

# Chapter 1

## Introduction

### 1.1 On Artificial Intelligence (AI) and Computational Linguistics

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

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

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

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

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

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

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

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

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

## 1.2 Living in technologically ubiquitous world

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

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

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

### 1.3 NLP for business

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

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

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

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

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

## 1.4 The linguistic framework

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

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

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

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

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

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

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

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

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

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

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

## 1.5 An example of Systemic Functional analysis

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

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

- (1) He gave the cake away.



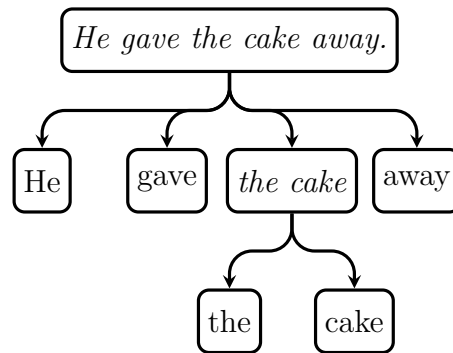


Fig. 1.1 Representation of the Example 1 as constituency tree

Figure 1.1 depicts the constituency division of the clause. 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. Table 1.1 provides a constituency analysis in SFL tradition. Here the sentence is formed of a single clause which has four constituting functional parts: a Subject, a Main Verb (also known as Predicate), a Complement and an Adjunct. Each of these functional parts is filled correspondingly by a pronoun, a verb, a nominal group and an adverb as assigned in the table below.

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

Table 1.1 Constituency analysis with unit classes and grammatical functions

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



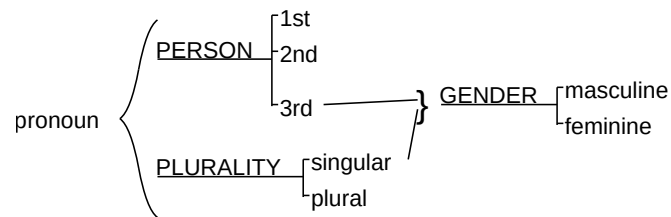


Fig. 1.2 The systematisation of three pronominal features in traditional grammar

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

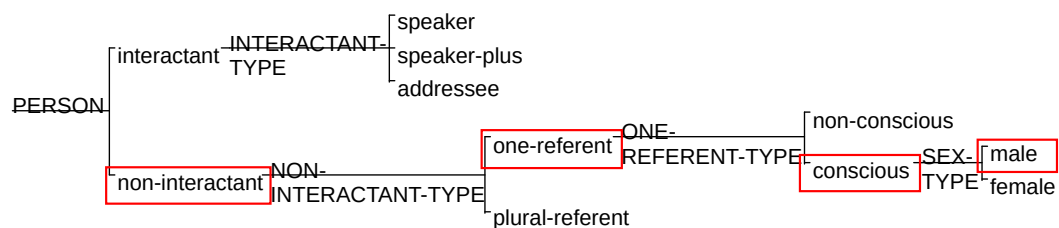


Fig. 1.3 The selections in Person system network from Halliday & Matthiessen (2013: 366)

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

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

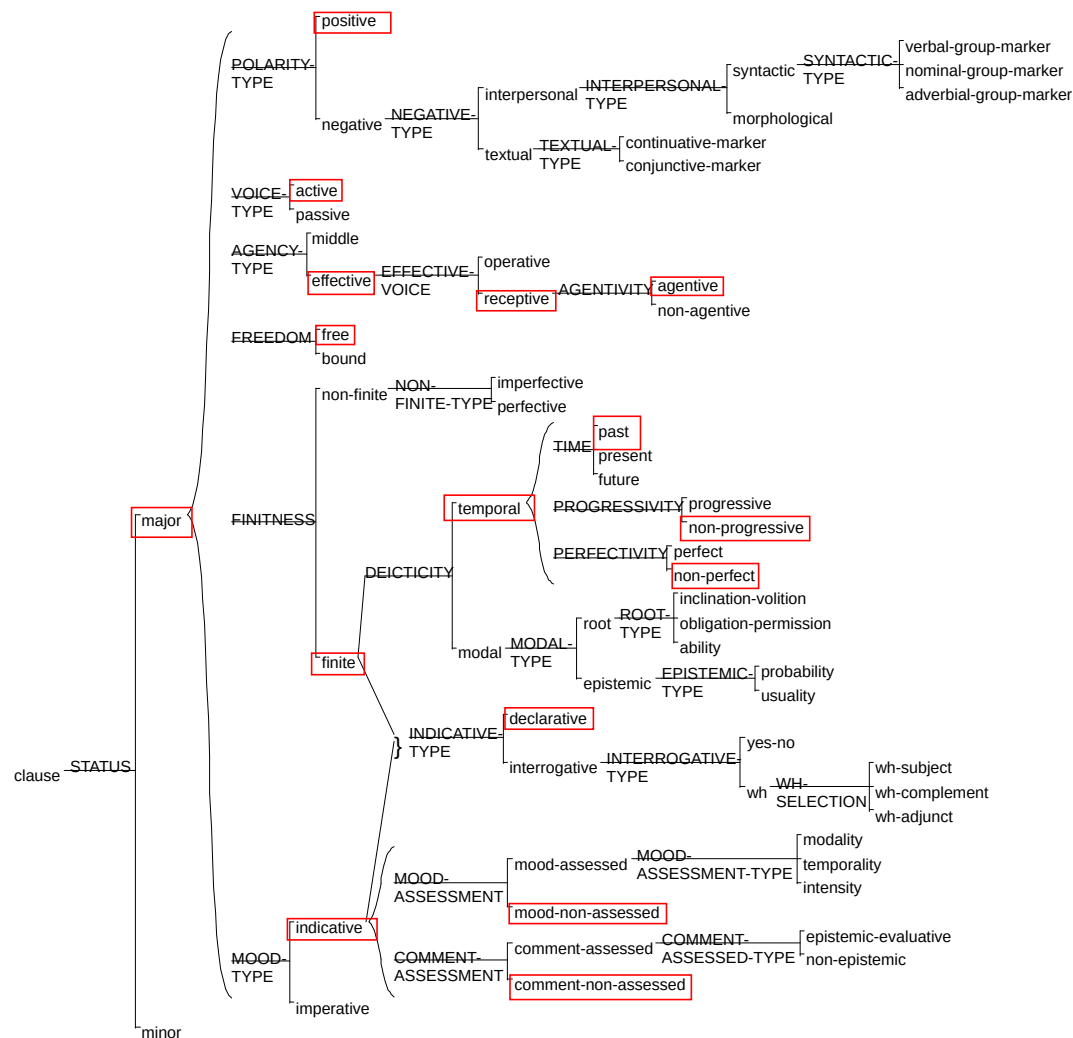


Fig. 1.4 The feature selections in the Mood system network for Example 1

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

- (2) [*Agent–Carrier* He] gave [*Possessed* the cake] away.

There are more functions and features that can be assigned to the constituents in the Example 1 but I stop here. The analysis provided so far highlights that SFG grammar has a variety of functions serving to express different meanings. The traditional grammar distinguishes them as syntactic and semantic functions but, as we will see in Chapter 3 below, SFL does not make such a distinction. Another aim of the current

section was to provide a glance of the feature rich grammar and I hope the example with Mood feature selection in Figure 1.4 fulfils this goal.

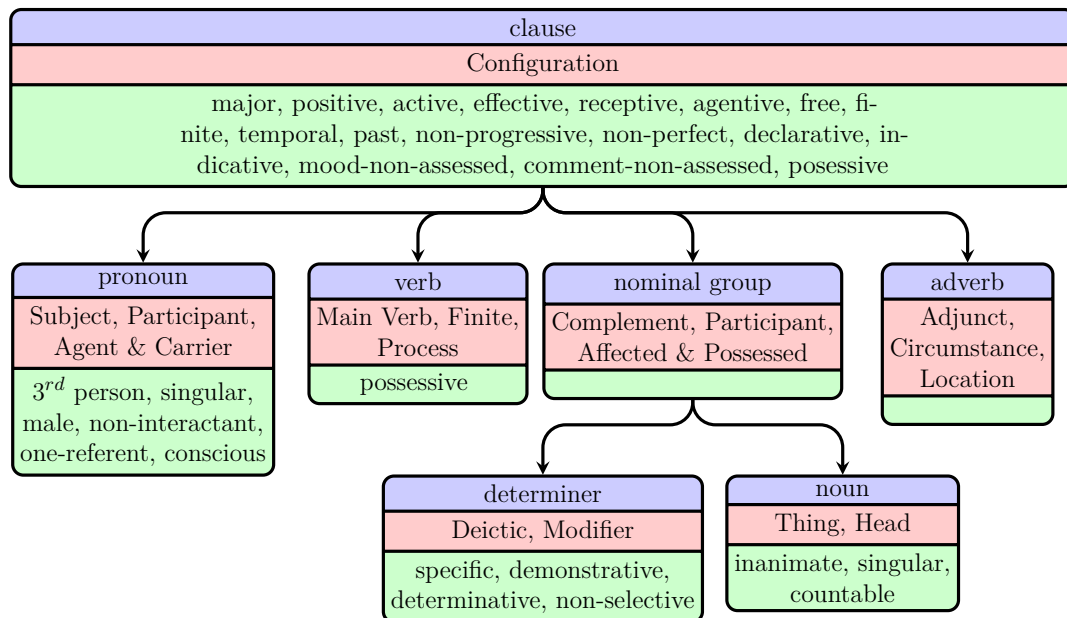


Fig. 1.5 Representation of Example 1 as feature rich constituency graph

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

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

## 1.6 The problem of parsing with SFGs

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

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

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

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

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

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

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

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

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

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

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

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

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

Another difficulty in parsing with SFGs ~~lays~~ in the fact that, as the analysis moves away from directly observable grammatical variations towards more abstract

semantic variations, the difficulty of generating an accurate account increases drastically. Transitivity system network for example consists of such semantic features and it is comparable to what is called in computational linguistics (shallow) *semantic parsing* or *Semantic Role Labelling* (SRL) (Carreras & Màrquez 2005).

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

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

- (3) [*Agent–Perceiver* Jacqueline] glanced [*Phenomenon* at her new watch].

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

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

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

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



<i>She</i>	<i>seemed</i>	<i>to</i>	<i>worry</i>	<i>about</i>	<i>missing</i>	<i>the</i>	<i>river</i>	<i>boat.</i>
clause								
Subject	Main Verb	Complement						
		clause						
		Infinitive Element	Main Verb	Complement				
		clause						
			Binder	Main Verb	Complement			

Table 1.3 SF constituency analysis in Cardiff grammar style

corresponds to the Subject constituent and the second to the Complement constituent. This way the verb meaning *seem*<sub>1</sub> corresponds to an Attributive configuration that distributes Carrier and Attribute roles to the Subject “She” and the Complement “to worry about missing the river boat”. In the case of *worry about*<sub>1</sub> and *miss*<sub>1</sub> 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*<sub>1</sub> 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*<sub>1</sub> that assigns an Affected & Carrier role to the empty Subject and the Possessed role to the Complement “the river boat”.

<i>verb meaning</i>	<i>semantic configuration</i>	<i>participant role mandatory distribution</i>
<i>seem</i> <sub>1</sub>	Attributive	Carrier + Attribute
<i>worry about</i> <sub>1</sub>	Two Role Cognition	Agent & Cognizant + Phenomenon
<i>miss</i> <sub>1</sub>	Possessive	Affected & Carrier + Possessed (thing)

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

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

- (5) *She* worried about missing the river boat.
- (6) *She* missed the river boat.
- (7) *She*<sub>*i*</sub> seemed [*null-Subject*<sub>*i*</sub> to worry [about *null-Subject*<sub>*i*</sub> missing the river boat]].

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

<i>She<sub>i</sub></i>	<i>seemed</i>	<i>null<sub>i</sub></i>	<i>to</i>	<i>worry</i>	<i>about</i>	<i>null<sub>i</sub></i>	<i>missing</i>	<i>the</i>	<i>river</i>	<i>boat.</i>
Attributive configuration										
Agent	Attribute									
Two Role Cognition configuration										
Agent & Cognizant				Phenomenon						
Possessive configuration										
						Affected & Carrier		Possessed		

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

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

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

Considering the problems above how could the vast search space of grammars such as Nigel be restricted to a reasonable size and how can be compensated the

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

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

## 1.7 On theoretical compatibility and reuse

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

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

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

the accomplishments in those domains I attempt at reusing in this thesis as motivated below.

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

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

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

## 1.8 Thesis Goals and Proposed Solution

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

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

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

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

### 1.8.1 Towards the syntagmatic account

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

This challenge can be addressed by filling the gap of the syntagmatic account within the SFL grammar directly. It involves, first, providing information about which grammatical functions operate at each rank, second, which grammatical functions can be filled by which classes of units and, third, providing relative and absolute description

of the element order for each unit class. This information in the grammar can guide the process of building the constituency structure.

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

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

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

### 1.8.2 Towards the paradigmatic account

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

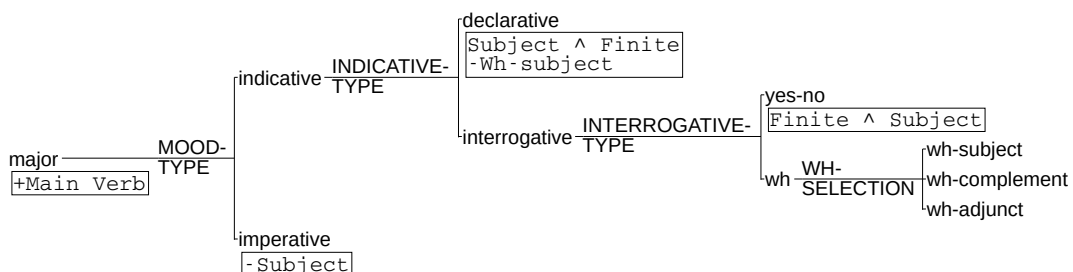


Fig. 1.6 A fragment of mood system from [Halliday & Matthiessen \(2013: 366\)](#)

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

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

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

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

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

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



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

### 1.8.3 The Implementation Architecture

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

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

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

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

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

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

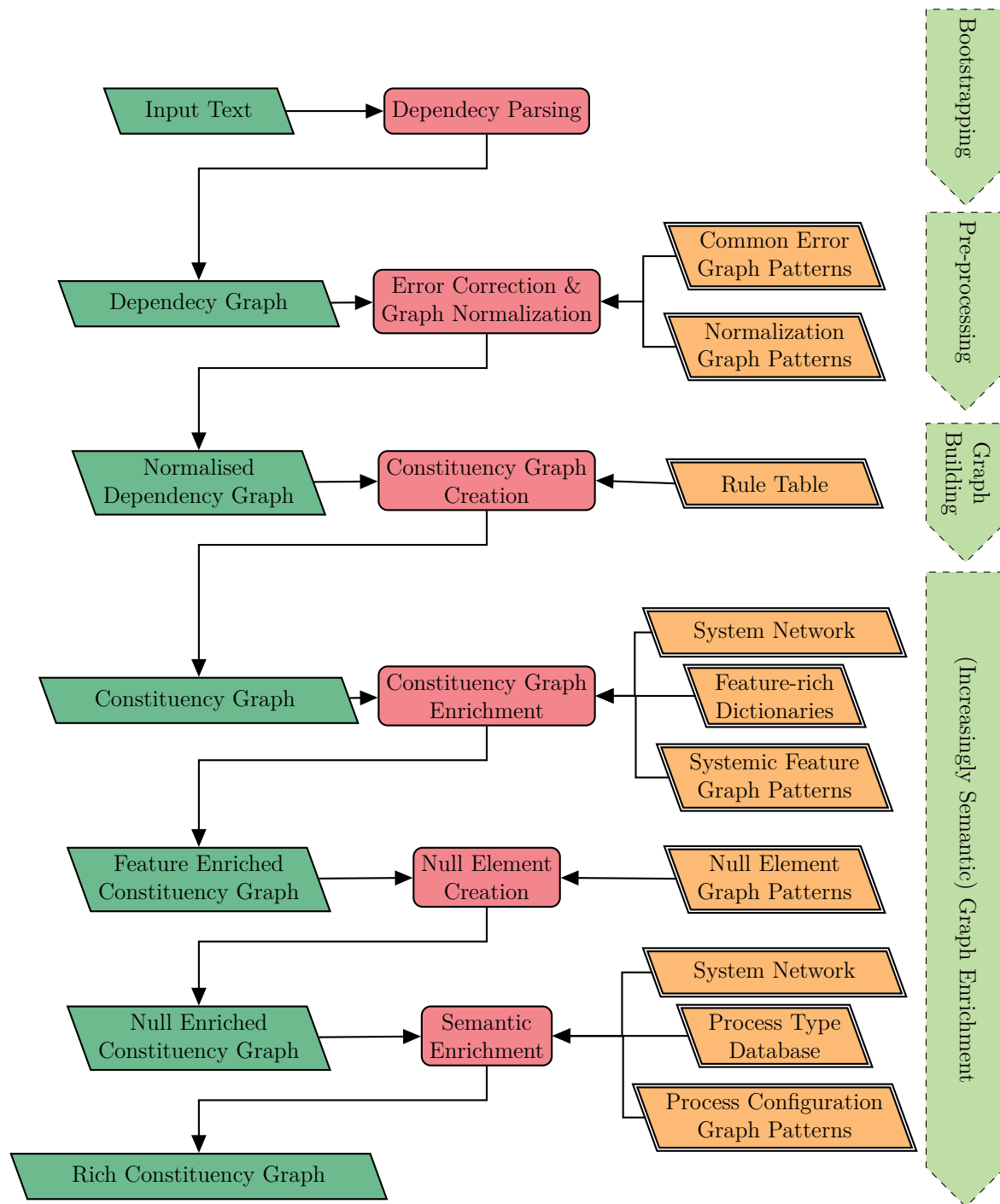


Fig. 1.7 The parsing process pipeline

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

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


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

#### 1.8.4 Research questions and contributions


This thesis addresses the following questions:

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

Also it brings the following theoretical and practical contributions:

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

## 1.9 Provisional Thesis Structure

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