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

Table of contents

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.

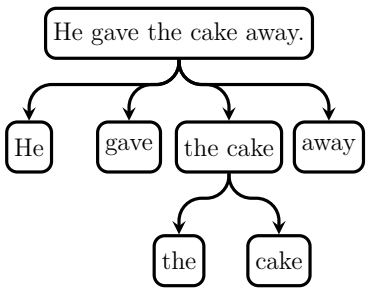

Fig. 1.1 Constituency diagram for Example 1

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.

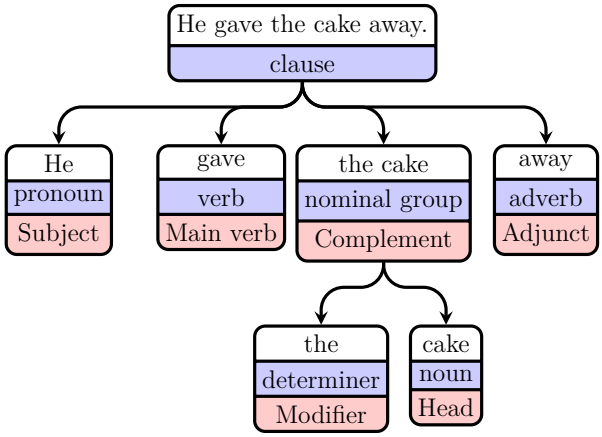

Fig. 1.2 Constituency analysis of Example 1 with unit classes and grammatical functions

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.

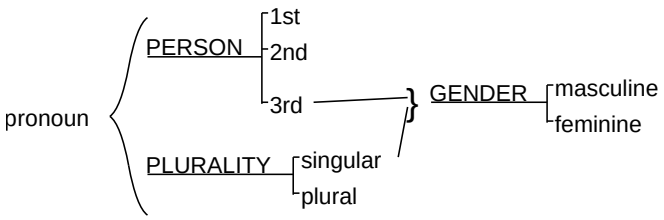

Fig. 1.3 The systematisation of three pronominal features in traditional grammar

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.

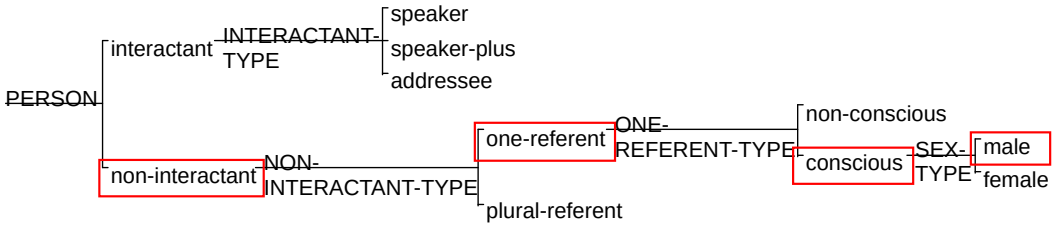

Fig. 1.4 The selections in Person system network from Halliday & Matthiessen (2013: 366) for pronoun "He"

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.

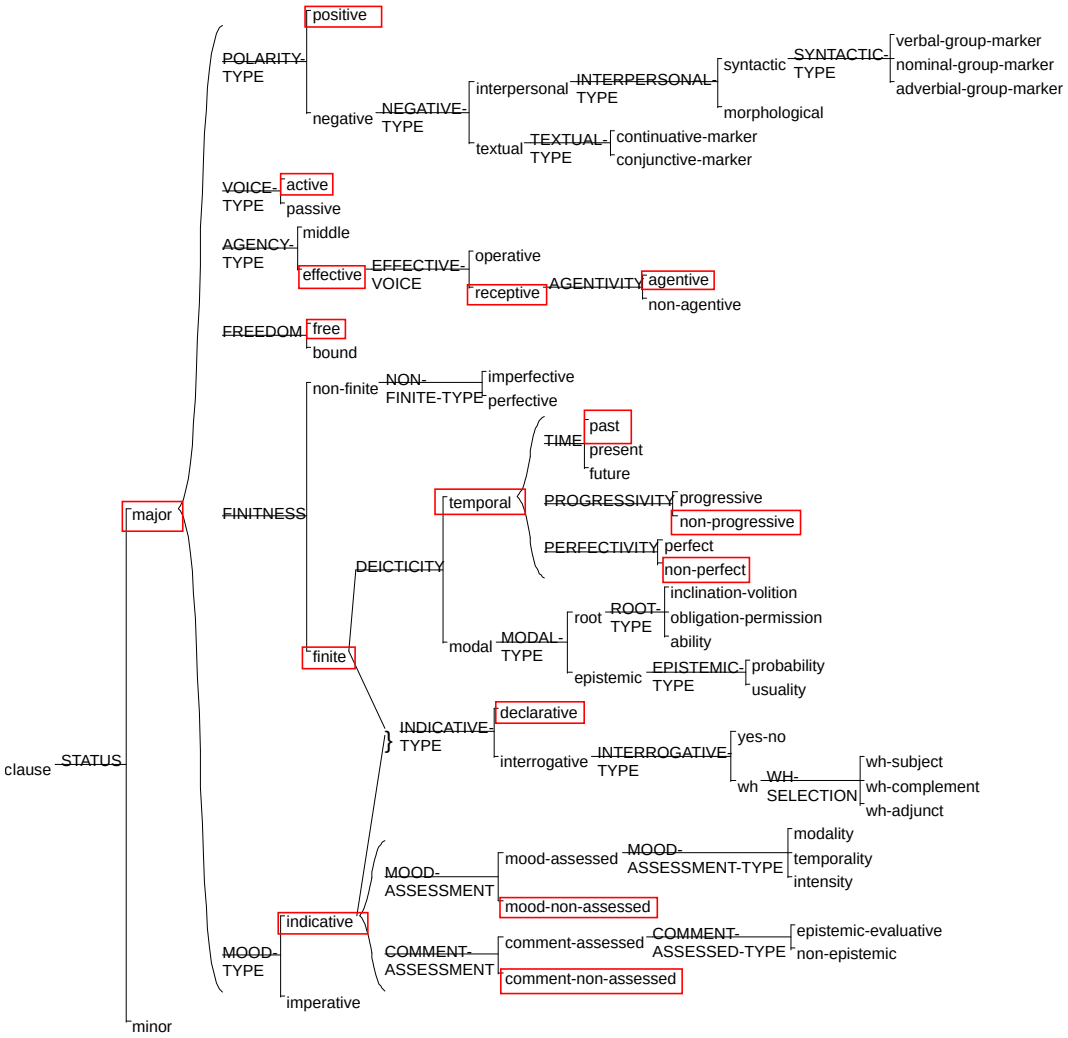

Fig. 1.5 The feature selections in the Mood system network for clause constituent in Example 1

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.

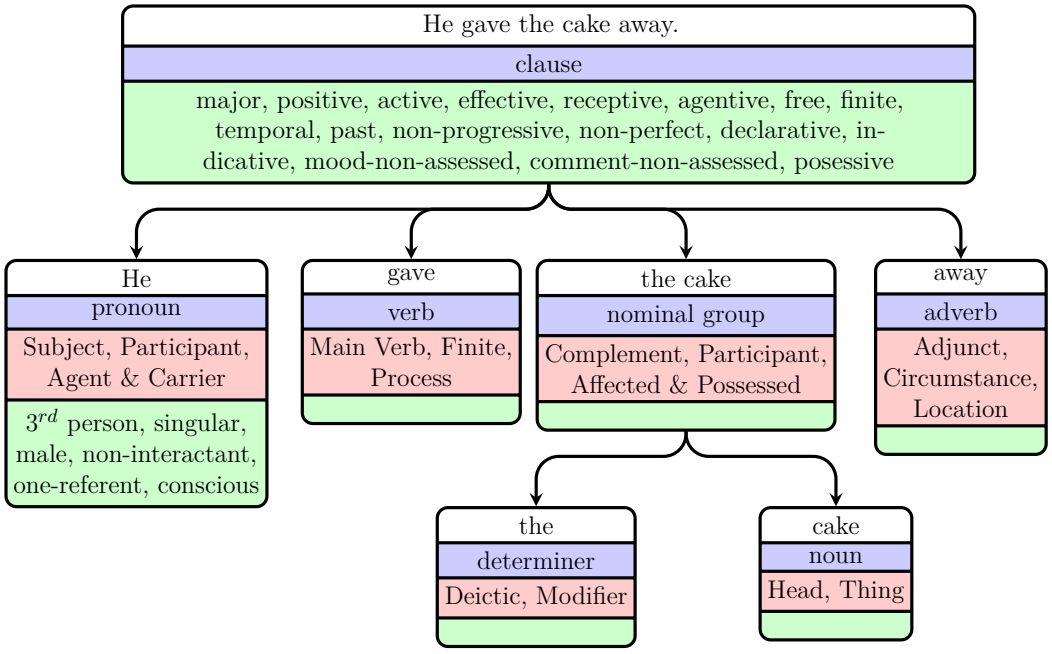

Fig. 1.6 Representation of Example 1 as feature rich constituency tree

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.

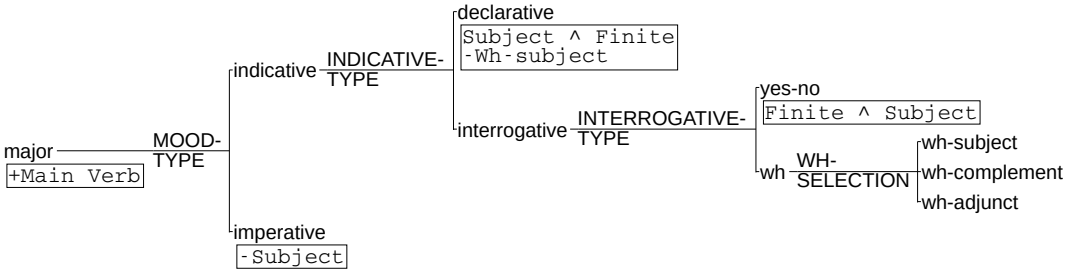

Fig. 1.7 A fragment of mood system from Halliday & Matthiessen (2013: 366)

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).

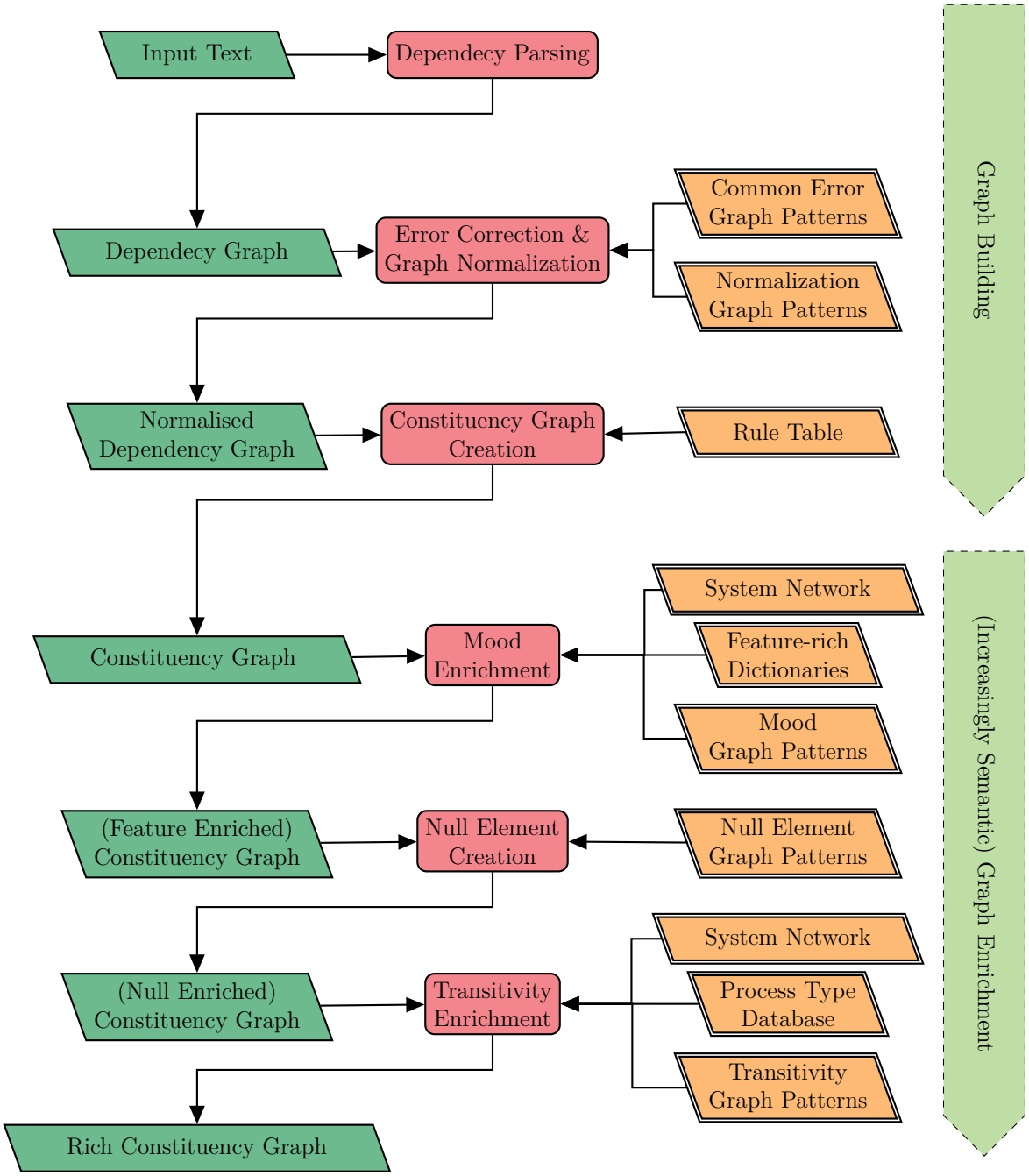

Fig. 1.8 The parsing process pipeline

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.