← the end of cg
6  end
```

The result of the Algorithm 16 is a set of segments that are indexed according to the raw text by iterating the resulting constituency graphs one by one and indexing each with respect to the offset given by the previous one.

The way each CG is indexed is described by the Algorithm 17 which returns a set of segments from the constituency graph knowing given an offset. First a token level indexing is performed and corresponding segments generated. Then for each CG constituent, segments are generated based on its word span. The indexes are set from beginning of the first word to the end of the last word. The labels of the segment are the CG class, functions and systemic features discussed in the next section.

$$d = \sqrt{(start_S - start_T)^2 + (end_S - end_T)^2} \tag{10.1}$$

Algorithm 17: Algorithm to generate segments for CG constituents indexed on the raw text

```
   input  : cg, text, sentence offset
 1 begin
 2     words ← get cg the list of words
 3     for word in list of sentence word segments:
 4         find the word in the text after a given sentence offset
 5         if word found:
 6             start ← get first word start index
 7             end ← get the last word end index
 8             create a new segment (start, end, word)
 9         else:
10             generate a warning (manual adjustment needed)
11     for node in cg in BFS postorder:
12         find the word span of the constituent
13         start ← get first word start index
14         end ← get the last word end index
15         labels ← get node class, function and features
16         create new segment (start, end, labels)
17     return set of segments
18 end
```

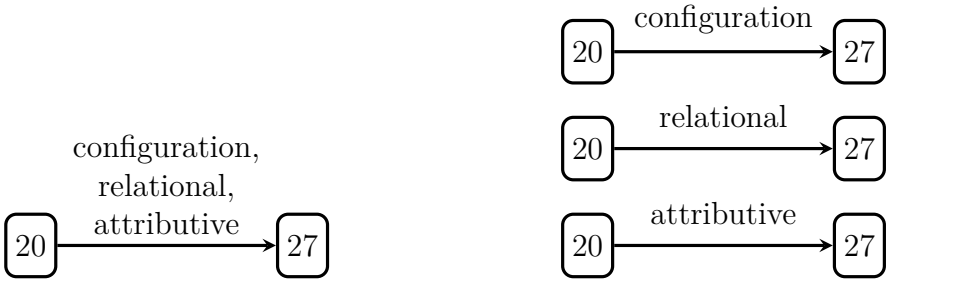
$$d = \sqrt{\Delta^2_{start} + \Delta^2_{end}} \tag{10.2}$$

Later in this chapter I address how the manually created segments and parser generated segments are aligned. Here it is noteworthy to mention that the manually created segments contain errors. Some segments are either shifted and include the adjacent spaces or, the converse, leave out one or two characters of a marginal word. For this reason the segment alignment process needs to tolerate a certain distance between the stand/end indexes of aligned segments. I adopted the geometric distance to measure the difference between the segment start and end indexes. For two segments $S(start_S, end_S)$ and $T(start_T, end_T)$ the geometric distance is defined in Equation 10.1. We can replace the difference between start and end indexes with Δ_{start} and Δ_{end} notation and obtain a reduced form provided in Equation 10.2.

10.3 Considering the segment labels

Provided that the task of indexing the segment indexes and content such that they "roughly" correspond to each other we need to start comparing the segment labels. However the labels of manually created segments differ from the segments provided by the parser. Besides shift in the offset mentioned above, another difference is in number of ascribed segments. In many instances, the manual annotation is less delicate and thus contain less features than the one provided by the parser. Still, some of the manual annotations contain features that are not provided by the parser (i.e. not yet implemented), such as the Thematic structure of the clause or the Modal Assessment features.

To address this issue, in the current evaluation, the segments are reduced to carry only one label. The consequence is that the segments with multiple features (Figure 10.2a) are broken down into multiple segments with the same span (Figure 10.2b) for each feature. When each segment contains exactly one feature the evaluation can be focused on one or a set of features of interest by selecting only the segments that contain exactly those.



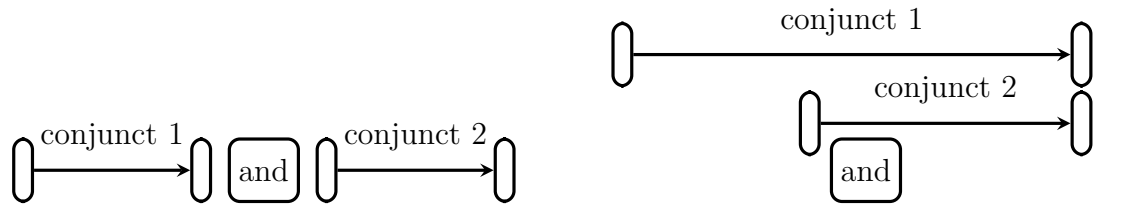
(a) A segment with a set of features (b) A set of segments with single features

Fig. 10.2 Example of breaking down a segment with multiple features into set of segments with a single feature

There are more differences in the grammar and annotation style between the manual and automatic processes. This means that both structure and feature names provided by the system networks differs (already mentioned above that the Theme system network is not implemented in the current version of the parser but provided in some manual annotations).

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 at the end of each adjacent segment. None of the conjunctions (such as "and", "but", "so", etc.) are included into the conjunct segments, they are considered markers in

the clause/group complexes rather than part of the constituent. The treatment of conjunction is discussed in Section 3.4.6. The parser, on the other hand, includes the conjunctions into the following adjacent segment. Moreover the conjuncts spans differ as well. Instead of being annotated in parallel, as depicted in Figure 10.3a, the segments are subsumed in a cascade from the former to the latter as depicted in Figure 10.3b.



(a) Conjuncts annotated as parallel segments (b) Conjuncts annotated as subsumed segments

Fig. 10.3 Treatment of conjunctions in the corpus compared to the parser

The differences in the grammar are due to the slight variations in the feature names and in the system network structure. For example the element names are spelled differently "predeictic" and "pre-deictic", "root" and "clause" or "explative-marker" and "expletive". Similar spelling variations are present in the feature names for example "two role" and "two-role-action" and other variations alike. To remedy this issue the current evaluation establishes mappings between the manual and automatically generated features.

Now that the main divergences have already been outlined, I proceed to explain how the stable matches between the segments are established.

10.4 The matching algorithm

The task of aligning two sets of labelled segments is almost the same as the well know problem in computer science called *stable marriage problem* (Gusfield & Irving 1989). The standard enunciation of the problem is provided below and is solved in an efficient algorithm named Gale-Shapley (Gale & Shapley 1962) after its authors.

> Given n men and n women, where each person has ranked all members of the opposite sex in order of preference, marry the men and women together such that there are no two people of opposite sex who would both rather have each other than their current partners. When there are no such pairs of people, the set of marriages is deemed stable Iwama & Miyazaki (2008).

In the context of this evaluation the group of men is associated with the segments generated automatically by the parser and the group of women with the segments available from the manual analysis.

The standard stable marriage problem is formulated 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 the preferences of every individual are known and expressed as a complete ordered list of individuals from the opposite group ranging from the most to the least preferred one. Thus the preference list must be *complete* and *fully ordered*.

To fulfil these requirements I construct a distance matrix from each automatically created segment to every and manually created one. The distance measure considered here is the Euclidean one provided in Equation 10.2 above. The matrix represents the complete and fully ordered set of preferences stipulated in the original problem formulation. In addition to distance the segments need to carry the same labels in order to be considered a match. This condition is not found in the original problem but is considerer in Algorithm 18 below.

Algorithm 18: The algorithm for matching automatic and manual segments

```
   input  : aut segments, man segments
 1 begin
 2     mark all aut segments and man segments free
 3     compute distances from each man segments to every aut segments
 4     while ∃ free aut segments:
 5         aut ← first free from aut segments
 6         if ∃ man segments not yet tested to match aut:
 7             man ← the nearest among man segments to aut with identical label
 8             if man is free:
 9                 match aut and man
10                 mark aut and man as non-free
11             else:
12                 aut′ ← the current match of man
13                 if aut is closer to man than aut′:
14                     match aut and man
15                     mark aut and man as non-free
16                     mark aut′ as free
17         else:
18             mark aut as non-free and non-matching
19     fintq
20 end
```

The input to the algorithm is the set of automatically generated segment list aut segments and the manually created segment list man segments. It begins with initialising all the segments as *free* meaning that none of them are matched yet (originally married). The segments can be marked either be as *non-free matching* or *non-free non-matching*. A Part of the initialisation is also creation of the distance matrix mentioned above representing a complete and ordered set of preferences for each segment.

The algorithm iterates over the aut segments for as long as there exist free ones. If there exists a man (manually created segment) which have not been attempted to match aut then attempt matching aut to the nearest man that has the same label. If man is free then it is a match. Otherwise compare aut to aut′ (currently assigned match of man) and then consider a match with the nearest one (i.e. shortest distance). When there are no man segments to test mark aut as non-free and non-matching meaning that it shall no longer be considered by the algorithm.

In the end the algorithm provides a list of successful matches out of which we can also deduce which aut segments man segments have not been matched. The next section provides the empirical measurements resulting from applying the current method.

10.5 The measurements

In this section I present a series of measurements addressing two quality criteria of the parser. Firstly, the degree of divergence in segment spans between manual and parse segments in all corpora (OCD, BTC and OE1). And secondly, the accuracy of the parser with respect to manual annotations measured through precision, recall and F_1 measures. The syntactic constituency and some Mood features are derived from OCD corpus only, followed by the semantic constituency and Transitivity features evaluation based on OE1 and BTC corpora.

10.5.1 Segment divergence: general findings

The segments don't perfectly align, as described in previous sections, due to minor differences in annotation approach, text normalisation and trimming before parsing (that was not performed before the manual annotation), errors in the annotations (missing or including characters). The misalignment is measured using the geometric distance on segments matched with Algorithm 18. The presented statistics are calculated on a number of over 12500 segment pairs whose mean distance is 4.84 with a standard deviation of 14.51. The distance is distributed between a minimum 0 and

maximum 219. A mean close to the minimum point and a large standard deviation indicates that most of the data points are situated between 0 and slightly over the mean continued by a very long tail of rare and distant data points. This intuition can be seen in the histogram depicted in Figure 10.4 below. This dataset resembles the Pareto distribution where 74% are almost not shifted (by up to 1 character) or 83% of the segments are slightly shifted up to 5 characters.

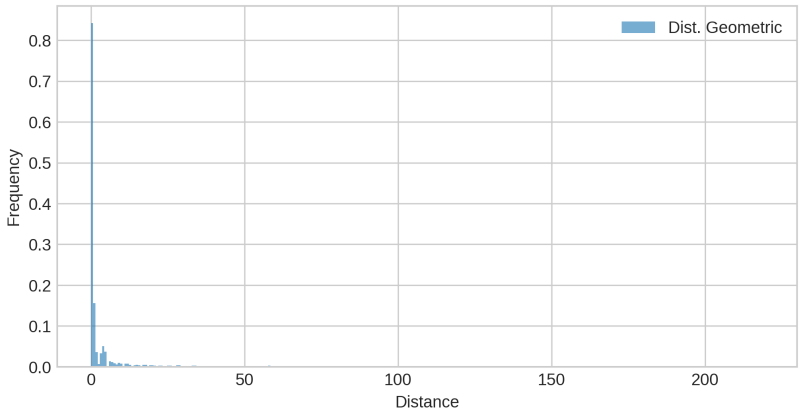


Fig. 10.4 Full histogram of the distance distribution of the matched segments (binning=300)

Because of a very long tail, I reduce the further analysis to the distance span between 0 and 40. Figure 10.6 depicts the histogram of distances between matched segments (in 300 bins). It shows that most segments (89%) are cumulated in first two bins with distance up to 2. The consequent three bins also contain a significant portion of segments with distance up to 5. The rest of the bins, represent a very long tail, spanning on distances up to maximum 219 and containing a small amount of segments.

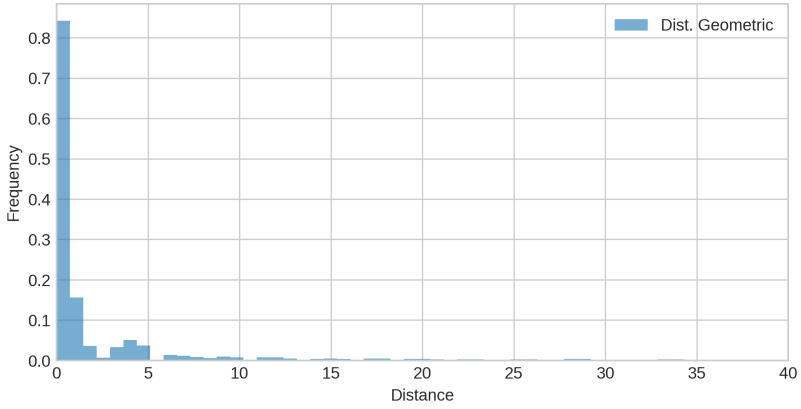


Fig. 10.5 Reduced histogram of the distance distribution of the matched segments (binning=300). View reduced to the a distance of 40 characters

This distribution can be viewed in it's cumulative form depicted in Figure 10.6. Here we see that over 51% of segments are perfectly aligned. 80% of the segments are slightly shifted up to a distance of maximum 5 characters. This can be explained by differences in (a) punctuation, (b) conjunction and (c)verbal group treatment described above. The next 5% of the segments are shifted between 5 and 10 characters. The last 10% cover the heavy shift of distances 20 to 219. This may be due 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).

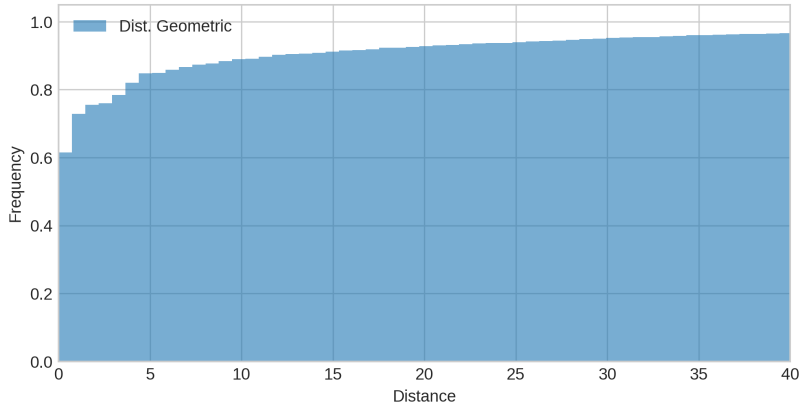


Fig. 10.6 Cumulative histogram of the distance distribution of the matched segments (binning=300). View reduced to the a distance of 40 characters

10.5.2 Segment divergence breakdown by element type

If we take a closer look at the Figure 10.5 we will notice some spikes and a tendency towards some local values. What would the histogram look like if we group values in a different manner considering these spikes and the long tail of the distribution. In Figure 10.5 we can observe several dips at around distances 3, 6, 11, 13, 17, 19, 22, 24 and so on. I decided to consider these dips for deciding bin borders in the new bin design. And, since the tail is very long with very few values, as the bins advance to the left, each is designed over longer span this way compressing the tail. In Table 10.2 is presented such binning design. Also each interval is assigned a category that means the degree of deviation.

Besides custom binning described above, looking at each segment label in part brings more clarity to the evaluation. The most relevant segment labels are the unit elements. In the current evaluation, the annotators provide clause level syntactic

degree	insignificant	tiny	little	moderate	significant	high
distance intervals	0-3	3-5	5-10	10-20	20-50	50-250

Table 10.2 The progressive binning scale considering the dataset properties

(Subject, Complement, Adjunct, Main verb, Finite) and semantic (Configuration, Main verb, Participant role) elements. These elements are used a dimension in the breakdown of the segment divergence that follows.

Using the bins defined in Table 10.2 the distribution of segment deviations grouped by the main syntactic elements is provided in Table 10.3 that is also depicted in Figure 10.7.

element	**% of segments per degree of deviation**					
	insignificant (0-3)	**tiny (3-5)**	**little (5-10)**	**moderate (10-20)**	**significant (20-50)**	**high (50-250)**
Main verb	29.38	0.95	0.30	0.10	0.00	0.05
Subject	22.70	0.10	0.25	0.30	0.05	0.00
Adjunct	13.86	0.45	0.60	0.15	0.85	0.20
Complement	12.10	0.75	1.46	2.26	1.96	1.86
Finite	9.19	0.00	0.00	0.00	0.05	0.05

Table 10.3 Percentage of segments deviated to a given degree for major syntactic elements

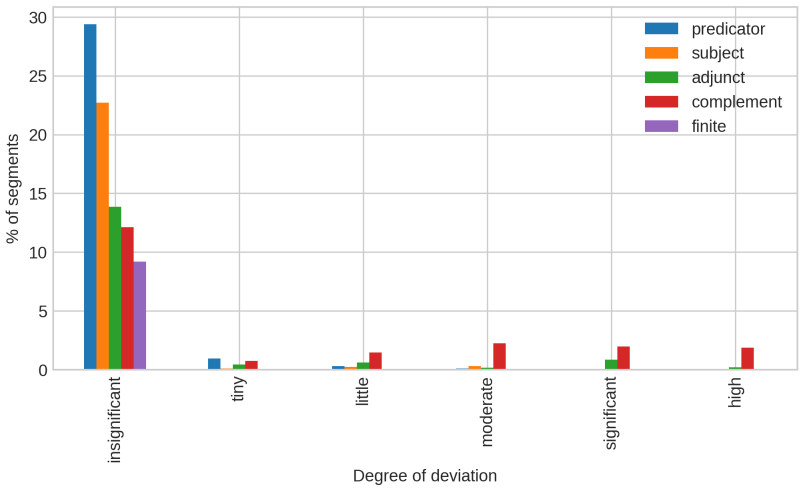


Fig. 10.7 Bar chart of the segments deviated to a given degree for major syntactic elements

In Figure 10.7 we can see that most deviations are insignificant. The number of segments in the rest of the bins, from tiny to high, is below 1% in every bin which perhaps reflects errors in annotation and/or matching algorithm requiring further

investigation. In case of complements, however, the proportion of segments is slightly higher (0.7–2.2%). This may be explained by the problem of prepositional phrase attachment and further analysis is needed to test this hypothesis.

When we switch to a grouping by the main Transitivity elements maintaining the same bins defined in Table 10.2, then the distribution of segment deviations looks as outlined in Table 10.4. The same data are depicted in Figure 10.8.

feature	**% of segments per degree of deviation**					
	insignificant (0-3)	**tiny (3-5)**	**little (5-10)**	**moderate (10-20)**	**significant (20-50)**	**high (50-250)**
participant-role	36.78	2.21	3.48	2.80	3.04	1.67
main	20.75	3.48	1.23	0.39	0.00	0.00
configuration	7.75	3.73	2.80	3.04	4.56	2.31

Table 10.4 Percentage of segments deviated to a given degree for major semantic elements

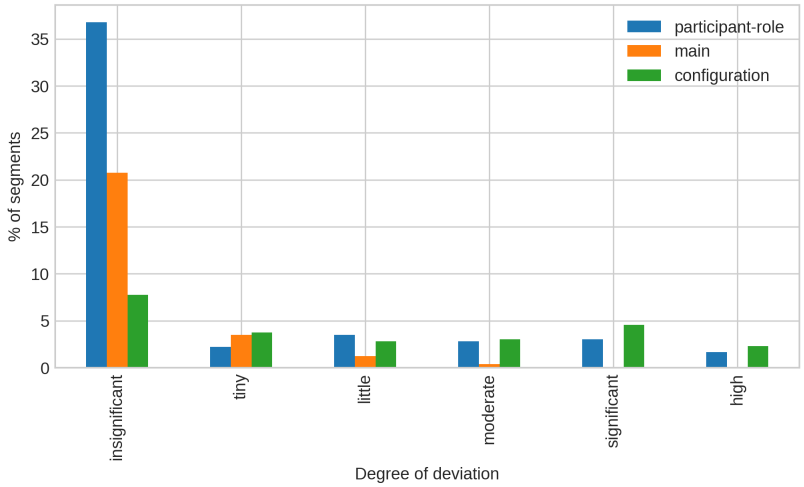


Fig. 10.8 Bar chart of the feature segments deviated to a given degree for major semantic elements

Similarly to Figure 10.7, in Figure 10.8 we can see that most of the deviations are insignificant (0-3 characters). In the rest of the bins representing higher degrees of deviation the amount of segments is up to 4.7% which reflects higher number of shifted segments. One exception is the main verb whose occurrences decreases towards higher degrees of deviation, i.e. it is shifted by an insignificant, tiny or little degree that is up to about 10 characters. This is explainable by the fact that the main verb is most of the time (if not always) a single word, which is always shorter than the configuration or participant elements which usually span clauses of groups comprising multiple words.

The large variations in participant seems to correlate with complement variation and might be explained by the attachment errors. In cases of high configuration deviation however further investigation are needed because these may possibly be errors in the segment matching algorithm or other unknown anomalies.

10.5.3 Syntactic evaluation: Constituency elements

The evaluation data in this and the following sections will be presented in tables with the same structure. Using Table 10.5 as example I explain next what the columns mean. The first column will contain the name of the unit type, element or feature. The next three columns *Match*, *Manual mn* and *Parse mn* represent the number of segments that are considered identical between the corpus and parser, the number of unmatched the corpus (manually created) segments and the number of unmatched parser (automatically generated) segments. The next three columns *Precision*, *Recall* and F_1 represent standard accuracy metrics indicating the fraction of relevant instances among the retrieved instances, fraction of relevant instances that have been retrieved over the total amount of relevant instances and the harmonic mean of the previous two. In addition, the column *%Total matched* represents the percentage the current item (row) in the table while the *% Manual nm* and *%Parse nm* represent the number of remaining unmatched segments of the current item (row) that represents a translation of the Manual and Parse mn columns into relative terms.

Unit type	Matched	Manual nm	Parse nm	Precision	Recall	F1	%Total matched	%Manual nm	%Parse nm
clause	612.00	64.00	78.00	0.89	0.91	0.90	37.00	9.47	11.30
nominal group	717.00	108.00	67.00	0.91	0.87	0.89	43.35	13.09	8.55
prepositional group	119.00	39.00	39.00	0.75	0.75	0.75	7.19	24.68	24.68
adverbial group	161.00	79.00	103.00	0.61	0.67	0.64	9.73	32.92	39.02
adjectival group	45.00	36.00	38.00	0.54	0.56	0.55	2.72	44.44	45.78

Table 10.5 The evaluation statistics for the main constituency unit types

The syntactic accuracy aims to measure how well the main unit types and the clause main elements have been detected by the parsed compared to the corpus. The evaluation is performed on the OCD corpus. This evaluation is restricted to the clause and four group types: nominal, prepositional, adverbial and adjectival. No clause complexes, group complexes or word types are included. The evaluation results are presented in Table 10.5 where we can see that clauses and nominal groups have an F_1 score of about 90%. Prepositional, adverbial and adjectival group scores decrease to 55% and requires investigation of the errors in the parsing and matching algorithms. There is also a contrast in the number of segments, visible in the bar chart from Figure

10.9, between the first two element types and the last three with a ratio of one to four or more.

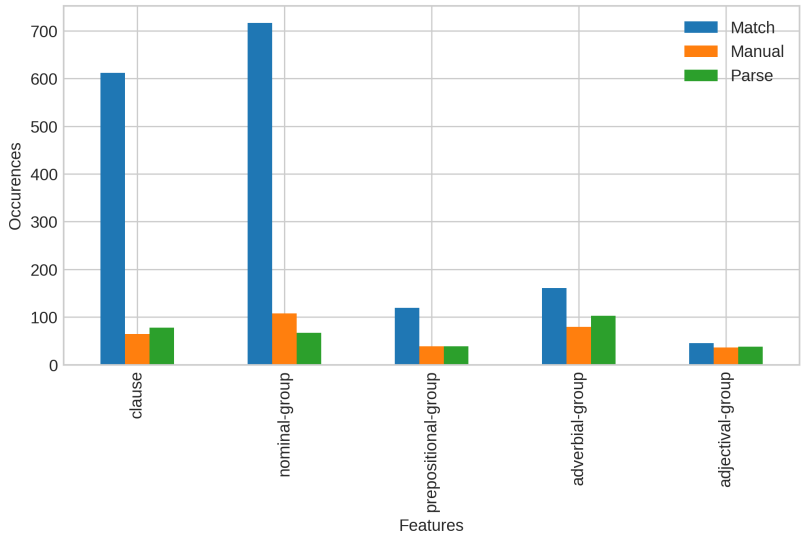


Fig. 10.9 Bar chart of matched and non-matched (manual and parse) segments of the main constituency unit types

Table 10.6 presents the evaluation result for the main clause elements. Some of them such as, Auxiliary verbs, Main verb extension, Negation particle, and others have been omitted in the corpus and thus missing in the present evaluation. For the Main verb and Subject elements the F_1 measure raises to 90%. For the complements and finite the F_1 score is 67% and 63%. Surprisingly, the Complements have a small number of corpus unmatched segments and a high number of parser unmatched segments. This is explained by a flaw in the annotation methodology because the clausal complements were often annotated directly as new clause and omitting to draw the same segment with complement element. This required the corpus revision and correction. Adjuncts however have a higher number of unmatched segments on both sides and this may be dues to bugs in the parser.

Clause element	**Matched**	**Manual nm**	**Parse nm**	**Precision**	**Recall**	**F1**	**%Total matched**	**%Manual nm**	**%Parse nm**
Main verb	613	60	79	0.89	0.91	0.90	30.79	8.92	11.42
Subject	466	22	86	0.84	0.95	0.90	23.41	4.51	15.58
Complement	406	43	350	0.54	0.90	0.67	20.39	9.58	46.30
Adjunct	321	159	224	0.59	0.67	0.63	16.12	33.12	41.10
Finite	185	3	392	0.32	0.98	0.48	9.29	1.60	67.94

Table 10.6 The evaluation statistics for the clause main elements

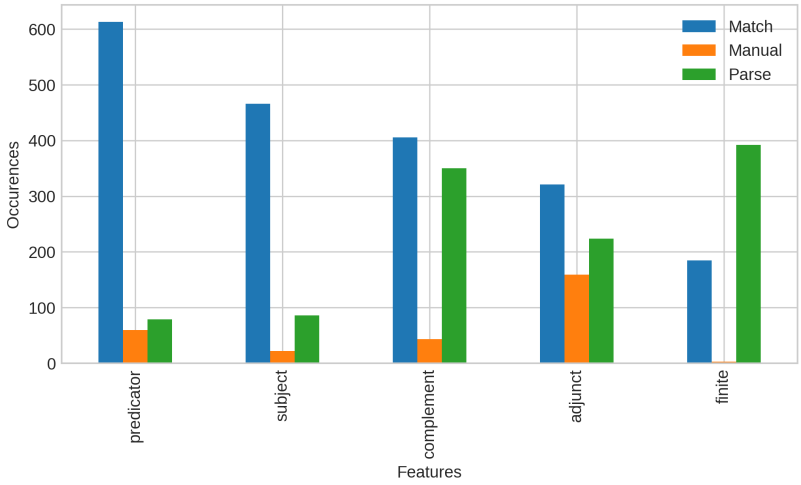


Fig. 10.10 Bar chart of matched and non-matched (manual and parse) segments of the clause main elements

10.5.4 Syntactic evaluation: Mood feature selections

In this section I present the evaluation of Mood system network selections. The corpus contains selections from a pat of the system network that is depicted in Figure 10.11. The full system network is provided in Figure 4.1. Employing the entire system network in the annotations was difficult because as the delicacy increases the time spent for the annotation process increases.

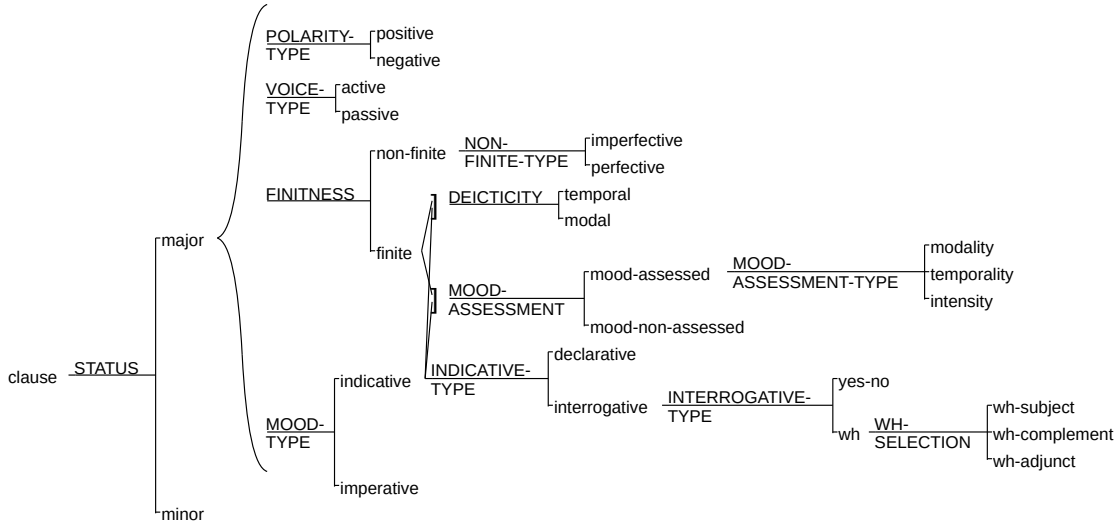


Fig. 10.11 The part of the Mood system network that has been used in OCD corpus annotation

Table 10.7 summarises the evaluation results for MOOD system network features grouped by system names (in capital). The precision and recall values vary quite a lot

Feature	Matched	Manual nm	Parse nm	Precision	Recall	F1	%Total matched	%Manual nm	%Parse nm
POLARITY-TYPE									
positive	485.00	125.00	55.00	0.90	0.80	0.84	89.48	20.49	10.19
negative	57.00	10.00	70.00	0.45	0.85	0.59	10.52	14.93	55.12
VOICE-TYPE									
active	553.00	102.00	68.00	0.89	0.84	0.87	98.05	15.57	10.95
passive	11.00	11.00	28.00	0.28	0.50	0.36	1.95	50.00	71.79
FINITNESS									
non-finite	99.00	19.00	38.00	0.72	0.84	0.78	15.84	16.10	27.74
finite	526.00	33.00	554.00	0.49	0.94	0.64	84.16	5.90	51.30
NON-FINITE-TYPE									
perfective	71.00	12.00	16.00	0.82	0.86	0.84	73.20	14.46	18.39
imperfective	26.00	9.00	24.00	0.52	0.74	0.61	26.80	25.71	48.00
DEICTICITY									
temporal	446.00	74.00	55.00	0.89	0.86	0.87	97.38	14.23	10.98
modal	12.00	33.00	6.00	0.67	0.27	0.38	2.62	73.33	33.33
MOOD-ASSESSMENT-TYPE									
temporality	35.00	17.00	27.00	0.56	0.67	0.61	56.45	32.69	43.55
modality	15.00	32.00	8.00	0.65	0.32	0.43	24.19	68.09	34.78
intensity	12.00	14.00	43.00	0.22	0.46	0.30	19.35	53.85	78.18
MOOD-TYPE									
indicative	455.00	216.00	37.00	0.92	0.68	0.78	99.13	32.19	7.52
imperative	4.00	1.00	31.00	0.11	0.80	0.20	0.87	20.00	88.57
INDICATIVE-TYPE									
declarative	355.00	260.00	27.00	0.93	0.58	0.71	88.31	42.28	7.07
interrogative	47.00	7.00	63.00	0.43	0.87	0.57	11.69	12.96	57.27
INTERROGATIVE-TYPE									
wh	40.00	6.00	57.00	0.41	0.87	0.56	88.89	13.04	58.76
yes-no	5.00	3.00	8.00	0.38	0.62	0.48	11.11	37.50	61.54
WH-SELECTION									
wh-subject	9.00	3.00	7.00	0.56	0.75	0.64	32.14	25.00	43.75
wh-adjunct	11.00	15.00	3.00	0.79	0.42	0.55	39.29	57.69	21.43
wh-complement	8.00	0.00	62.00	0.11	1.00	0.21	28.57	0.00	88.57

Table 10.7 The evaluation statistics for POLARITY-TYPE systemic choices

from a minimum of 11% up to a maximum of 93% and the harmonic mean, the F_1 score, between 30% up to 87% averaging to almost 60%. The details can be read in Table 10.8. The distribution of these values can be seen in Figures 10.12 and 10.13. A noticeable feature is the presence of two peaks in the precision and recall distributions: one around 50% and the other one around 90%. They translate into a similar F_1 distribution with peaks at 60% and 85%.

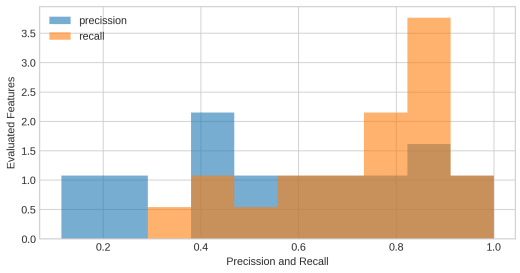


Fig. 10.12 The distribution of precision and recall for selected features from the Mood system network

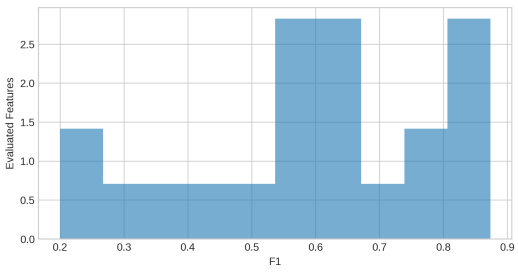


Fig. 10.13 The distribution of F1 score for selected features from the Mood system network

	precision	**recall**	**F1**
feature count	21.00	21.00	21.00
mean	0.57	0.73	0.59
standard deviation	0.27	0.18	0.21
min value	0.11	0.32	0.20
25% quantile	0.41	0.62	0.48
50% quantile	0.56	0.80	0.61
75% quantile	0.82	0.86	0.78
max value	0.93	1.00	0.87

Table 10.8 Descriptive statistics of the precision, recall and F1 scores

Within most systems, the F_1 scores exhibit high contrast from one feature to the other. What does it means is discussed in the next two cases. For example in POLARITY-TYPE system the positive polarity feature scores 84% F_1 measure and negative one almost 60%. As per Definition 3.2.10, the system features are mutually exclusive. The polarity of an English clause is positive by default unless a negation marker is found and this represents only 10% of the clauses in the corpus. This means that it should be sufficient for the parser to detect one feature with a reasonably high F_1 score then the converse feature should be detectable with the same degree of accuracy. Yet the current data invalidate this hypothesis.

In case of POLARITY-TYPE system, the phenomena may be explained, as a consequence of low delicacy in the parsing algorithm. Current implementation determines polarity by checking for the presence of the negation verbal marker only. Nonetheless a more delicate polarity testing will have to take into consideration polarity indicators from the subject, complement and adjuncts of various type that may have been taken into consideration during the annotation process. Providing an incomplete, less delicate implementation for systemic choices may be a source of errors. This hypothesis requires further investigation.

Nonetheless, the same phenomena can be seen if we look at the systems that do not span other ones with higher degrees of delicacy. For example, detection mechanism for VOICE-TYPE is implemented similarly to POLARITY-TYPE. The parser checks whether there is a passive order of elements in the clause otherwise the active voice is selected. Detection of the positive voice scores a much higher F_1 measure of 87% than the negative one of 36%. There is no more the problem of low delicacy and still there is a discrepancy between the F_1 scores of the two features.

This high discrepancy between the feature F_1 scores can be seen as follows. Most instances of active voice are easy to detect. But there is a portion of cases, regardless of

the voice, that are difficult for the parser to distinguish. The passive voice selections are executed mostly for these ambiguous cases. More investigation is required to pinpoint exactly how such phenomena materialises.

Nonetheless, in the future implementation, we can take advantage of the mutual exclusivity of system features and capitalise on the above finding, at least for the leaf systems. Taking into consideration that the score of one feature is higher, clause elements should then be provided with selections of those features only. While the complement features shall remain unselected. This means that only features that can be provided with a high confidence (using F_1 score as a measure of that) shall be assigned. Then an additional process can be implemented to select the remaining unassigned features.

10.5.5 Semantic evaluation: Constituency elements

This section describes the empirical findings regarding the semantic parsing. This evaluation is based on the OE1 and BTC corpora created by Anke Schultz for her PhD project. The text is annotated with Cardiff Transitivity system network. The employed elements are Configuration, Participant role and Main verb and Circumstances are excluded from the study. Table 10.9 summarises the empirical findings. The F_1 score of about 82%, precision of about 74% and recall of about 92% for these elements is surprisingly stable across elements compared to variate statistics of the syntactic elements presented in Table 10.6 above. This difference may be explained by higher annotation quality in the corpus.

	Match	**Manual nm**	**Parse nm**	**Precision**	**Recall**	**F1**	**%Total matched**	**%Manual nm**	**%Parse nm**
Participant role	2780	233	1002	0.74	0.92	0.82	49.54	7.73	26.49
Configuration	1376	127	493	0.74	0.92	0.82	24.52	8.45	26.38
Main Verb	1456	160	605	0.71	0.90	0.79	25.94	9.90	29.35

Table 10.9 The evaluation statistics for the main semantic elements

Another explanation could be the lower granularity of elements. Configuration segments in this case correspond to clause segments, Participant roles to the set union of Subjects and Complements while Main verb segments are the same. The merge of Subjects and Complements can be seen in Figure 10.14

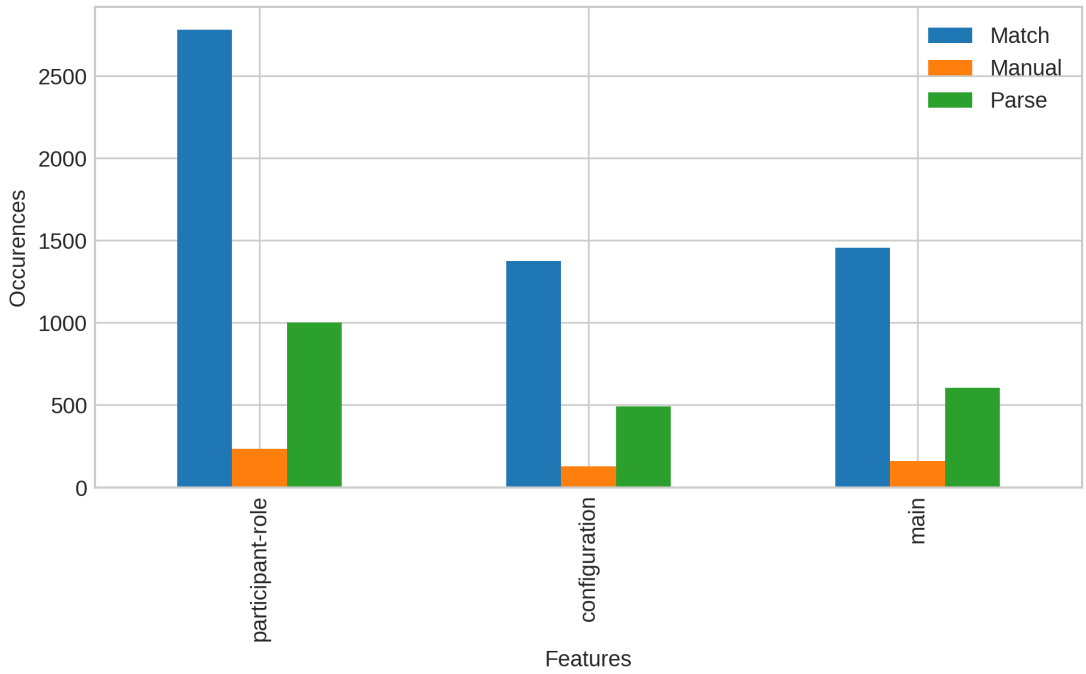


Fig. 10.14 Bar chart of matched and non-matched (manual and parse) segments of the main semantic elements

10.5.6 Semantic evaluation: Transitivity feature selections

Transitivity analysis does not primarily depend on the form of the 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.

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 10.10 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.

10.6 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 11

Conclusions

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

In this process a primary milestone the first steps is the creation of constituency structure. Chapter 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 5. The practical aspects and concrete algorithms are described in Chapter 8 together with the mapping rules used in the process.

To make clear what are the basic ingredients and how the algorithms are cooked, Chapter 7 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 7. Then in Chapters 8 and 9 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 6 and then a concrete implementation described in Chapter 9.

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 10.6 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.

11.1 Practical applications

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

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

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

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

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

11.2 Impact on future research

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

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

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.

11.2.1 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 4.1.1) 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 11.1 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 11.2 Cardiff sample analysis of a clause *embedded* into another

Check the sample analyses in Table 11.1 and 11.2. 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 11.1.

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 11.2.1 (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 11.2.1. 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 11.1 and 11.2 – but also of the modal assessment that includes modality, as demonstrated by the Examples 126 and 127.

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

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

Examples 126 and 127 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*.

11.2.2 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 4.1.

For instance, Nominal class structure proposed in Cardiff grammar (discussed in Section 4.1.3), 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 4.1.4) 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.

11.2.3 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.

11.2.4 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.

11.2.5 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.

11.2.6 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.

11.2.7 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.

11.2.8 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.

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

11.3 A final word

References

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

Allen, James F. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM* 26(11). 832–843. doi:10.1145/182.358434. <http://portal.acm.org/citation.cfm?doid=182.358434>.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Costetchi, Eugeniu. 2013. Semantic role labelling as SFL transitivity analysis. In *Esslli student session 2013 preproceedings*, 29–37.

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. 1973. Generating a sentence in systemic functional grammar. In M.A.K. Halliday & James Martin (eds.), *Readings in systemic linguistics*, Bastford.

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

Fawcett, Robin P. 1987. The semantics of clause and verb for relational processes in English. In Robin P. Fawcett & David J. Young (eds.), *New developments in systemic linguistics: theory and description*, vol. 1, London: Pinter.

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. 1996. A systemic functional approach to complementation in English. In Christopher Butler, Margaret Berry, Robin Fawcett & Guowen Huang (eds.), *Meaning and form: systemic functional interpretations*, Norwood, NJ: Ablex.

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

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

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

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.

Gusfield, Dan & Robert W. Irving. 1989. *The stable marriage problem: Structure and algorithms*. Cambridge, MA, USA: MIT Press.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Iwama, Kazuo & Shuichi Miyazaki. 2008. A survey of the stable marriage problem and its variants. In *International conference on informatics education and research for knowledge-circulating society*, 131–136. IEEE.

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 12^{th} 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 *4^{th} international workshop on linguistically interpreted corpora (linc03)*, <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.15.3277>.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Pradhan, S, E Loper, D Dligach & M Palmer. 2007. Semeval-2007 task-17: English lexical sample, srl and all words. *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)* 87–92. <papers2://publication/uuid/74521E4E-811F-4EB7-A1BE-973D69EBC6C2>.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Zeman, Daniel, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis M. Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinková, Jaroslava Hlavácová, Václava Kettnerová, Zdenka Uresová, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria de Paiva, Kira Droganova, Héctor Martínez Alonso, Çagri Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali El-Kahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonça, Tatiana Lando, Rattima Nitisaroj & Josie Li. 2017. Conll 2017 shared task: multilingual parsing from raw text to universal dependencies. *Proceedings of the CoNLL Shared Task* 1–19.

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

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

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

SFL Syntactic Overview

.1 Cardiff Syntax

Elements found in all groups: Linker (&), Inferer (I), Starter (st), Ender (e)
Units: Sentence (Σ), Clause (Cl), Nominal Group (ngp), Prepositional Group (pgp), Quality Group (qlgp), Quantity Group (qtgp), Genitive Cluster (gencl)

.1.1 Clause

Relative Order of Elements in the Unit Structure:
& |B |L |F |A |C |O |S |O |N |A |I |X |M |Mex |C |A |V |E
Clause May fill: Σ (85%), C (7%), A (4%), Q (2%), f (0.5%), s, qtf, S, m, cv, po
Elements of the Clause: Adjunct (A), Binder (B), Complement (C), Formulaic Element (F), Infinitive Element (I), Let Element (L),Main Verb (M), Main Verb Extension (Mex), Negator (N), Operator (O), Subject (S), Vocative (V), Auxiliary Verb (A), X extension (Xex), Linker (&), Starter (St), Ender(E)

.1.2 Nominal Group

Possible Relative Order of Elements in the Unit Structure:
& |rd |v |pd |v |qd |v |sd |v |od |v |td |v |dd |m |h |q |e
Filling probabilities of the ngp: S (45%), C (32%), cv (15%), A (3%), m (2%), Mex, V, rd, pd, fd, qd, td, q, dt, po
Elements of the ngp: Representational determiner (rd), Selector (v), Partitive Determiner (pd), Fractionative Determiner (fd), Quantifying Determiner (qd), Superlative Determiner (sd), Ordinative Determiner (od), Qualifier-Introducing Determiner (qid), Typic Determiner (td), Deictic Determiner (dd), Modifier (m), Head (h), Qualifier (q)

.1.3 Prepositional Group

Possible Relative Order of Elements in the Unit Structure:
& |pt |p |cv |p |e
Filling Probabilities of the pgp: C (55%), a (30%), q (12%), s (2%) Mex, S, cv, f, qtf
Elements of the pgp: Preposition (p), Prepositional Temperer (pt), Completive (c)

.1.4 Quality Group

Possible Relative Order of Elements in the Unit Structure:
& |qld |qlq |et |dt |at |a |dt |s |f |s |e
Filling probabilities of the qgp: c (38%), m (36%), A (24%), sd (0.5%), Mex, Xex, od, q, dt, at, p, S
Elements of the qlgp: Quality Group Deictic (qld), Quality Group Quantifier (qlq), Emphasizing Temperer (et), Degree Temperer (dt), Adjunctival Temperer (at), Apex (a), Scope (s), Finisher (f)

.1.5 Quantity Group

Possible Relative Order of Elements in the Unit Structure:
ad |am |qtf |e **Filling probabilities of the qtgp:** qd (85%), A (8%), dt (6%), B, p, ad, fd, sd **Elements of the qtgp** Adjustor (ad), Amount (am), Quantity Finisher (qf)

.1.6 Genitive Cluster

Possible Relative Order of Elements in the Unit Structure:
& |po |g |o |e
Filling probabilities of the gencl: dd (99%), h, m, qld
Elements of the gencl: Possessor (po), Genitive Element (g), Own Element (o)

.2 Sydney Syntax

.2.1 Logical

Possible Relative Order of Elements in the Unit Structure:
Pre-Modifier |Head |Post-Modifier

.2.2 Textual

Possible Relative Order of Elements in the Clause Structure:
Theme |Rheme
New |Given |New

.2.3 Interactional

Possible Relative Order of Elements in the Clause Structure:
Residue |Mood |Residue |Mood tag
Adjunct |Complement |Finite |Subject |Finite |Adjunct |Predicator |Complement| Adjunct

.2.4 Experiential

Possible Relative Order of Elements in the Clause Structure:
Circumstance |Participant |Circumstance |Process| Participant |Circumstance
Possible Relative Order of Elements in the Nominal Group Structure:
Deictic |Numerative |Epithet | Classifier| Thing |Qualifier
Possible Relative Order of Elements in the Verbal Group Structure:
Finite |Marker |Auxiliary |Event
Possible Relative Order of Elements in the Adverbial and Preposition Group Structure: Modifier |Head |Post-Modifier
Possible Relative Order of Elements in the Prepositional Phrase Structure:
Predicator |Complement
Process |Range

.2.5 Taxis

Possible Relative Order of Elements in the Parataxis Structure:
Initiating |Continuing
Possible Relative Order of Elements in the Hypoataxis Structure:
Dependent |Dominant |Dependent

Stanford Dependency schema

The Stanford dependency relations as defined in Stanford typed dependencies manual (Marneffe & Manning 2008a)

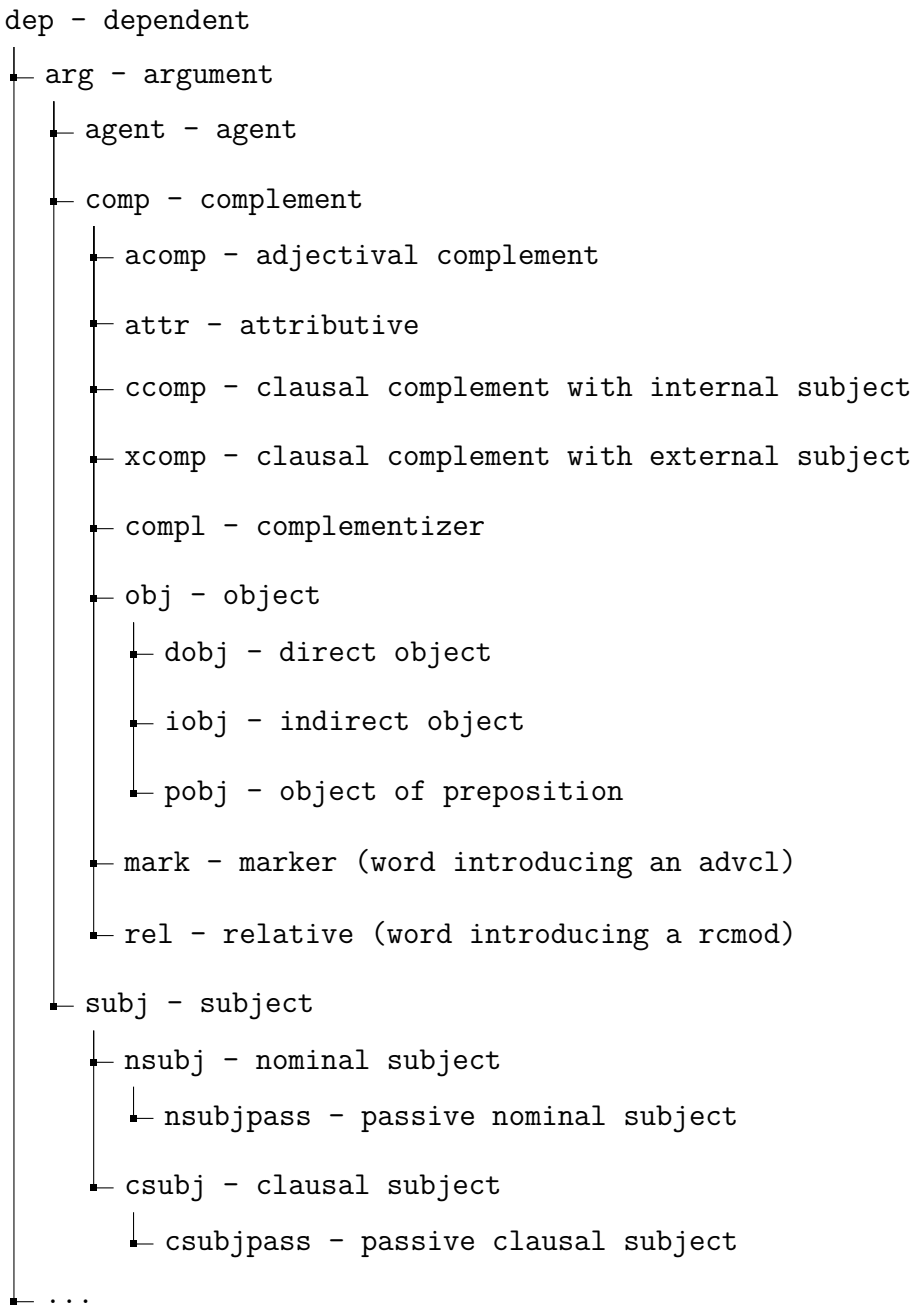


Fig. 1 The Stanford dependency scheme - part one

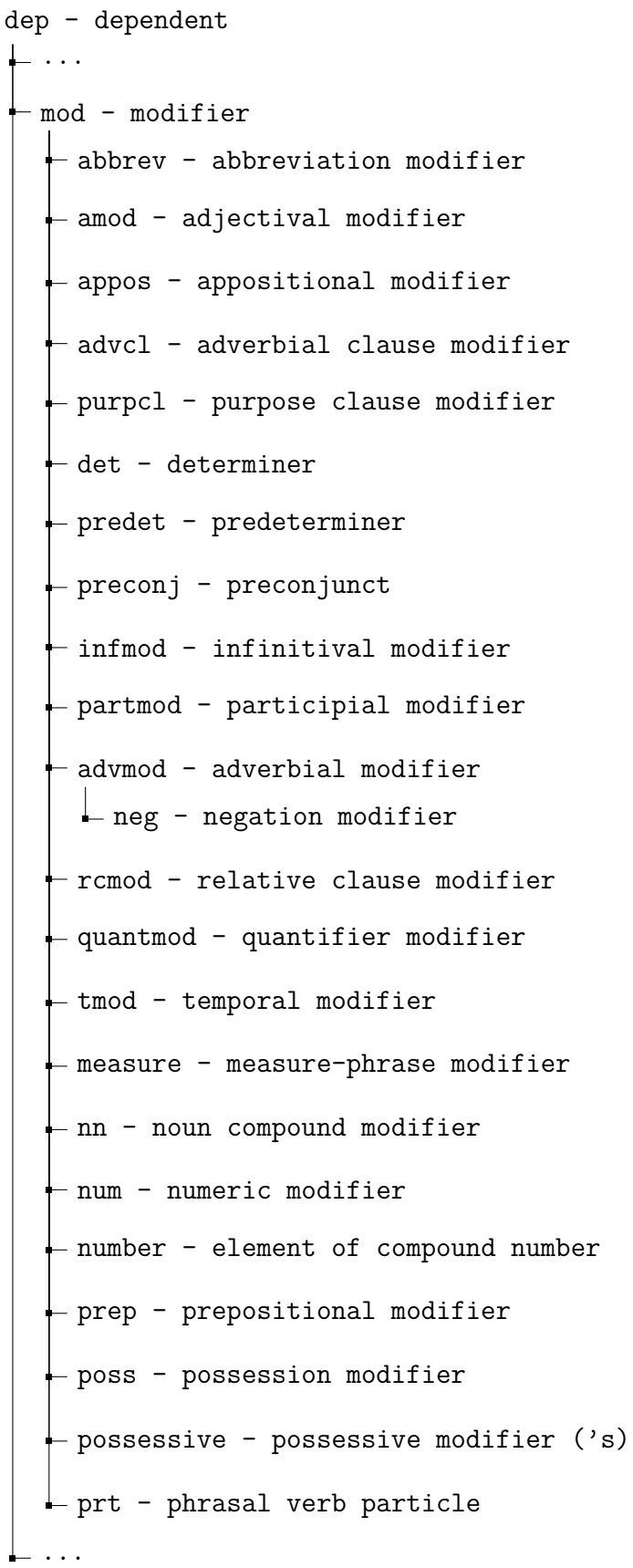


Fig. 2 The Stanford dependency scheme - part two

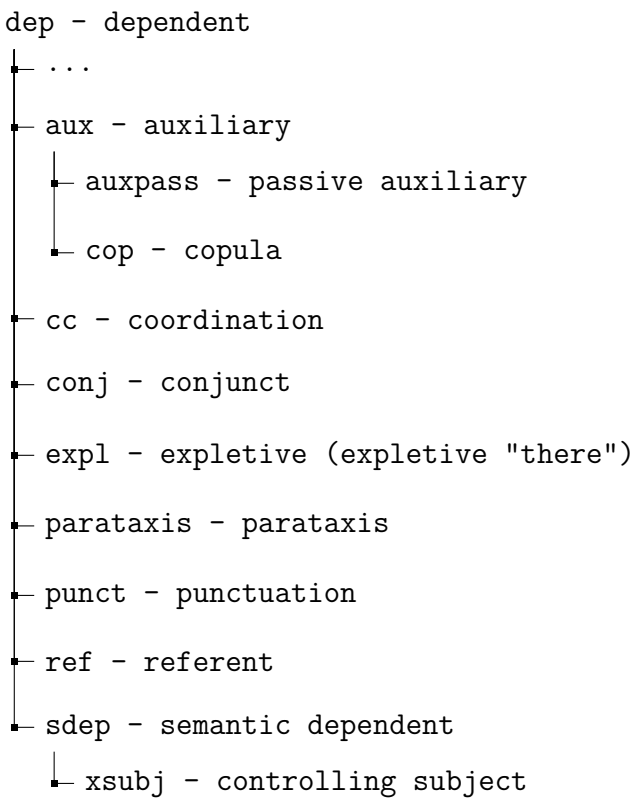


Fig. 3 The Stanford dependency scheme - part three

Penn treebank tag-set

Tag	**Description**	**Example**
CC	conjunction, coordinating	and, or, but
CD	cardinal number	five, three, 13%
DT	determiner	the, a, these
EX	existential there	there were six boys
FW	foreign word	mais
IN	conjunction, subordinating or preposition	of, on, before, unless
JJ	adjective	nice, easy
JJR	adjective, comparative	nicer, easier
JJS	adjective, superlative	nicest, easiest
LS	list item marker	
MD	verb, modal auxillary	may, should
NN	noun, singular or mass	tiger, chair, laughter
NNS	noun, plural	tigers, chairs, insects
NNP	noun, proper singular	Germany, God, Alice
NNPS	noun, proper plural	we met two Christmases ago
PDT	predeterminer	both his children
POS	possessive ending	's
PRP	pronoun, personal	me, you, it
PRP$	pronoun, possessive	my, your, our
RB	adverb	extremely, loudly, hard
RBR	adverb, comparative	better
RBS	adverb, superlative	best
RP	adverb, particle	about, off, up
SYM	symbol	%
TO	infinitival to	what to do?
UH	interjection	oh, oops, gosh
VB	verb, base form	think
VBZ	verb, 3rd person singular present	she thinks
VBP	verb, non-3rd person singular present	I think
VBD	verb, past tense	they thought
VBN	verb, past participle	a sunken ship
VBG	verb, gerund or present participle	thinking is fun
WDT	wh-determiner	which, whatever, whichever
WP	wh-pronoun, personal	what, who, whom
WP$	wh-pronoun, possessive	whose, whosever
WRB	wh-adverb	where, when
.	punctuation mark, sentence closer	.;?*
,	punctuation mark, comma	,
:	punctuation mark, colon	:
(	contextual separator, left paren	(
)	contextual separator, right paren	)

Table 3 Penn Treebank tag set

Mapping dependency to constituency graph

key	***operation***	***parameter***
acomp	new constituent	COMPLEMENT
advcl	new constituent	ADJUNCT
advmod	extend current	None
amod	new constituent	EPITHET_CLASSIFIER_OR_ORDINAL
agent	new constituent	COMPLEMENT_AGENT
appos	new constituent	APPOSITION
aux	extend current	None
auxpass	extend current	None
complm	new constituent	MARKER
conj	extend current	None
csubj	new constituent	SUBJECT
csubjpass	new constituent	SUBJECT
det	new constituent	DEICTIC
dobj	new constituent	COMPLEMENT
expl	new constituent	EXPLETIVE_MARKER
infmod	new constituent	QUALIFIER
iobj	new constituent	COMPLEMENT_DATIVE
mark	new constituent	MARKER
mwe	extend current	None
neg	extend current	None
nn	extend current	None
npadvmod	new constituent	ADJUNCT
nsubj	new constituent	SUBJECT
nsubjpass	new constituent	SUBJECT
num	new constituent	CARDINAL_NUMERATIVE
number	extend current	None
parataxis	new constituent	CLAUSE
partmod	new constituent	QUALIFIER
vmod	new constituent	QUALIFIER
pobj	extend current	None
poss	new constituent	POSESSOR
possessive	new constituent	POSESSOR
preconj	extend current	None
predet	new constituent	PREDEICTIC
prepc	new constituent	COMPLEMENT_ADJUNCT
prt	new constituent	MARKER
punct	extend current	None
purpcl	new constituent	CLAUSE
quantmod	extend current	None
rcmod	new constituent	QUALIFIER
ref	extend current	None
rel	new constituent	CLAUSE
tmod	new constituent	ADJUNCT
xcomp	new constituent	COMPLEMENT

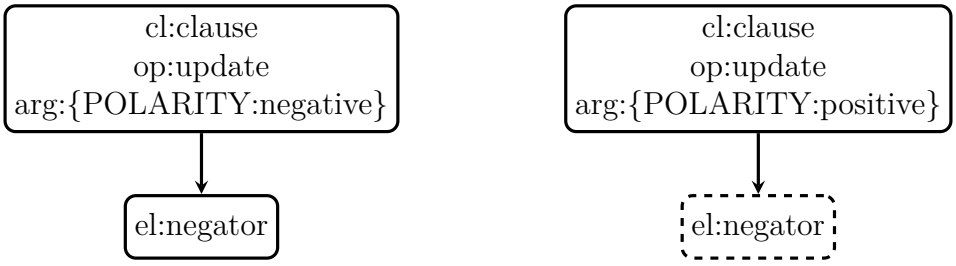
Key	***Operation***	***Value***
JJ-dep-IN	new constituent	MARKER
VB-dep-IN	new constituent	MARKER
VB-dep-VB	new constituent	CLAUSE
NN-dep-NN	extend current	None
NN-dep-VB	new constituent	CLAUSE
VB-dep-WP	new constituent	COMPLEMENT_ADJUNCT
VB-dep-NN	new constituent	ADJUNCT
RB-dep-IN	extend current	None
WR-dep-JJ	extend current	None
VB-dep-JJ	new constituent	ADJUNCT
VB-conj-VB	new constituent	CLAUSE
VB-cc-CC	new constituent	MARKER
NN-cc-CC	extend current	None
VB-prep-NN	new constituent	COMPLEMENT_ADJUNCT
VB-prep-JJ	new constituent	COMPLEMENT_ADJUNCT
VB-prep-PR	new constituent	COMPLEMENT_ADJUNCT
VB-prep-WP	new constituent	COMPLEMENT_ADJUNCT
VB-prep-CD	new constituent	COMPLEMENT_ADJUNCT
NN-prep-NN	new constituent	QUALIFIER
NN-prep-PR	new constituent	QUALIFIER
RB-npadvmod-NN	extend current	None
NN-npadvmod-NN	extend current	None
VB-npadvmod-NN	new constituent	ADJUNCT
JJ-npadvmod-RB	extend current	None
VB-advmod-RB	new constituent	ADJUNCT
VB-advmod-JJ	new constituent	ADJUNCT
VB-advmod-WR	new constituent	COMPLEMENT
NN-advmod-RB	new constituent	PREDEICTIC
VB-ccomp-NN	new constituent	COMPLEMENT
VB-ccomp-VB	new constituent	COMPLEMENT
IN-pcomp-IN	new constituent	COMPLEMENT_ADJUNCT
IN-pcomp-NN	new constituent	COMPLEMENT_ADJUNCT
IN-pcomp-CD	new constituent	COMPLEMENT_ADJUNCT
IN-pcomp-JJ	new constituent	COMPLEMENT_ADJUNCT
NN-amod-CD	new constituent	CARDINAL_NUMERATIVE
NN-infmod-VB	new constituent	QUALIFIER
CD-prep-NN	new constituent	QUALIFIER
NN-vmod-VB	new constituent	QUALIFIER
NN-prep-JJ	new constituent	QUALIFIER
DT-prep-NN	new constituent	QUALIFIER
JJ-prep-NN	extend current	None

Normalization of PTDB and Cardiff TRANSITIVITY system

major	*minor*	*new index*	*min # roles*	*max # roles*
action	one role	one-role-action	1	1
	two role	two-role-action	2	2
	three role	three-role-action	3	3
relational	attributive	attributive	2	3
	locational	locational	2	3
	directional	directional	2	5
	possessive	possessive	2	3
	matching	matching	2	3
emotion	desiderative	desiderative	2	2
	plux xxx	emotive	2	3
perception	xxx	two-role-perception	2	2
	3 p Ag	three-role-perception	3	3
cognition	xxx	two-role-cognition	2	2
	3 p Ag/ matchee	three-role-cognition	3	3
environmental		environmental	x	x
influential	starting	starting	1	2
	continuing	continuing	1	2
	ceasing	ceasing	1	2
	succeeding	succeeding	1	2
	failing	failing	1	2
	causative	causative	1	2
	permissive	permissive	1	2
	enabling	enabling	1	2
	preventing	preventing	1	2
	delaying	delaying	1	2
	tentative	tentative	1	2

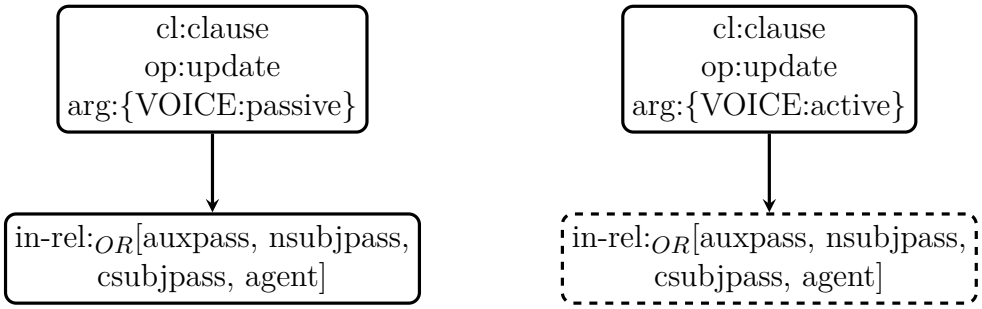
The process type column in PTDB contains two words separated by comma. The first one I call *major* as it represents a high level selection in the TRANSITIVITY system and the second one I call *minor* hints at selecting a particular participant configuration. The re-indexing consists of replacing the two features with a more delicate selection in the TRANSITIVITY system.

A selection of graph patterns



(a) Negative polarity (b) Negative polarity

Fig. 4 Polarity detection graph patterns



(a) Passive voice (b) Active voice

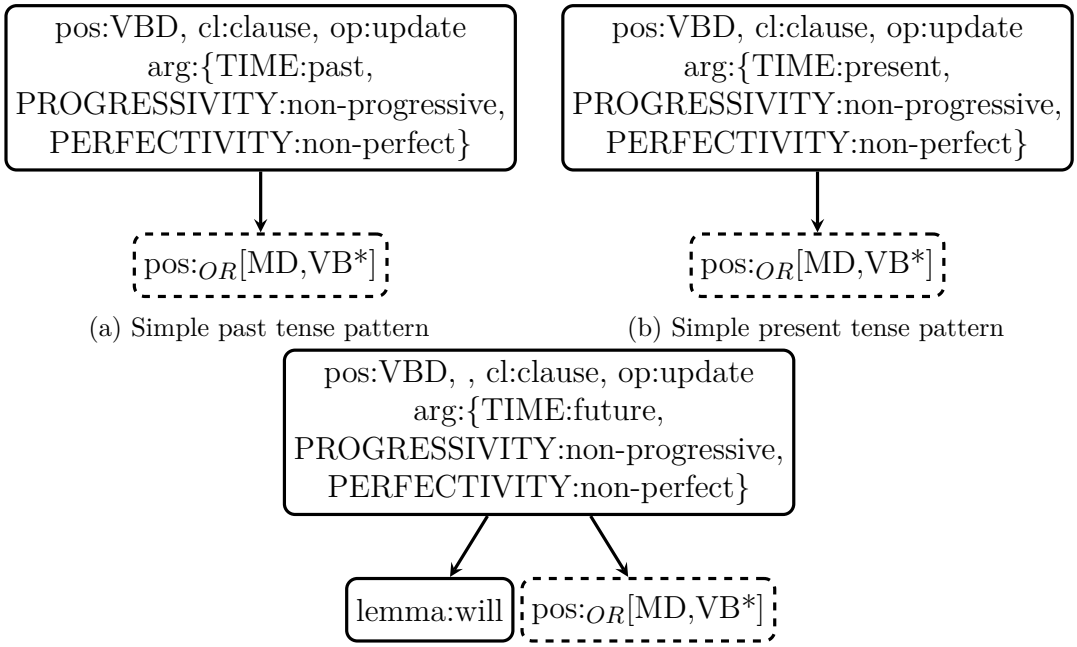
Fig. 5 Voice detection graph patterns

Tense simple

Tense progressive

Tense perfect

Tense perfect

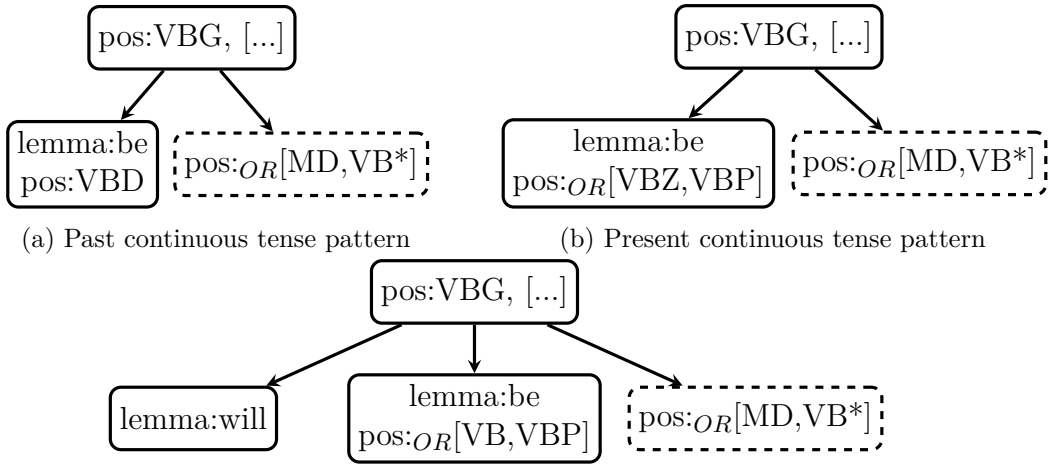


(a) Simple past tense pattern

(b) Simple present tense pattern

(c) Simple future tense pattern

Fig. 6 Simple past, present and future tense patterns

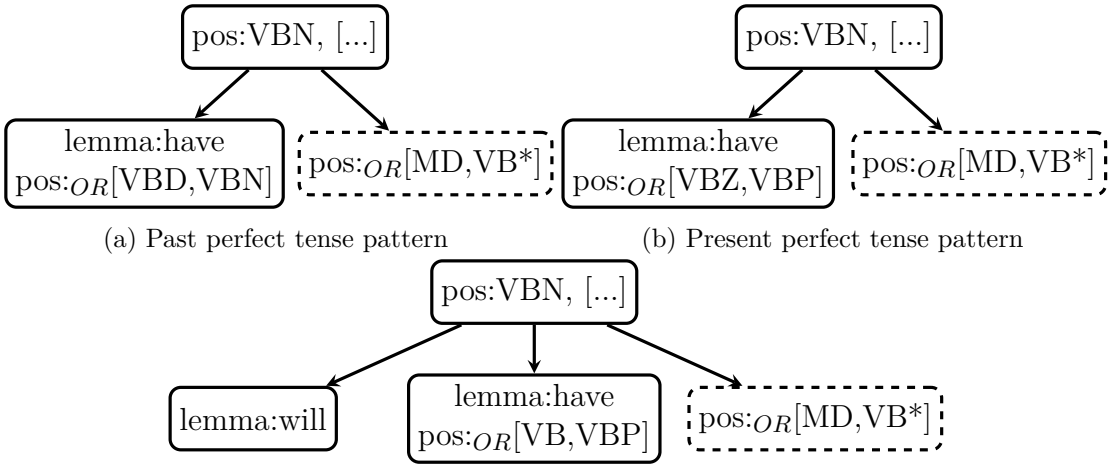


(a) Past continuous tense pattern

(b) Present continuous tense pattern

(c) Future continuous tense pattern

Fig. 7 Past, present and future continuous tense patterns

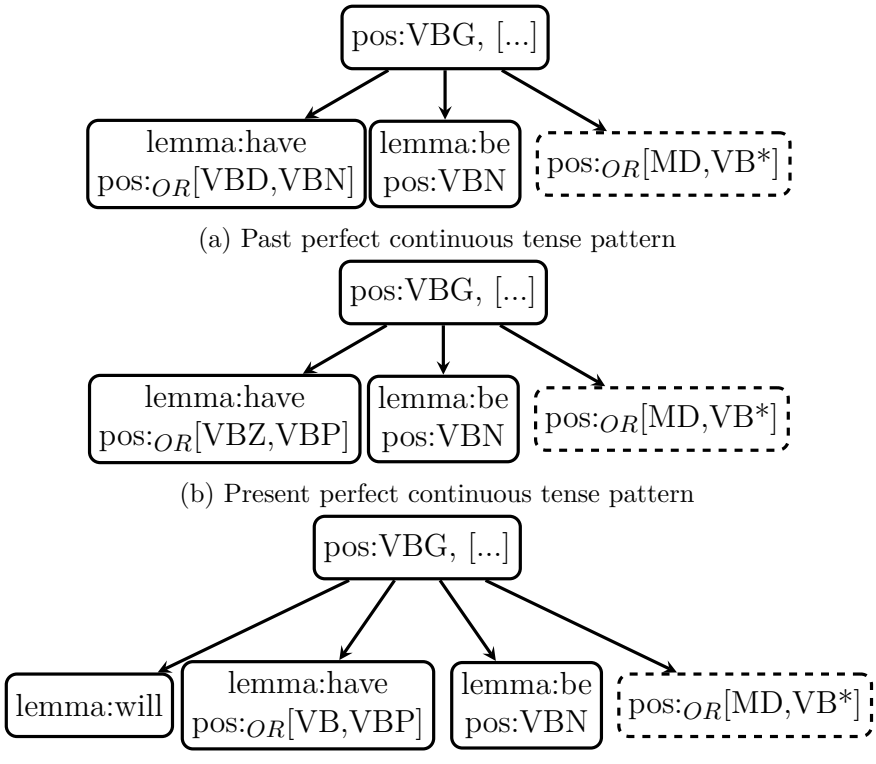


(a) Past perfect tense pattern

(b) Present perfect tense pattern

(c) Future perfect tense pattern

Fig. 8 Past, present and future perfect tense patterns



(a) Past perfect continuous tense pattern

(b) Present perfect continuous tense pattern

(c) Future perfect continuous pattern

Fig. 9 Past, present and future perfect continuous tense patterns

Algorithms with lesser importance

The Algorithm 19 and 20 below are part of the Algorithm 12 for creating the Wh null elements.

Algorithm 19: 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 20: 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     for config in higher clause configurations:
 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
       /* check if the lower clauses might miss an argument       */
 7     for clause in lower clauses:
 8         for config in clause configurations:
 9             if number of clause theta constituents < number of config arguments:
10                 lower is eligible ← True
11                 break
12     if higher is eligible and lower is eligible:
13         if higher clause has "that" complementizer:
14             create Object Wh-trace in the lowest clause
15         else:
16             if Wh-group has case:
17                 if Wh-group case is nominative(subjective):
18                     create Subject Wh-trace in lowest clause
19                 else:
20                     create Object Wh-trace in lowest clause
21             else:
22                 create Wh-trace with Subject function and attempt to assign
                    theta roles
23                 if theta roles not successfully assigned in lower clause:
24                     change the Wh-trace to Object function and assign theta roles
25 end
```