Constituentless Compositionality: A Compositional Account of Dependency Grammar

MSc Thesis *(Afstudeerscriptie)*

written by

Ryan Nefdt
Master of Logic
University of Amsterdam
(born 8th September, 1987 in Cape Town, South Africa)

under the supervision of **Dr Henk Zeevat**, and submitted to the Board of Examiners in partial fulfillment of the requirements for the degree of

MSc in Logic

at the *Universiteit van Amsterdam.*

Date of the public defense:	**Members of the Thesis Committee:**
August 21, 2013	Dr Jakub Szymanik
	Dr Michael Franke
	Dr Theo Janssen
	Dr Henk Zeevat

Contents

I Compositionality and Natural Language **4**

1.1 Introduction . . . 5

1.2 Disambiguating Compositionality . . . 6

1.3 (Bad) Reasons for compositionality . . . 8

1.3.1 Productivity . . . 8

1.3.2 Systematicity . . . 10

1.3.3 Novelty and Learnability . . . 13

1.3.4 Infinity . . . 14

1.4 Conclusion . . . 20

II Compositionality and Formal Language **22**

2.1 Introduction . . . 23

2.2 Formal Language Theory . . . 24

2.2.1 Regular Languages (Type 3) . . . 25

2.2.2 Context-free Languages (Type 2) . . . 26

2.2.3 Context-Sensitive Languages (Type 1) and Unrestricted Languages (Type 0) . . . 28

2.2.4 Beyond the Chomsky Hierarchy: Mildly context-sensitive Languages . 28

2.2.5 A Computational Caveat . . . 29

2.3 (Good) Reasons for Compositionality . . . 30

2.3.1 Computational Complexity . . . 30

2.3.2 Infinity as a design choice . . . 31

2.3.3 Subconclusion . . . 32

2.4 Formal Definition of Compositionality . . . 33

2.4.1 Formal Preliminaries . . . 33

2.4.2 Standard Compositionality . . . 34

2.4.3 Strong Compositionality . . . 36

2.4.4 Direct Compositionality . . . 37

2.4.5 A Formal Description of Constituency . . . 39

2.4.6 The Principle of Syntactic Integrity . . . 42

2.5 Conclusion . . . 44

III A Compositional Semantics for Dependency Grammar 45
3.1 Dependency Grammar . . . 46
3.1.1 Elementary Structures and Constituency . . . 46
3.1.2 Lexicalization and Dependency Grammar . . . 48
3.1.3 Dependency Structures . . . 50
3.1.4 Projective and Non-projective structures . . . 51
3.1.5 Why Dependency Grammar? . . . 52
3.2 Compositionality applied . . . 53
3.2.1 Dependency Algebra . . . 53
3.3 Challenges for a Type Semantics of Dependency Grammar . . . 56
3.3.1 Extensional Montague Grammar . . . 56
3.3.2 Scope Ambiguity . . . 58
3.3.3 Problems for Type-theoretic Dependency Semantics . . . 59
3.4 A Resolved Montague Grammar for Dependency Structures . . . 62
3.4.1 F-Structures in Lexical-Functional Grammar . . . 62
3.4.2 The Semantics . . . 63
3.4.3 Montague Semantics for Dependency Grammar . . . 65
3.4.4 *DS*s as Semantic Structures . . . 66
3.4.5 Translation Procedure . . . 70
3.4.6 Resolving Scope Ambiguity . . . 74
3.4.7 Compositionality and Underspecification . . . 83
3.4.8 Advantages of this Framework . . . 85
3.4.9 Other Frameworks . . . 86
3.5 Conclusion . . . 88

Acknowledgements

I would like to thank my parents for their infinite support of my academic career, without them this would not have been possible. I would also like to thank Ms Anastacia Saliukova for her assistance with the verification of the cross-linguistic data used in this thesis. And many thanks to my supervisor Dr. Henk Zeevat who ushered me into the previously unknown territory of formal linguistics from my background in philosophy research.

Finally, I would like to thank the Erasmus Mundus to South Africa scholarship scheme for generously funding my research at the ILLC.

Preface

The problem of compositionality in linguistics and philosophy has become a prominent issue in contemporary scholarship concerning natural language. The debate usually takes two (not unrelated) forms, (1) what are the correct formal descriptions of the principle of compositionality and (2) whether or not such a principle is indeed consonant with linguistic data. In this thesis, I offer no new comment on either of these points. Instead, I plan to argue that notwithstanding the lack of a definitive answer to (2), there is reason to incorporate a compositional semantics for a given syntactic formalism. The first parts of this thesis both offer these reasons and attempt to define more precisely what type of relationship syntax needs to have with semantics to achieve the coveted title of compositional, i.e. an answer to (1). In the second part, I attempt to provide a compositional semantics for dependency grammar, a formalism which has historically been recalcitrant to such description, by means of a modified type-theoretic treatment of its semantics.

In part I, I briefly introduce some background and set up the controversy surrounding compositionality. Some of these issues will be relevant for forthcoming sections, others are meant to serve as foregrounding for the debate in general.

Part II will focus on formal definitions of compositionality with relation to formal language theory and motivations for the principle in terms of these definitions. I will also discuss the central matter of constituency and its role in the formal definition of the principle of compositionality.

Finally, part III offers a novel compositional account of dependency grammar in terms of Montague Grammar. Importantly, this analysis aims to respect the structures generated by the dependency syntax. The final part also tests this account on recalcitrant natural language phenomena.

The main contribution of the current work is to provide a compositional treatment of a traditionally constituentless syntactic formalism thereby reconceiving and expanding upon the definition of compositionality as it is interpreted in the literature.

Part I

Compositionality and Natural Language

1.1 Introduction

The genesis of the principle of compositionality has most often been linked to the writings of Gottleb Frege, hence the term "Frege's principle" (sometimes used synonymously). In 'Sinn und Bedeutung', Frege challenges a simple notion of compositional meaning in terms of co-reference due to Mill by testing the substitution thesis (which has been shown to be equivalent to a version of compositionality which we will be using, Hodges (1998)). Yet his distinction between sense (Sinn) and reference (Bedeutung) aims to rescue a compositional account of meaning in some form. The modern idea of the principle can be found in Partee (2004) and Montague (1974) among others and it is can be summed up as:

> The meaning of a complex expression is determined by the meaning of its component parts and the way in which they are combined.

Janssen (2012) argues that Frege was not the source (nor an adherent) of the concept of compositionality. In fact, he prescribed to a quite different principle for natural language semantics. He argues that its true origins can actually be traced further back than Frege to Lotze, Wundt and Trendelenburg. Furthermore, he claims that there is no evidence to suggest that Frege ever truly abandoned the context principle (given below) which is also referred to as Frege's principle:

> Never ask for the meaning of a word in isolation, but only in the context of a sentence (1884, x^e).

In Pagin and Westerstahl (2010a) these two principles are reconciled. The context principle is interpreted as a constraint on compositional meaning.[1] In addition, they trace the principle of compositionality back to 4th century indian thought, others to Aristotle. The exact beginnings of the principle and Frege's exact position on it are somewhat irrelevant for the present discussion. It is important, however, to note that it has held sway with formal logicians and philosophers such as Montague where it derived one of its most influential incarnations. Clearly, the principle has been assumed to be susceptible to formal description and we will follow that assumption in this paper.

In the methodology of logic and computer science, it has been considered the standard way of interpreting formal or programming languages (although alternatives do exist). Tarski's (1933) definition of truth for formal languages has a natural compositional interpretation (Janssen 2012). Davidson (1967) used what he called Tarski's T-theorem as a basis for a compositional semantics for natural language. In the 20th century the principle was widely adopted in the philosophy of language and logic, through Carnap, Quine, Davidson and various others. Now, it has become an essential part of most linguistic theories including generative theories. The connectionist debate has brought this issue to the fore in the cognitive scientific arena and in the philosophy of mind. There are many issues which connect the compositionality debate in language and mind, in fact Fodor (2001) argues that only one of these can be compositional and since language seems not to be in certain cases, concepts

[1]Dummett (1973) attempts to do something similar from a more philosophical perspective.

must be. For the most part, neither the compositionality of mind nor natural language will directly concern us here. In this part, however, we investigate the debate surrounding the principle as it is applied to natural language.

1.2 Disambiguating Compositionality

Unfortunately, there is no consensus on the correct definition of compositionality (see section 2.4 for details). Furthermore, it is not clear if it is exclusively a methodological principle or can be empirically tested (Dowty 2007). For the purposes of this thesis, we will assume some preliminaries. For one, compositionality will not be thought of as a property of a given semantics or syntax for that matter. We will follow Montague in defining it as a relationship between a certain syntax and a certain semantics. Consider the expression above again:

> The meaning of a complex expression is determined by the meaning of its component parts and the way in which they are combined.

This statement is vague and in need of clarification. The problem is that there doesn't seem to be a neutral way of going about this clarification. In this section, I attempt to stay as neutral as possible. Starting with the term "complex expression" which will be characterised as a syntactic object, as will its components. We will remain characteristically reticent about meanings and what they are precisely.

"Determined by" is usually interpreted functionally, which suggests that given a syntactic object as an input it produces a semantic object as an output. The immediate problem with this naive functional analysis is that it generates a unique semantic output for every syntactic expression but in natural language this tends to overgenerate as there are distinct expressions which arguably should be assigned the same meanings. This is a strong constraint on meaning. Consider the sentences below:

(1.2.1) Jimmy threw the ball

(1.2.2) The ball was thrown by Jimmy

The sentences above both seem to express the same meaning but consist of different lexical items such as the preposition *by* and a different method of combination. It is quite apparent from the literature, that the term 'function of' (when used for 'determined by') is not to be conceived of in its strict mathematical sense. However, if we start from the atomic elements and assign a meaning to each of those, then define a semantic rule for every syntactic rule, we have a compositionally semantic procedure which parallels every syntactic one (this is in essence Montague's homomorphism definition). This does not overgenerate, since the rules that combine (1.2.1) and (1.2.2) may be different but along with the meaning of the words they could produce the same meaning for the expressions. The functions sqrt and cbrt will both produce the number 2 when the input is 4 or 8 respectively. We will be more precise about these definitions below. But, for now, this gives us an idea of why the sentences above do not constitute a counterexample to compositionality simply stated so far.

A worry one might have from a syntactic perspective is that a ban on identical syntactic expressions with unidentical meaning follows from the simple notion of compositionality.[2] "If a language is compositional, it cannot contain a pair of non-synonymous complex expressions with identical structure and pairwise synonymous constituents (Szabo, 2007). This amounts to a ban on ambiguity. Pseudo-conditionals appear to conflict with this condition. Consider the (1.2.3) and (1.2.4) below from Kay and Michaelis (2011):

(1.2.3) If you're George W. Bush, you're now allowed to lie in the faces of trusting young voters.

(1.2.4) If you're pleased with outcome, you may feel like celebrating.

(1.2.3) and (1.2.4) seem to have the same syntactic structure and yet (1.2.3) does not express a hypothetical statement of any kind while (1.2.4) is a conventional conditional which does. The semantics of these constructions seems to be quite different. "[N]o hypothetical situation is posed; it appears that a categorical judgement is expressed...and the subject of that judgement is not the addressee but the person identified as x [George Bush]" (Kay and Michaelis, 2011:2).

A semantic worry is brought out by scope ambiguity. Consider the example below:

(1.2.5) Every boy loves some girl.

There are at least two possible readings for this sentence. The first is that every boy loves some girl in the sense that each boy loves a distinct girl. The second is that there is one girl who is extremely popular. Given our definition of compositionality, this seems in complete violation since the components and method of combination are the same and yet the resulting semantic analyses differ.

In terms of the former problem, this can easily be solved. By instead interpreting the statement about method of combination as involving the meanings of the components and not the syntactic components themselves.

> [This] permits the existence of non-synonymous complex expressions with identical syntactic structure and pairwise synonymous constituents, as long as we have different semantic rules associated with the same syntactic rules (Szabo, 2012: 70).

The latter problem of scope ambiguity has been addressed in many ways ranging from alternative syntactic combination and type shifting to Cooper Storage (Cooper, 1975). We shall address this issue in a more substantial way in section 3.4.

Lastly, what is meant by "component" as I have used it here. The convention in the literature is to take this to mean constituent. In linguistics, this is a loaded term. Usually, it refers to sets of linguistic items which act as a structural units in given expressions. There are various tests for constituency such as coordination, deletion, modification etc. Constituents are the items that appear in the hierarchical tree diagrams of phrase structure grammar. Consider the sentence (1.2.6):

[2]Or rather identical expression types to be more accurate.

(1.2.6) The host speaks to the caterer after the ceremony.

This sentence can be separated into distinct constituents. For example, the verb or predicate *speaks to* and its two arguments *the host* (subject) and *the caterer* (complement). Items such as *After the ceremony* are sometimes called adjuncts and can float independently of the other constituents. Linguists often interpret compositionality and "components" as not only involving constituents but immediate constituents. "In assigning meaning to a complex expression [the principle of compositionality] allows us to look at the meanings of constituents deep down the tree representing its syntactic structure, while (Λ) [the alternative with immediate constituency] permits looking down only one level (Szabo, 2012: 79). As we will see later, certain grammar formalisms do not involve the sort of hierarchy which incorporates multiple layers of syntactic structure and thus do not differentiate between immediate and deep constituency, in particular dependency grammars. Bloomberg defines constituents as linguistic objects which can be found in different expressions. And immediate constituents "appear at the first level in the analysis of the form into ultimate constituents" (Hodges, 2012: 249).[3]

Therefore, our tentative definition can be modified to incorporate the clarifications mentioned in this section.

> The meaning of a complex syntactic expression is a function of the meanings of its constituents and the syntactic rule used to combine these constituents.

Although, I have attempted to disambiguate certain vague notions in the basic definition of compositionality found in the literature, this task is far from over and will be picked up below. The next attempt at unpacking this concept will be concerned with the ubiquitous and most oft cited reasons expressed in favour of the principle. By grasping what is it that the principle is thought to explain, we may be able to grasp a little more of its nature. I will also briefly show why these reasons fail to establish the principle.

1.3 (Bad) Reasons for compositionality

There are three main arguments in favour of compositionality. Although, these arguments have undeniable intuitive appeal they do not establish the principle of compositionality for natural language. I will present each in turn and show why they are defective. In addition, I will argue against an assumption which is common to most of these types of arguments, namely linguistic infinity.

1.3.1 Productivity

This argument starts by observing the phenomenon of productivity in language (this is also sometimes called creativity). As language users we are able to produce an infinite number of sentences. One clear component of this type of argument (when it is argued), is that of

[3]Here Hodges offers a more formal definition of constituency which lies outside of the scope of this section.

recursion or iteration. This is one way of generating an infinite set of expressions. Another way is recombination. In everyday parlance we encounter variations of sentences often composed of the same vocabulary. And yet as Chomsky (1971: 74) puts it:

> The most striking aspect of linguistic competence is what may call the 'creativity of language', that is, the speaker's ability to produce new sentences that are immediately understood by other speakers although they bear no physical resemblance to sentences that are 'familiar'.

Natural language affords us many examples of the productivity property, from iterative constructions to our ability to generate unfamiliar yet perfectly intelligible expressions. Presumably there is something at the heart of this possibility. The best explanation is that compositionality is at the base of the productive nature of language. It is a corollary of the principle.

There are a number of problems with this argument. For the most part it is too weak. According to Pagin and Westerstahl, all that is needed to account for this phenomenon is that there is an infinite set of assignments of propositions to sentences and furthermore it "does not have to be systematic in any way, and all the syntax that is needed for the infinity itself is simple concatenation" (2010b:4).

In addition, this does not account for all natural language phenomena, for instance idioms tend to be completely unproductive. Of course, as long as we admit only a finite set of idioms into each language, this problem can be contained.[4]A more serious concern is described in Szabo (2012). If productivity is explained by compositionality, then two related things are implied. Firstly, there are infinitely many complex expressions which have been heard by no-one before. Secondly, are we committed to a view that we already understand these unheard sentences?

> This is not just the general Humean worry about induction - we have good reasons to doubt that the complex expressions we hear form an unbiased sample of all complex expressions (Szabo, 2012).

I think that this objection is baseless. It assumes that productivity is the ability to produce as actual infinity of sentences. If, however, we follow the intuitionistic conception of a constructive or potential infinity, we are not committed to the existence of unheard sentences. The problems suggested by Szabo imply that natural language constitutes an actual infinite set in terms of a completed mathematical set with infinite expressions. On the contrary, a finite syntax with a lexicon (compositionality) only implies that we have an infinite procedure for producing expressions not that we have an actual infinity of them. I think that this blocks both arguments since the potential we have to produce and understand unheard expressions is a far less controversial claim upon which to base compositionality than the claim that there is an actual set of all of these expressions.

[4]It becomes a difficulty when we consider arguments which take noncompositional "constructions" to be ubiquitous in natural language as is the case in Construction and Radical Construction Grammar (Goldberg 2003, Croft, 2001)

There is a much more serious worry for adherents of this argument, namely the possibility of productivity without compositionality. If compositionality can be shown to not be a necessary condition for productivity then even if we accept that language is productive we needn't accept that it is compositional.

Werning (2005) claims that a simple language with a productive rule of holophrastic quotation and a meaning function which allows for synonymy can be proven to be non-compositional[5]. First we specify the syntactic operation of quotation

> if $q : E \rightarrow E$ and $s \mapsto s'$ such that the corresponding semantic operation yields $\mu(q(s)) = s$

A language which contains this rule is automatically productive since quotation can be iterated. "This account of quotation might be called holophrastic quotation because it takes well-formed phrases as unanalyzed wholes and sets them in quotation marks. The meaning of a quotation is the quoted expressions (Werning, 2005: 296). From this, we can show that this simple language is not compositional. Let compositionality be defined as a rule (C) such that for any semantic function μ_q for the syntactic operation q, $\mu(q(\theta)) = \mu_q(\mu(\theta))$

(a) Take two sentences $\ulcorner\phi\urcorner$ and $\ulcorner\psi\urcorner$ such that $\mu(\ulcorner\phi\urcorner) = \mu(\ulcorner\psi\urcorner)$ but $\ulcorner\phi\urcorner \neq \ulcorner\psi\urcorner$, where $\ulcorner\urcorner$ denotes metaquotation.
(b) From the quotation rule the following holds: $\mu(\ulcorner\phi\urcorner) = \mu(q(\ulcorner\phi\urcorner)) = \ulcorner\phi\urcorner$ and $\mu(\ulcorner\psi\urcorner) = \mu(q(\ulcorner\psi\urcorner)) = \ulcorner\psi\urcorner$.
(c) Given (a) and (b), we know that $\mu(q(\ulcorner\phi\urcorner)) \neq \mu(q(\ulcorner\psi\urcorner))$ since $\ulcorner\phi\urcorner \neq \ulcorner\psi\urcorner$.
(e) Compositionality (C) entails that $\mu(q(\ulcorner\phi\urcorner)) = \mu_q(\mu(\ulcorner\phi\urcorner))$ and substitution of identicals into (a) gives us $\mu(q(\ulcorner\phi\urcorner)) = \mu_q(\mu(\ulcorner\psi\urcorner))$
(f) By (C) again, we get $\mu(q(\ulcorner\phi\urcorner)) = \mu(q(\ulcorner\psi\urcorner))$ which is a contradiction, given (c).[6]

The above argument shows that productivity does not imply compositionality. In addition, there are no good arguments to show that it does.

1.3.2 Systematicity

Systematicity is supposedly another underlying property of natural language upon which compositionality is based. The idea is that there are certain patterns that we encounter which allow for recombination in such a way that it implies compositional meaning. For instance, if we understand xRy then we can understand yRx merely by understanding the meanings of the components of the relation and the way in which they are combined. A concrete example of this phenomenon is that given a certain sentence 'The man loves the woman' we can deduce the meaning of 'The woman loves the man'. If it were not for the systematic nature of language, it is argued, then we could not account for patterns like these.

[5]The property which allows for synonymy is called non-hyper-distinction. I will formally define this property in section 2.4.3

[6]There are compositional accounts of quotation on the market. However, the possibility of a compositional analysis does not affect the main thrust of the argument presented here.

Another way to put this is that there is a nomological connection between these sentences and this connection is best explained by compositionality.

Fodor is especially fond of this property of natural language understanding and cites the inability of connectionist cognitive architectures to capture it as one of their fatal failures. However, it seems that the property is more convincing from a cognitive point of view than from a linguistic one.[7] Consider the concept of a red circle inside of a blue square. It seems reasonable to claim that an individual who understands the concept of a red circle and the concept of a blue square can understand both the initial concept and the derived concept of a blue circle inside of a red square. There seems to be a systematic relationship between concepts which may or may not be based on compositionality of thought but the case for language is less obvious. The conventional examples of systematicity in natural language are often misleading. Consider the examples below:

(1.3.1) The woman loves the man $\Leftrightarrow$ the man loves the woman

(1.3.2) The woman loves the car $\not\Leftrightarrow$ the car loves the woman

In the pair (1.3.1) the systemic link between the sentences is clear but from (1.3.2) we see that this pattern is not generalizable, i.e. the systematic interpretation is blocked. There are many examples of such substitution failure for systematic links between sentences and this serves to indicate that there is something about the meaning of the specific units of such pairs at work. Werning claims that the "systematic correlation is warranted only if the permuted words belong to the same *Bedeutungskategorie* - *semantic category* or simply *category* - as defined by Husserl" (2005: 302). Thus, the property of systematicity is limited in scope and hence rather weak.

Secondly, the argument against the compositionality of adjectives provides an interesting (and novel) extension against the systematicity. Standardly, we think of adjectives compositionally. For example, the semantic contribution that the adjective 'red' makes to an expression, or rather noun x, is thought to be the same the one it makes to a noun y. The adjective makes the same contribution semantically to the adjective-noun pair 'red table' as it does to the pair 'red bookshelf'. The claim here is that even in the cases which are paradigmatic or assumed to be trivial, this compositionality assumption does not hold. In addition, to understand a novel adjective-noun expression is to appreciate linguistic and world knowledge beyond the mere composition of already understood semantic units.

The first thing to do is to reformulate the principle of compositionality in terms of adjectives. Lahav accomplishes this task by attempting a theory neutral definition of the 'applicability conditions' of an adjective:

> The conditions that have to be satisfied by any object under any (correctly ascribed) noun in order for the adjective to correctly apply to that object; for example, the conditions under which an object is describable by 'red N', for any noun N (1989:262).

[7]We would favour Fodor's argument that compositionality holds for thought as opposed to natural language, "As between the two, at least one of thought and language must be compositional" (2001).

Thus, an extension of the principle of compositionality is possible for adjectives

> (C”) Every adjective *a* has uniform applicability conditions across all compound expressions in which it is embedded.

The idea behind this is that if two adjectives have different applicability conditions then clearly they also possess different meanings. The principle requires that adjectives have uniform applicability conditions in all the standard complex expressions in which they occur. Thus, the conditions that have to be met for something to be appropriately described ’red table’ should be the same set of conditions that are to be met for ’red bookshelf’ or ’red fish’. Of course context dependence should be allowed for, “hence, by requiring that the applicability conditions of an adjective be uniform across all linguistic contexts, the principle should be understood as requiring that for every adjective there is a general function from contexts to applicability conditions” (Lahav, 1989: 262).

The problem arises when we appreciate the different types of conditions which determine the applicability of a given adjective. The conditions for which a person is said to be ’good’ is different to those for a computer, similarly for other adjectives. This is a trivial fact. A good person is virtuous and kind while a good computer is one with a large hard-drive and fast processing speed which has nothing to do with personality or virtue. Lahav claims that “virtually all” adjectives behave in this way such that there is no fixed set of conditions which determine their applicability to different objects. In other words, the applicability conditions of adjectives is largely a noun-dependent matter. This problem arises even if we consider only the simplest colour predicates. The conditions for a bird to be red or blue are different than for a car or a crystal to be these colours. Arguably, the former only have to have red feathers or body surface apart from its feet and beak etc. while a car need only be red on the outside and a crystal needs to be red all over. “In short, what counts for one type of thing to be red is not what counts for another” (Lahav, 1989: 264). Similarly for almost every other adjective.

The main point to be drawn from this is that the applicability conditions of adjectives vary in such a way from one noun to another that one cannot speak of a general function from contexts to applicability conditions. Therefore, the semantics of adjectives must be non-compositional in nature. The absence of applicability conditions for some adjectives to certain objects corroborates this conclusion. The unintelligibility of adjective-noun pairs such as ’the loud tree’, ’an orange thought’, ’annoyed building’ etc. does not mean that the meanings of these expressions are *a priori* or inherently unintelligible (cf. ’the true falsehood’ or the ’round square’). We could imagine circumstances in which it would be felicitous to use these expressions. Thus, argues Lahav, the applicability conditions of adjectives develops piecemeal and not uniformly.

It is not clear that adjectival composition cannot be accommodated in a composition semantics (in terms of constraints or parameters on the meaning function). However, the reasoning that led to this counterargument provides a more sound basis for criticizing systematicity. It suggests that the following pairs of more obviously linked sentences are not even as systematic as we would expect

(1.3.3) The red bird and the yellow fruit $\not\Leftrightarrow$ the yellow bird and the red fruit

(1.3.4) The thin man and the lame excuse $\not\Leftrightarrow$ the lame man and the thin excuse

Therefore, not only is the property of systematicity rather weak in that it doesn't seem to support more than certain obvious cases of semantically linked phrases but it doesn't seem to support even these simple cases upon closer inspection.

1.3.3 Novelty and Learnability

The novelty and learnability arguments are related to both systematicity and productivity mentioned above in some ways. Novelty can be summed up as "speakers are able to understand sentences they have never heard before, which is possible only if the language is compositional" (Pagin & Westerstahl, 2010: 3). Very often as speakers of a language, we hear sentences which we have not heard before, according to the latter reasoning the only way in which we can understand these sentences is on the basis of their component parts and a finite set of composition rules. For a hearer to attach a meaning to a completely new sentence, she has to compute this from the meanings of previous fragments of language which she has already encountered.

This argument is inherently related to the argument from learnability, mostly famously presented by Davidson (1967:17):

> It is conceded by most philosophers of language, and recently by some linguists, that a satisfactory theory of meaning must give an account of how the meanings of sentences depend upon the meanings of words. Unless such an account could be supplied for a particular language, it is argued, there would be no explaining the fact that we can learn the language: no explaining the fact that, on mastering a finite vocabulary and a finite set of rules, we are prepared to produce and understand any of a potential infinitude of sentences.

This 'infinity' of natural language which Davidson takes for granted is challenged by Groenendijk and Stokhof (2004). They ask exactly what sort of infinity is supposed to be captured by natural languages. Surely it cannot be the same as the infinity of formal languages since these are defined to be infinite. The matter is different for natural languages, "for natural languages the issue of its finiteness or infiniteness must be an empirical one, and hence of a completely different order?" (Groenendijk & Stokhof, 2004:8). Furthermore, the type of evidence Davidson had in mind will not do. For example, the claim that a simple sentence can be extended *ad infinitum* by means of conjunction/disjunction or embeddings of various sorts is not empirically valid. This debate then turns on the Chomskyan distinction between competence and performance. Natural language as it is performed is fraught with limitations, limitations which Groenendijk and Stokhof argue are not contingent but necessary corollaries of our finite cognitive capacities, i.e. our competence. This will be further argued in the next section.

It is on the basis of the assumed novelty and learning arguments that Chomsky proposed innate linguistic categories in the human brain or the well-known Innateness Hypothesis. This is an account of language acquisition which claims that there is an initial store of linguistic information already present in the human brain at birth. Fodor takes it a step further with the Language of Thought (LOT) hypothesis (1975) positing that you need to know a language to learn one. Thus, our linguistic experiences fit neatly into preordained categories and compositional rules, whether in the form of the phrase structure rules of generative grammar or otherwise. The basic idea behind LOT is that the process of thought is conducted in a ”mentalese” or a symbolic mental language which is somehow realised in the physical brain. This mental language is essentially compositional in nature.

An empirical objection to this account of learning comes from the work of Tomasello on language acquisition. In arguing for a usage-based account of language acquisition, Tomasello puts forward the 'Verb Island Hypothesis'. Essentially, what this hypothesis claims is that evidence points to the fact that during the early acquisition period, children isolate each of the newly acquired verb forms in terms of its own idiosyncratic linguistic space that does not correspond to the syntactic categories posited by generative grammar. More specifically, “children's earliest language is based on the specific linguistic items and expressions they comprehend and produce” (Tomasello, 2000: 161).

This evidence would serve as a powerful objection to the Innateness Hypothesis, if it proves correct. Furthermore, there is some evidence to suggest that certain linguistic constructions are stored as individual units (see Bybee and Scheibman 1999) and not as part of larger rules and compositional hierarchies. “[I]t is obvious to all empirically orientated students of language acquisition that children operate with different psycholinguistic units than adults (Tomasello, 2000: 62).

Pagin and Westerstahl argue further that even if the infinity premise is granted, it would only mean that the semantics of NL is computable. And computability does not entail compositionality by itself. These considerations all point to the realization that the principle of compositionality is theoretical in nature and not empirically transparent. This, of course, does not mean that the principle is false just that it is not obviously true. And some of the premises of the usual arguments in its favour are in need of more justification.

1.3.4 Infinity

All of these arguments, except for the systematicity argument, rely on a fundamental assumption about natural language, namely that it constitutes a countably infinite set of expressions. This assumption is in serious need of justification in order to do the work for which is it used in these arguments.

NL as recursively enumerable: An Argument in favour

In this section, I investigate the standard argument for infinity in NL. One of the major concerns in the literature on infinity in NL is that there is a patent lack of a positive argument, I attempt to offer one such argument here. I will then show that there is reason to doubt the validity of such arguments in general.

I will briefly describe a response or rather type of response that the defender of NL-infinitude could provide. Although, this is the received position, it has not often been argued for in precise terms. One clear component of this type of argument (when it is argued), is that of recursion or iteration. In other words, they are usually centered around the idea that there is no longest expression in natural language. If the sentence *I am hungry* and the sentence *I am very hungry* are distinct sentences and there is no upper bounds on the amount of modifiers *very* we can attach, then presumably we can have an infinite number of sentences capable of such modification. Pullum and Scholz (2010: 3) offer other syntactic facts to capture this:

> I exist is a declarative clause, and so is I know that I exist, and so is I know that I know that I exist...;very nice is a adjective phrase, and so is very nice, and so is very nice; and so on for many other examples and types of examples.

The intuition is that at no obvious point do these sentences become ungrammatical. From this, they derive the No Maximal Length claim (NML), i.e. that for every English sentence there is a longer one. Notice, that this denotes a constructive set or rather this does not amount to a claim that there is an actual set containing all such sentences. The claim is that from a given sentence such a sequence can always be constructed.

Infinitude is generally assumed to be a consequence of NML or similar observations. As we can see from Stabler (1999:321) when he states that:

> There seems to be no longest sentence, and consequently no maximally complex linguistic structure, and we can conclude that human languages are infinite.

In the next sections, following Pullum and Scholz, we investigate how we get from NML to infinitude by mathematical means.

Problems for the standard argument

Although, the intuition behind the above argument seems clear, the actual mathematics is much harder to establish. It is not at all clear that inductive generalisation or mathematical induction is at work here. When we argue for the infinity of the natural numbers, we first have to characterise $\mathbb{N}$ in terms of Peano's axioms, then we run a mathematical induction argument to show that a certain property is inherited under succession. But this only works if we have Peano's successor and unique successor axioms. "That is, we must assume both that every English expression length has a successor, and that no two English expressions lengths share a successor. But to assume this is to assume the NML claim" (Pullum & Scholz, 2010: 7). Obviously, this will not do.

The argument from generative grammars fares no better. Since "from the viewpoint of effective calculability, general recursion and Turing computability are equivalent" furthermore "a universal Turing machine can enumerate any recursively enumerable formal language...as can general recursion" (Luuk and Luuk, 2011: 1943). It is argued that in order to capture the syntactic facts which lead to NML, we need to use a generative grammar with an element of recursion. There are two points of difficulty here, firstly there are generative frameworks with recursive rules that do not entail NML or infinity and secondly, there are non-generative means of representing the syntactic facts under discussion. We consider only the second problem since the first is not very general.[8]

There are three alternative ways of representing the facts of multiple iteration which does not involve recursion or any size claims, through transducers, category theory based transformational grammar and constraint-based grammars such as HPSG, GPSG, LFG etc. All of these options are completely neutral with respect to infinity. Thus, generative grammars with recursive rules are only sufficient but not necessary for infinity (and perhaps not even sufficient if we do consider the generative grammars that do not produce infinite expressions).

> For example, suppose the grammar of English includes statements requiring (i) that adverb modifiers in adjective phrases precede the head adjective; (ii) that an internal complement of know must be a finite clause or NP or PP headed by of or about; (iii) that all content-clause complements follow the lexical heads of their immediately containing phrases; and (iv) that the subject of a clause precedes the predicate (Pullum and Scholz, 2010: 11).

(i) to (iv) above are able to capture all of the linguistic facts that led to NML but are perfectly compatible with different answers to the infinity question, i.e. with a finite collection too.

An Argument from ℕ to NL

I do, however, think that there is a better argument lurking in the claims of generative linguistics. Consider this passage from Pinker (1994: 86):

> By the same logic that shows that there are an infinite number of integers - if you ever think you have the largest integer, just add 1 to it and you will have another - there must be an infinite number of sentences.

Clearly, the above reasoning falls prey to the problems of the subsection above but there is an interesting notion of the relationship between NL and the natural numbers here, picked up in Chomsky (2010) when he says:

> That brings up a problem posed by Alfred Russell Wallace 125 years ago: in his words, the "gigantic development of the mathematical capacity is wholly unexplained by the theory of natural selection, and must be due to some altogether distinct cause," if only because it remained unused. One possibility is that it is derivative from language. It is not hard to show that if the lexicon is reduced to a single element, then unbounded Merge will yield arithmetic.

The *Merge* operation takes two syntactic objects and merges them into one such object. According to Chomsky, unbounded *Merge* yields a discrete infinity of expressions in NL. The idea in this paper is that recursion (which is a form of unbounded *Merge*, according to

[8]It could be argued that such grammars can be ruled out in the stipulation of a linguistic theory. Cf Pullum and Scholz (2010).

Chomsky) in the language faculty or rather our "cognitive organ" itself, is responsible for infinity in both NL and $\mathbb{N}$.

I will first present his argument and then a mathematical proof loosely based on it.

> Firstly, (a) the successor function for $\mathbb{N}$ is recursive in nature,
> (2) arithmetical computations can be made by humans given enough time etc.
> (3) Therefore, recursion is neurally implemented for $\mathbb{N}$ computations.

The reason for this is the infinity present in NL and $\mathbb{N}$ and since $\mathbb{N}$ is generated by the language faculty, recursion is neural component of our cognitive organ. The above argument is fraught with problems. As with the standard argument, it seems to assume infinity and use neural recursion to explain it, "the concept of neurally implemented recursion is largely motivated by the 'discrete infinity' property of natural language" (Luuk & Luuk, 2011: 1945).[9] Furthermore, it is not clear that we have to posit neural recursion to account for our ability to compute $\mathbb{N}$, iteration might work just as well. But importantly, almost any means of calculation could replace recursion. Consider the example from Luuk and Luuk (2011):

> We can conceive that all natural numbers are derived from the number 20098 by ± 1 operations, i.e. each time we conceptualize a natural number x that is less than 20098, we subtract 1 from 20098 until we get x and each time we conceptualize a natural number y that is greater than 20098, we add 1 from 20098 until we get to y (1946).

The point is that this principle is equally conceivable and yet it is unlikely that it is neurally implemented. Of course, the recursion principle can account for the discrete infinity of expressions but as we concluded above, this infinity was assumed and not proven.

A Proof of the Countable Infinity of NL

From the relationship suggested in the previous section, one can construct a better argument in favour of the 'discrete infinity' of NL.

Proof: Claim - NLs are countably infinite.

a. Let NL be the set of all the expressions of a given natural language.

b. Now, let $N_{\mathbb{N}}$ be the set containing all the names of the natural numbers such that $N_{\mathbb{N}} \subset NL$. We can see that NML holds for $N_{\mathbb{N}}$ since there is no largest number name. We also know that there is a bijection $\pi : N_{\mathbb{N}} \longmapsto \mathbb{N}$.

c. Thus, $N_{\mathbb{N}}$ is countably infinite since $\mathbb{N}$ is.

d. Now, we define a function f such that $\forall x \in NL$, where x is a well-formed formula of NL (i.e. a grammatical sentence), $f(x) = n_i$ where $n \in N_{\mathbb{N}}$ and $i \in \mathbb{N}$. Thus, there is a bijection $\pi' : NL \longmapsto N_{\mathbb{N}}$.

e. Since we know that $N_{\mathbb{N}} \subset NL$ and that $N_{\mathbb{N}}$ is countably infinite, we can conclude

[9] Tiede and Stout (2010) explain this question begging behaviour as a modeling choice not derivable from fact or proof but stipulated in models of NL.

that NL is countably infinite since there is a bijection between it and one of its proper subsets.

Therefore, given (a)-(e) the claim is true.

For now, the above proof can serve as an explicit statement of the claim that NL is countably infinite and thus recursively enumerable.[10]This proof has not been represented in the literature thus far (as far as I am aware) and it mathematically captures the nature of our intuition about infinity in natural language in a straightforward way. It relies on the connection between natural language and the natural numbers brought out by Chomsky above. However, it represents this connection in a benign manner as a claim that the names of the natural numbers are a part of natural language. A claim with which most people would agree.

The next step is also relatively uncontroversial, namely that there exists a one-to-one mapping between these names and the numbers they denote.

The last step involves labeling each element of the natural language in question with a natural number name. This step tacitly assumes a countable infinity otherwise this labeling would not work. Furthermore, it presupposes that certain natural language structures can be captured by discrete mathematics. In the next section, we question this assumption.

Well-Definition vs ill-Definition

In Hockett's book, *The State of the Art* (1968), he criticizes the generative tradition from many perspectives. The main challenge seems to be rooted in the idea of NL as a well-defined system. He argues that it is this assumption that has led to numerous mischaracterisations of natural language. He starts by differentiating between two definitions, well-definedness and ill-definedness respectively in terms of one another. The important definition is given below:

> Well-defined system: one which can be characterised completely in terms of deterministic functions (in terms of Turing machines and the like).

The assumption here is that if a given set is infinite, it can be expressed by a finite device which serves to enumerate the set. In terms of NL, the grammar would serve as this finite device. Of course, if indeed NL is charactizable by such devices, then this would mean that NL is denumerably infinite. For Chomsky, NL grammars are just types of Turing machines, "the weakest condition that can significantly be placed on grammars is that F be included in the class of general, unrestricted Turing machines" (1959: 138). Importantly though, Turing machines are based on iteration as opposed to recursion which is the mechanism Chomsky believes is at the heart of NL grammars.[11]

[10]There is an argument due to Langendoen and Postal (1984) in which it is argued that NL is strictly larger than $\aleph_0$ and thus equal in size to the set/class of all sets, i.e. non-denumerable. It is, however, beyond the scope of the current paper to deal with this possibility. Suffice to say, the Vastness proof (as it is called by Langendoen and Postal) is a well-argued formal result which is not to be as easily disregarded as it has been.

[11]Recursion involves self-reference while iteration does not and thus requires control flow loops to define infinite sets (Luuk & Luuk, 2011: 1942). However, recursive systems can be shown to be equivalent to Turing machines for finite outputs.

So, like a Turing machine, a grammar can supposedly enumerate all the members of NL through a fixed number of operations and a finite number of steps. The appeal of such characterisations can be seen in arguments from learnability and novelty, ubiquitous in philosophy and linguistics. As Hockett, correctly notes, holding that NL is a type of Turing machine or any such device presupposes that NL is a well-defined system. Let us consider a less controversial well-defined set, namely the set of n-place decimal approximations of π. Not only does this constitute an infinite set but "there is an algorithm which enables us, given any positive integer n, to compute the n-place approximation" (Hockett, 1968: 46). Thus, we have a well-defined set since there is a finistic procedure for enumerating the objects of the infinite domain of π approximations.

What of the concept of an ill-defined set/system? Simply enough, this is defined as a system or set which cannot be characterised by the computable functions of the sorts we have been discussing (important to note is that these sets are neither computable nor noncomputable (Hockett, 1966, 47)). Consider the set of possible cricket scores in a given game (assume a one day international). Since there are possible scores from 1 to 6, thus the final score can be given by the expression $a + 2b + 3c + 4d + 5e + 6f$ where $a, b, c, d, e, f \in \mathbb{Z}^{+}$. Given the set of rules of cricket (which is quite a large set by comparison to other sports), and the limit of one day for a game, certain scores are not possible. For instance, scoring 100 000 runs (points) in a given game should be ruled out. But surely, the highest score on record (of 872 runs) could have been higher than it is. Thus the set of possible scores does not have a precise upperbound but it does have limits such as a score of 100 000 runs. Attributing precise upperbounds to this set is not possible since we are not dealing with a well-defined system. Even to attribute finitude (in the mathematical sense) would be strictly inappropriate. The problem is that cricket has factors such as player fatigue, talent and skill which contribute to the potential score of a given game and these cannot be characterised by discrete mathematical means.

This is a general problem with physical systems, Hockett argues. It is hard to see how any physical system could be defined in terms of deterministic functions since they are subject to eventual decay and "thermodynamic indeterminacy" only characterisable in terms of the tools of continuous mathematics. These systems are outside of the purview of computability theory.

The ill-definition of NL

From the previous section, we know a set can only be considered infinite if it is well-defined. It remains to be shown that NL is not well-defined. Furthermore, in some sense, if a set is not infinite then it is finite. For instance, the possible score of a cricket game is finite in this sense even if not in the purely set theoretic one.

The argument for why NL is ill-defined is quite simple. First, Hockett assumes a seemingly innocuous principle, namely the 'Law of Conservation of Well-Definition'. This principle guarantees that one well-defined system can beget another (1968: 58). He then asks from which well-defined system language comes, if it is assumed to be one. He suggests two options for the genesis of language: either through genes or through cultural transmission.

If we consider the first option, we fall into the previous problems of physical systems. The biological system that is responsible for our organism is certainly not a well-defined system in the way that we have been discussing. The system of biology or neurobiology (since language is probably more related to this field) is not characterisable by computable or deterministic functions.

This leaves the cultural transmission of language as a candidate for a well-defined system. Clearly, this will not do either. Historical linguistics has shown that this process is anything but mathematically precise. Language evolution is subject to language contact, change and myriad environmental factors. If NL is well-defined, then it cannot be due to its cultural transmission.

Therefore, one can either accept the above or deny the Law of Conservation of Well-definition. Chomsky opts for the former, but maintains that NL is a well-defined system.

> The speed and precision of vocabulary acquisition leaves no real alternative to the conclusion that the child somehow has the concepts available before experience with language and is basically learning labels for concepts that are already part of his or her conceptual apparatus. This is why dictionary definitions can be sufficient for their purpose, though they are so imprecise. The rough approximation suffices because the basic principles of the word meaning (whatever they are) are known to the dictionary user, as they are to the language learner, independent of any instruction or experience (1987).

This innate conceptual system is a well-defined system of linguistic universals which are then instantiated for particular languages by experience. The difference between competence and performance rests in the distinction between capacity and experience. Ultimately, the goal of Government & Binding and Minimalism is to discover the characteristics of this innate grammar. This is the reason behind the methodology of utilising introspective linguistic intuitions as opposed to corpus data. Unfortunately, we do not have any other insight into this innate grammar besides evidence from generative linguistics which is compatible with other explanations.

It is at this point that the structuralist opposition enters the fray. Hockett claims that the distinction between competence and performance is an illusion and to assume an underlying structure to performance which meets the criteria of well-definition simply begs the question.

These matters have been argued at length (and more recently from the computational linguistics perspectives)[12] and I do not see any point in rehashing the debate here. The main point is that infinity cannot be the basis for compositionality since it has never truly been shown that natural language is in fact infinite in the formal sense of the word.

1.4 Conclusion

From the above sections, it seems clear that the appeal to compositionality for natural language is an appeal to the best explanation. Furthermore, a central assumption, namely the

[12]Cf Lenci (2008).

countable infinity of natural language, upon which many of these arguments are based turns out to be unjustified. In addition, there are a plethora of linguistic examples which apparently force noncompositional analyses of language (I mentioned a few above, conditionals, adjectives, idioms etc.).

The main point of this part of the research is that compositionality is not a principle drawn from empirical observation and in the next part I argue that it is better seen as a methodological principle. The principle will be shown to be a useful tool for capturing the syntax-semantics interface in formal theories of language where it fails to do so conclusively for natural language.

Part II

Compositionality and Formal Language

2.1 Introduction

The concept of a formal language is familiar from logic and mathematics. It is a useful tool in identifying mathematical structures as well as modelling various phenomena from metaphysical entities to natural languages. For the most part, the principle of compositionality is taken for granted in formal logic. Propositional logic syntax is defined recursively and for each syntactic rule there is a corresponding semantic interpretation. Consider the rules for conjunction and disjunction. Let PL be the language of propositional logic with the usual individuals, variables and connectives.

(1) If $\phi \in PL$ and $\psi \in PL$ then $\phi \wedge \psi \in PL$.
(2) If $\phi \in PL$ and $\psi \in PL$ then $\phi \vee \psi \in PL$.

For semantics, we have a two-valued interpretation function I that maps sentences of PL to either 0 or 1. I assigns to all 2-place logical constants a fixed function from $\{0,1\} \times \{0,1\}$ into $\{0,1\}$ (this is just another way to write a truth-table). So the valuation $V_I(\phi)$ for complex sentences under the interpretation I is compositionally defined as

If $\phi = (\psi \diamond \chi)$, then $V_I(\phi) = I(\diamond)(V_I(\psi), (\chi))$

Thus the *meaning* of conjoined or disjoined sentences is given as

(1') $V_I(\phi \wedge \psi) = 1$ iff $V_I(\phi) = 1$ and $V_I(\psi) = 1$
(2') $V_I(\phi \vee \psi) = 1$ iff $V_I(\phi) = 1$ or $V_I(\psi) = 1$

Propositional logic is a good example of a formal language with a simple compositional semantics. The meaning of a formula is a truth value and the meaning of a complex formula is a function of the meanings/truth values of its components. Predicate logic is not as simple a matter. Following Pratt (1979), we know that "there is no function such that meaning of $\forall x \phi$ can be specified with a constraint of the form $\mathcal{M}(\forall x \phi) = F(\mathcal{M}(\phi))$" (Janssen, 1997). In other words, the meaning of a universally quantified formula is not straightforwardly given in terms of a function from the meaning of its parts, at least not by means of the standard Tarskian interpretation. However, the situation is resolvable

> The standard (Tarskian) interpretation of predicate logic is not a meaning assignment but a recursive, parameterized definition of truth for predicate logic. It can easily be turned into a compositional meaning assignment by incorporating the parameter (viz. the assignment to variables) into the concept of meaning. Then meaning becomes a function with assignments as domain (Janssen, 1997).

Programming languages also have compositional interpretations. This is partly to keep track of large-scale programs in a proportional way. Denotational semantics, also known as mathematical or Strachey-Scott semantics, in a popular method of interpreting programs. It is concerned with defining mathematical objects called domains which represent what computer programs do. Importantly, denotational semantics is compositional since the denotation of a program phrase needs be built out of the denotations of its subphrases.

Interpretation of programming languages and Discourse Representation Theory (DRT) (Kamp 1981) informed the dynamic turn in semantics. In DRT, meanings are considered to be discourse representation structures or DRSs which are similar to databases. This theory both offers an account of mental representation (in terms of DRSs) and anaphora resolution. However, due the nature of discourse referents, the theory failed to be completely compositional in the strict sense given its method of pronoun resolution.[13] Dynamic Semantics was developed as a compositional treatment of dynamic meaning which aimed at representing discourse phenomena and anaphora resolution (Groenendijk and Stokhof 1991a). Dynamic logic incorporates a theory which takes meaning not as truth-conditions but rather as potential information change. "Dynamic semanticists do hold that compositional meanings have the nature of functions or relations, while the classical meanings have the status of projections of the compositional meanings" (van Eijck and Visser, 2012).

Semantics and its compositional development has been largely divorced from the development of formal syntax. Where semantics was primarily informed by the philosophy of language and the principle of compositionality, formal syntax took inspiration from formal language theory and linguistics. The result has been that formal semanticists and formal syntacticians have not been directly involved with each other's research and advances in either field have not always been translated into the other. Thus a central research question has become concerned with the syntax-semantics interface. Essentially, this is wherein the compositionality question lies. And the purpose of the next part is to review the syntactic side of the compositionality debate and to formally define the relevant notions needed to describe the principle.

2.2 Formal Language Theory

Formal language theory is concerned with natural language syntax, primarily its descriptive adequacy and parsing complexity. Through the use of various abstractions, the syntactic structure of natural languages are characterised using mathematical tools. The components of formal language theory are expressions which are viewed as (finite) strings of symbols and languages which are sets of these strings. The famous 'Chomsky Hierarchy' is a means of representing these languages in a sequence of ascending nested complexity. There are four traditional levels of the hierarchy, but in this section we will introduce another level. I will briefly describe each level below. Before I do that, I will provide some details which characterise the formal system of formal language theory.

Firstly, a formal language is a set of sequences of strings over a finite vocabulary which we will call T. The members of this set vary according the field to which we apply formal language theory, i.e. words if we are talking about natural languages or states if we are talking about programming languages etc. Furthermore, in formal language theory we are concerned with the finite ways in which these languages can be described, "FLT deals with formal languages (= sets of strings) that can be defined by finite means, even if the language

[13]Although, there are a number of compositional versions of DRT, see Zeevat (1989), van Eijck and Kamp (1997).

itself is infinite" (Jäger and Rogers, 2012). This is done by means of grammars. In essence, grammars are just sets of rules by which we construct well-formed sentences

> A grammar $G = \langle T, NT, S, R\rangle$ where T is a set of terminals (the vocabulary), NT is the set of nonterminals (the alphabet is $T \cup NT$ such that $T \cap NT = \emptyset$), S which is a unique start symbol and R is the set of production rules.

This is all very abstract and the specific types of production rules will distinguish the different languages in the hierarchy. In general rules look like $\alpha \to \beta$ where the arrow denotes "replace α with β" and α and β are sets of stings from the alphabet (either terminal or nonterminal).

> G will be said to *generate* a string w consisting of symbols from Σ if and only if it is possible to start with S and produce w through some finite sequence of rule applications. The sequence of modified strings that proceeds from S to w is called a *derivation* of w. The set of all strings that G can generate is called the *language* of G, and is notated $\mathcal{L}(G)$ (Jäger and Rogers, 2012).

Another important component of formal language theory is decidability. Given a string w and a formal language $\mathcal{L}(G)$, there is a finite procedure for deciding whether $w \in \mathcal{L}(G)$, i.e. a Turing machine which outputs "yes" or "no" in finite time. In other words, a language $\mathcal{L}(G)$ is decidable if G is a decidable grammar. This is called the membership problem. In what follows, I will briefly describe each level of the Chomsky Hierarchy guided by the question of which formal language best captures the syntax of natural language.

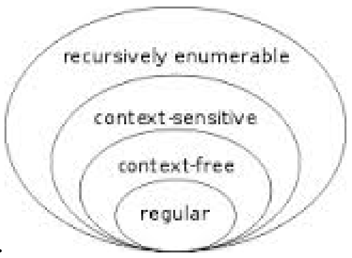

Fig 1.

2.2.1 Regular Languages (Type 3)

Regular languages have the most restricted type of generating grammar in terms of the nested hierarchy (it contains none of the other grammars) and in most cases is the simplest to parse since it lacks recursive structures. The production rules for regular languages are $A \to a$,$A \to aB$ or $A \to \varepsilon$ (ε is the empty string) such that $A, B \in NT$ and $a \in T$. Regular grammars can be represented as finite state automata (FSA) and thus are often referred to as finite state languages. S is identified with the initial state and the symbols of the string are the transitions. Since there is a finite nonterminal alphabet there is a terminal state. Thus, given a string w if there is a path through a FSA, then w is generated by the grammar and vice versa where an FSA is just a machine that processes one symbol at a time and changes its state in accordance with the symbol just processed. Below is a diagram of a normal FSA.

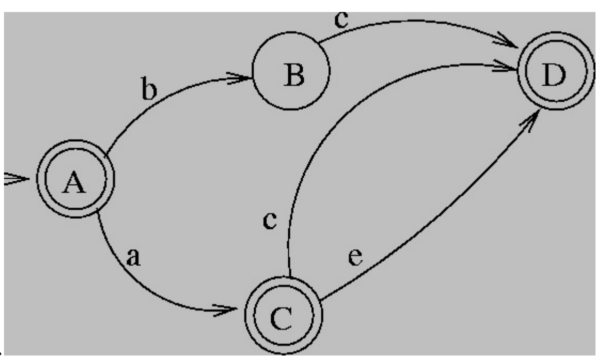

Fig 2.

It can be shown that natural language is not regular in this way or in other words, that there is no FSA that can represent certain structures. This result was famously shown by Chomsky in 'Three Models for the Description of Language' (1956). I will provide an outline of it here.

We know that patterns such as $a^n b^n$ are not analysable on a FSA since (informally) it would have to remember this number n while waiting for the end of the b's.[14]The argument goes that certain English constructions can be modelled on $a^n b^n$. The claim is that paired dependencies such as *either...or*, *if...then* and certain agreement relations can nest within one another to an arbitrary depth.

> Anyone$_1$ who feels that if$_2$ so-many$_3$ more$_4$ students$_5$whom we$_6$ haven't$_6$ actually admitted are$_5$sitting in on the course than$_4$ones we have that$_3$ the room had to be changed, then$_2$ probably auditors will have to be excluded, is$_1$ likely to agree that the curriculum needs revision (Chomsky and Miller, 1963).

The above example illustrates that English may have structures that go beyond the bounds of regular languages. Thus central embedding[15] is a common natural language phenomenon which cannot be captured by FSA and consequently regular grammars. These examples are enough to suggest that if English is not a regular language then other natural languages probably are not either. At the next level of the hierarchy we find languages which can capture these phenomena and more.

2.2.2 Context-free Languages (Type 2)

Context-free languages (CFG) were believed to contain the grammars that can capture the syntax of natural languages. The production rules for CFGs are one of two possibilities, either $S \rightarrow ab$ or $S \rightarrow aSb$. As we can glean from the latter rule, CFGs allow for recursive structures. We can also see that CFGs contain regular languages. Furthermore, CFGs are

[14]Another way of putting this is that regular languages can count up to a certain point but there is a upper bound on this number and no regular language is able to count two sets of symbols and compare this size if it is potentially infinite.

[15]Constructions which allow for two non-adjacent elements which may contain further instances of the same construction between those elements.

intimately related to phrase-structure grammars. If we take the non-terminals to be syntactic categories and read the arrow as "consists of":

> Therefore the derivation of a string x in such a grammar implicitly imposes a hierarchical structure of x into ever larger sub-phrases. For this reason, context-free grammars/languages are sometimes referred to as phrase structure grammars/languages, and it is assumed that such languages have an intrinsic hierarchical structure (Jäger and Rogers, 2012).

The generating grammar of the class of context-free languages is

$\mathcal{L}(G) = \{w \in T | S \Rightarrow_G^* w\}$ such that $\alpha A \beta \Rightarrow_G \alpha\gamma\beta$ iff $A \to \gamma \in R$ ($\Rightarrow_G^*$ means derivable in a finite set of steps). The class of FSAs associated with context-free languages is push-down automata. The push-down automaton (or stack) incorporates "memory" in a way that the previous FSA could not. The way in which this is executed is that the machine has more options than just state-changing, i.e. an item can be added to or removed from the stack. As we saw in the previous section, patterns such as $a^n b^n$ are not regular but they can be accommodated by context-free grammars/languages as evinced by this push-down mechanism.

Furthermore, in terms of complexity, we still have a manageable resolution of the membership problem. Of course, since the hierarchy is organised in terms of increasing complexity, context-free languages cannot be processed in linear time as regular languages can but rather in cubic time.

The guiding question resurfaces at this point, is this where natural languages are located in the formal language hierarchy? Chomsky seemed to think so. However, there are reasons to think that natural languages are not context-free. It was shown in Shieber (1985), that Swiss German contains cross serial dependencies between verbs and objects which are potentially unbounded in length.[16] Consider the sentence from Shieber:

(2.2.1) dass mer d'chind em Hans es Huus lönd hälfe aanstriiche

that we the children-acc Hans-dat the house-acc let help paint

that we let the children help Hans paint the house

The noun phrase "d'chind" is the argument of "lönd", "em Hans" is the argument of "hälfe" and lastly "es Huus" is the argument of "aanstriiche".[17] The claim is that for a language to satisfy such structures, it would have to allow expressions such as $a^n b^m c^n d^m$ (where $m, n > 0$). Context-free languages cannot accommodate this pattern since they can only capture unbounded *nested* dependencies not serial ones. There is a stronger claim lurking here, namely that context-free grammar cannot generate inflected languages (languages with case) since they can reproduce this pattern. The copy language also has this property. This is the (nonnatural) language generated by $\mathcal{L}(G) = \left\{ww^R | S \Rightarrow_G^* w\right\}$ which states that each string of the language has the form ww. The same pattern of dependencies are generatable in this formal language. Clearly, a less restricted grammar is needed to capture serial dependencies and thus natural language data.

[16]Similar results have been shown for Dutch by Bresnan et al (1982), but these results relied somewhat on formal syntactic theories.

[17]In Shieber (1985) this phenomenon is shown to be robust.

2.2.3 Context-Sensitive Languages (Type 1) and Unrestricted Languages (Type 0)

Although in principle these languages are decidable in finite time, they are often PSPACE hard which renders some of them intractable in practice. Thus, I will not focus on them in much detail here, as it is unlikely that natural language falls within these rings of the hierarchy.

Context-Sensitive grammars have rules of the following type $\alpha \to \beta$ where $\alpha, \beta \in (T \cup NT)$ and $|\alpha| \leq |\beta|$. Thus, if we start with a string w, we can apply the rule backwards finitely many times. The strings that we attain through this procedure are never longer than w itself. This is why context-sensitive grammars specify that the left-hand side of the rule is never longer than the right hand side since if we repeat the procedure of applying the rule backwards we either get shorter strings or loops which are inconsequential for present purposes.

The automata associated with these languages are linearly bounded automata. These are similar to Turing machines, however, they are not infinite. Technically, the finite state automaton contains a tape that is unbounded in length, although only a finite part of the tape, whose length is a linear function of the length of the initial input, can be accessed by the operations during the state transitions.

Given the computational intractability of these grammars, natural language is often assumed to be located somewhere between context-sensitive and context-free. We return to this issue below.

The last ring of the hierarchy is just the set of all formal languages or the computably enumerable languages. Moreover, "it can be shown that any kind of formal, algorithmic procedure that can be precisely defined can also be expressed by some grammar" (Jäger and Rogers, 2012). This is the essence of the "well-definition" of Hockett. Any system that can be defined precisely such as the rules of chess or the derivations of logic or all of the other types of formal languages fall within this ring of the hierarchy. Thus, it is not very useful or interesting for a description of natural language syntax to locate it within the computably enumerable languages. In addition, these languages represent the last level of complexity and are often hard to parse. They can be decided by Turing machines but they are only semi-decidable which means that a machine could run on forever before it decides if a given string is generated by the grammar. Thus, natural language syntax still needs to be located and if it can be located, it is not to be found in the Chomsky Hierarchy.

2.2.4 Beyond the Chomsky Hierarchy: Mildly context-sensitive Languages

It has been proposed that natural languages lie somewhere between context-sensitive languages and context-free languages. The idea behind this is obvious. Context-free grammars are not adequate to capture the range of natural languages while context-sensitive languages are often computationally intractable. Thus, the search for well-behaved formal languages which can capture the range of natural language syntactic phenomena took off in the 80's. The most prominent examples are Tree Adjoining Grammar (to be discussed in Part 3), Combinatory Categorial Grammar (Steedman 2000) and the various forms of Head Grammars (Sag and Pollard 1984). These formalisms have been shown to be equivalent and capable of describing cross serial dependencies in an efficient way ($\mathcal{O}(n^6)$ or in the time of the 6th power to the length of a given string). Thus, these languages are parsable in polynomial time.

It is easy to see why these languages go beyond context-freeness as they are able to capture cross-serial dependencies and even languages such as the copy language. It is only when we consider a certain property of mildly context-sensitive languages that we can appreciate why they are not quite context-sensitive. The property in question is known as the "constant growth property" (cf Joshi 1985). This property states that if we order all the strings of a language according to their length, then the length will grow in a linear fashion. This property also captures the NML property of section 1.3.4. since each sentence can be extended to another grammatical one by adding a single symbol/word. Thus, we have eliminated the context-sensitive languages which lack this property, e.g. the set of square numbers, the set of powers of two and the set of prime numbers.

In summation, we can define this set of languages precisely as

> A set of languages $\mathcal{L}$ is mildly context-sensitive iff (1) $\mathcal{L}$ contains all the context-free languages, (2) $\mathcal{L}$ can describe the copy language and certain cross-serial dependencies of that sort, (3) $\mathcal{L}$ is parsable in polynomial time and (4) $\mathcal{L}$ has the constant growth property.

In part III, we aim to provide a compositional semantic account of a formal language which falls under this purview, namely nonprojective Dependency Grammar. For for moment, however, we will move on to a discussion about why compositionality may be a favourable characteristic for the semantics of a formal language in the first place. But before that, an important clarification.

2.2.5 A Computational Caveat

The above progression through and beyond the Chomsky Hierarchy could be somewhat misleading. It suggests that by producing examples of certain constructions the complexity of which exceed the bounds of a specific ring of the hierarchy, we are forced to move to another more inclusive type of formal language. The claim is then made that natural language could not possibly be contained in that ring of the hierarchy and we should search outside of it. Such claims, although convincing (and ubiquitous), are not necessarily true. The illusion of mathematical certainty in these cases is just that, an illusion. Most of these types of arguments are usually based on a fallacy. However, despite the fact that they may be technically flawed the conclusions drawn are often unchanged (notwithstanding the weaker foundation).

Generally speaking, these arguments are presented in the following way. A construction of complexity p is proffered as part of language NL such that L' is generated where $L' \subseteq NL$ which is in turn at a position R' in the Chomsky Hierarchy. It is then determined that the original language NL must be at least at complexity level p and ring $R \geq R'$. In terms of section 2.2.2 language L' is context-free, so NL must be at least context-free.

This is not strictly the case though, since the complexity of a subset L' of a formal language L doesn't tell us much about the complexity of L itself. Languages of lesser complexity may contain languages of greater complexity as subsets, "a regular language may well contain a subset that is strictly context-free or context-sensitive" (Mohri and Sproat, 2006: 434). For instance, a regular language L could contain the copy language as a subset and still remain regular.

This is not the say that the conclusions of such fallacious arguments are not true. It should just be understood that is not a matter of mathematical proof but rather a suggestive argument in favour of choosing a different formal language type. There are also cases, such as Shieber's Swiss German argument, which are not prey to this fallacy. Specifically, Shieber establishes a homomorphism between verb and noun pairs in cross-serial constructions and then intersects this pattern with a regular language which results in a weakly context-sensitive language since context-free languages are supposed to be closed under homomorphisms and intersections with regular languages.

The purpose of this caveat is to ensure that from the complexity of specific constructions we do not assume that we know the complexity of an entire languages.

2.3 (Good) Reasons for Compositionality

In the analogous section of the previous part, we investigated the common reasons for assuming that natural language has a compositional semantics. It was concluded that these reasons are far from adequate for establishing the principle. Furthermore, they were based on an assumption about the infinitude of natural language expressions which turned out to be unjustified. In this section, I investigate some reasons for why we might want formal languages to have compositional semantics and I conclude that these reasons are much more convincing.

2.3.1 Computational Complexity

From Part I, we know that the validity of the principle of compositionality for natural language is at best an inference to the best explanation. In fact, the most that we do know is that the semantics is computable from the syntax. As we have seen from formal languages such as computer programs, compositional interpretations allow for practical decomposition which is useful for communication of algorithms. The reason for this is that compositional semantics usually involves a minimal amount of processing steps for interpretation, steps which can be easily retraced. The same might apply for online interpretation of natural language constructions. In both cases, compositionality seems to be the simplest method of deriving meanings.

Computational complexity is a central issue in describing and defining formal languages. The Chomsky Hierarchy is arranged in accordance with it and the development of mildly context-sensitive languages was in part prompted by complexity issues. Therefore, complexity should play a role in the interpretation of formal languages as well.

If syntactic structures are to be polynomially parsable then there is some reason to think that they should be interpreted in polynomial time as well.

> If the semantic function is polynomial, as is the case in all standard examples of compositional semantics, the hearer can compute the meaning of a term by means of immediate interpretations of the elements of the term (occurrences of operators or atomic terms). No intermediate computation steps will be needed. Since there is reason to think that polynomial meaning operations are desirable, there is a reason to that that compositionality is a desirable (Pagin and Westerstahl, 2010: 8).

Syntax needs to be easy to interpret if it is to mirror the way in which we actually interpret language. Computability is not a guarantee of tractability, the principle of compositionality is thus needed to improve tractability which is essential for interpretation.

2.3.2 Infinity as a design choice

In previous part, it was shown that infinity is not an empirical observation but a theoretical posit for natural languages. Moreover, this posit was assumed to be the case and never proven so. Natural languages may not be the kind of things that can be called infinite (or well-defined) but formal languages are precisely such things.

The key to this argument is understanding a formal language as a model of certain aspects of a natural language, in this case its syntax. These models involve abstraction over certain other features such as phonology, semantics, performance limitations etc. Specifically, the grammars which we have been discussing can be viewed as formal models depicting our syntactic knowledge.

> The object of modeling is not to gain a perfect representation of an empirical reality, but rather to focus on simpler structures which give useful information about the more complicated reality...Things we can say about the model should give information about the empirical reality, but we should not confuse the model with the reality (Tiede and Stout, 2010).

The modelling step was apparent from section 2.2 in which expressions of a language were taken to be sets of strings and words taken to be symbols. Furthermore, grammars are finite devices used to capture the potential infinity of sequences of sentences. This posit is meant to capture that fact that the human brain is a finite system from which language emerges.

As we saw with CFGs, the inclusion of recursive rules accounted for certain sequences of symbols which could not be dealt with by regular grammars. Recursion has another effect on grammar rules in that it can generate infinitely many expressions from a finite set of symbols. Thus it seems that recursion begets infinity. The problem is that recursion is often explained in terms of infinity. And this leads to the circularity of before.

The major difference this time around is that infinity does not have to be established or derived for formal languages, it can be assumed as a modelling choice. As long as we can justify the choice, we need not worry about using infinite models to describe natural languages.

> Assuming an infinite model does not imply that natural languages actually are infinite, just as modeling natural languages as sets of strings does not imply that natural languages are sets of strings (Tiede and Stout, 2010).

As long as there is independent motivation for positing infinite models, there is nothing unjustified about doing so. Here the natural language properties of productivity, systematicity and novelty come to the fore. Although these properties were not sufficient to conclusively prove the validity of the principle of compositionality for natural language since they assumed linguistic infinity (among other reasons), they are enough to warrant assuming infinity in formal models of natural language.

Consider the case for productivity. If we were to assume a finite model for natural language, then this would establish a least upper bound on the length of sentences. Although performance considerations may suggest that certain lengths are not possible (as in the case of the impossible cricket score in Part 1), there is no fixed upper bound which would be the case if our models were finite.

Returning to mildly context-sensitive languages for the moment, we can even more so appreciate the need for infinite models. It would not be possible to capture the constant growth property in a finite model.

Furthermore, with the explication of formal languages as models of natural languages we have a neat way of capturing the performance/competence distinction as one between natural and formal languages.

2.3.3 Subconclusion

In terms of both the arguments from complexity and the justification of the infinity assumption, it seems that compositional semantics is a legitimate modeling choice for formal languages. Of course, like any modeling considerations it does not mean that natural language interpretation is necessarily compositional. However, it does provide enough justification for adopting a compositional semantics for formal languages used to model natural ones . In the next section, I discuss the various ways in which this can be achieved.

2.4 Formal Definition of Compositionality

2.4.1 Formal Preliminaries

Definition 1. An algebra is a pair $\langle A, F\rangle$ such that A is a nonempty set called the carrier of A and the universe of A and $F = \langle f_i : i \in I\rangle$ are the basic operations on A which are functions defined on A. So i corresponds to each $n - ary$ function in $\mathcal{F}$, i.e. for each $n - ary$ function f in $\mathcal{F}$ there is an $n - ary$ operation f^A in A. The elements of the carrier are called elements of the algebra. For our purposes, we will deal with algebraic signatures, i.e. a signature with no relational symbols. The signature of an algebra is the list of operations on A or a set of names of operations in A. More formally, a signature can be given by a pair $F = \langle F, f\rangle$ which contains the set of functions F and a function f gives the arity of the function by assigning a natural number to every function in F (constants are 0-ary functions).

I will be more precise about exactly what types of algebras we are dealing with in the next section. However, this definition allows us to move to the definition of a core component of the Montagovian notion of compositional semantics, that of a homomorphism between syntactic and semantic algebras.

Definition 2. A homomorphism is defined when there are two algebras $\langle A, *\rangle$ and $\langle B, \circ\rangle$ of the same language (or type) $\mathcal{F}$, and a function $f : A \rightarrow B$. It is a homomorphism if the following two conditions hold.

(1) For all $n - ary$ relations R of $*$, if $\langle a_1, ..., a_n\rangle \in R_A$ then $\langle f(a_1), ..., f(a_n)\rangle \in R_B$ and

(2) For all $n - ary$ operations F, if $F_A(a_1, ..., a_n) = a$ then $F_B(f(a_1), ..., f(a_n))= f(a)$.

If we only deal with algebraic signatures, then (1) is trivially true so we need only concern ourselves with (2).

The homomorphism is not usually defined between the surface syntax algebra and the semantic algebra. But rather between a term algebra and the semantics. Consider the following definition.

Definition 3. A grammar G is usually defined as a *term algebra*. This is a set $T(E, A, F)$ of terms defined in the following way (I add the non-empty set E of expressions such that E is recursively generated from the set of atomic elements A through F to the algebra, for obvious linguistic reasons).

(a) Every variable $\phi, \psi, \eta, \zeta, ...$ is in T and is an atomic term,

(b) Every expression is in T and is an atomic term and,

(c) If α is a syntactic rule of arity n and $t_0, ..., t_{n-1}$ are all in T, then the term "$\alpha(t_0, ..., t_{n-1})$" is in T; it is a complex term and its immediate constituents are the occurrences of the terms $t_0, ..., t_{n-1}$ in it.

We can be even more specific and talk about a grammatical term algebra as a set $GT(E, A, F)$. (i) All atomic expressions are in GT and their values are just themselves, (ii) if α is a syntactic rule of arity n, the terms $t_a, ..., t_{n-1}$ are all in GT and have values $e_0, ..., e_{n-1}$ respectively, and the expression $\alpha(e_0, ..., e_{n-1})$ is defined, then the term "$\alpha(t_0, ..., t_{n-1})$" is in GT and its value is the expression $\alpha(e_0, ...e_{n-1})$.

In addition, there is surjective map val: $GT \rightarrow E$ such that each term maps onto its value.

2.4.2 Standard Compositionality

This definition takes the form of a homomorphism between the syntax algebra or grammatical term algebra and the semantics algebra. Therefore there is a mapping μ for the set of expressions E whose domain is either $GT(E)$ or a subset of it. This is the semantics.

Principle: The meaning of a complex expression is a function of the meaning of its constituents and their mode of combination.

Definition 4. For every rule $\alpha \in F$ there is a meaning operation r_μ such that if $\alpha(u_1, ..., u_n)$ has meaning, $\mu(\alpha(u_1, ..., u_n)) = r_\mu(\mu(u_1), ..., \mu(u_n))$.
This amounts to the syntax algebra $\langle GT(G), \{\alpha_1, ..., \alpha_j\}\rangle$, where $\{\alpha_1, ..., \alpha_j\}$ is the set of basic syntactic operations, being homomorphous to the semantic algebra $\left\langle \mu[GT(G)], \{r_{\alpha(1)}, ..., r_{\alpha(j)}\}\right\rangle$.

We need to say a bit more about the components of this definition formally. We will start with how the notion of a syntax is precisely captured as an algebra. The first point of departure from the strict Montagovian framework is that we will follow Janssen (1986) is viewing the syntax as a multi-sorted algebra[18](There are a number of formal advantages to this approach, for a survey see Janssen (1986:90)). The definition is given below:

Definition 5. $\langle (A_s)_{s\in S}, (F_\gamma)_{\gamma\in\Gamma}\rangle$ is a multi-sorted algebra of signature π iff
(a) S is a non-empty set of sorts;
(b) $(A_s)_{s\in S}$ is an indexed family of sets (A_s is the carrier of s)
(c) Γ is a set of operator indices
(d) π is a type-assigning function which assigns to each $\gamma \in \Gamma$ a pair $\langle\langle s_1, ..., s_n\rangle, s_{n+1}\rangle$, where $n \in \mathbb{N}^+$, $s_1 \in S, ..., s_{n+1} \in S$; and
(e) $(F_\gamma)_{\gamma\in\Gamma}$ is an indexed family of operations such that if $\pi(\gamma) = \langle\langle s_1, ..., s_n\rangle, s_{n+1}\rangle$, then $F_\gamma : A_{s_1} \times ... \times A_{s_n} \to A_{s_{n+1}}$.

As before, A is the set of expressions of the language and F a set of syntactic operations. In general, we do not speak of the meaning of an expression in isolation but in terms of a sort and a specific syntactic derivation. This is often referred to as a derivational history and is captured by the *term algebra.*

> The carriers of term algebras consist of symbols, "syntactic terms," which can be seen as representations of the derivational histories of the generated algebra with which they are associated (Hendriks, 2001).

The reason we concern ourselves with term algebras or derivational histories is to avoid the syntactic ambiguity inherent in natural language. Term algebras are defined as *free algebras* which are syntactically transparent, i.e. permitting no ambiguities. The term algebra is defined in terms of both the syntactic algebra and its generated algebra which is defined below.

[18]Montague viewed it as a one-sorted algebra.

Definition 6. There is a generated algebra $H = (H_s)_{s \in S}$ where S is as before and H consists of noncompound lexical expressions. Thus, given an algebra $\langle (A_s)_{s \in S}, (F_\gamma)_{\gamma \in \Gamma} \rangle$ and set H such that $H_s \subseteq A_s$ for all $s \in S$. Then $\langle [H], (F_\gamma)_{\gamma \in \Gamma} \rangle$ is the smallest algebra containing H and is called the generated (by H) algebra. Furthermore, if $\langle [H], (F_\gamma)_{\gamma \in \Gamma} \rangle = \langle (A_s)_{s \in S}, (F_\gamma)_{\gamma \in \Gamma} \rangle$ then H is the generating set of A and all $h \in H$ are the generators.

The term algebra $T_{A,H} = \left\langle (T_{A,H,s})_{s \in S}, (F^T_\gamma)_{\gamma \in \Gamma} \right\rangle$ is then characterised as a generated algebra with the addition that it is a *free algebra*a as well. To be free in this sense, an algebra needs to fulfill three properties (Hendriks, 2001):

> (1) The members of the generating family $(H_s)_{s \in S}$ are not in the range of some operator F_γ in $(F_\gamma)_{\gamma \in \Gamma}$: if $a_{n+1} \in H_{s_{n+1}}$, then for all F_γ with $\pi(\gamma) = \langle \langle s_1, ..., s_n \rangle, s_{n+1} \rangle$ and for all $a_1 \in A_{s_1}, ..., a_n \in A_{s_n} : a_{n+1} \neq F_\gamma(a_1, ..., a_n)$
> (2) The operators in $(F_\gamma)_{\gamma \in \Gamma}$ are injections that have disjoint ranges: if $F_\gamma(a_1, ..., a_n) = F_{\gamma'}(a'_1, ..., a'_m)$, then $\langle a_1, ..., a_n \rangle = \langle a'_1, ..., a'_m \rangle$ and $F_\gamma = F_{\gamma'}$
> (3) Every member of a member of $(A_s)_{s \in S}$ is a member of exactly one carrier A_s : if $a \in A_s$ and $a \in A_{s'}$ then $s = s'$.

This procedure eliminates syntactic ambiguity. The meaning function applies to the term algebra which contains the syntax algebra and its generated algebra in that they are defined in terms of the carriers of the term algebra. In fact given this, the meanings will constitute a multi-sorted semantic algebra in accordance with the syntax. In general, this procedure involves defining semantic types which mirror the syntactic types such that elements of the same syntactic types are assumed to have the same semantic type. The universe of the semantic function is the indexed family of sets $(B_t)_{t \in T}$ where T is the set of semantic types. To cement the relationship between syntactic and semantic types, there is a function f such that for all $s \in S$, $f(s) = t$ for some $t \in T$. There is also a corresponding indexed family of operators R_μ which act as a semantic function taking the meaning of every syntactic operation in F_γ as input and outputing the meanings of the components such that

$$\mu(F_\gamma(a_1, ..., a_n)) = R_\mu(\mu(a_1), ..., \mu(a_n))$$

Therefore, compositionality amounts to three key components, (1) the syntax is viewed as an algebra $\langle (A_s)_{s \in S}, (F_\gamma)_{\gamma \in \Gamma} \rangle$ and a generated algebra $\langle [H], (F_\gamma)_{\gamma \in \Gamma} \rangle$ (defined as before), (2) the semantic algebra $\langle (B_t)_{t \in T}, (R_\mu)_{\mu \in M} \rangle$ whose domain is a subset of the term algebra $T_{A,H} = \left\langle (T_{A,H,s})_{s \in S}, (F^T_\gamma)_{\gamma \in \Gamma} \right\rangle$ and (3) the meaning assignment is a homomorphism between $T_{A,H}$ and B.

One important fact about compositional semantics (or at least versions inspired by Montague) is that it requires an intervening formal language. Janssen puts the point thus:

> However, one always uses, in practice, some formal (logical) language as auxiliary language, and the language of which one wishes to define the meanings is translated into this formal language. Thus the meaning assignment is performed indirectly (1986, Part 1: 81).

In most cases, this is more than just a practical matter and the principle of compositionality is applied to this formal language as a proxy for natural language. This is part of the reason that compositionality could be a valid methodological principle for formal languages and natural language could, in fact, be noncompositional without loss of consistency. The compositional treatment of such an intervening language is the main purpose of Part III of this research.

2.4.3 Strong Compositionality

The principle as it is stated above is not a very strong constraint on the semantics of a language. Stronger versions take the form of constraints on the semantic function. In other words, the more limitating requirements which are placed on the semantic function the stronger the semantics. To show that this is an, in fact necessary, exercise consider that there is a simple identity mapping which respects the initial definition and is completely vacuous.[19]

> Every syntax may serve as a semantics for itself. For, in that case we have an isomorphism between syntax and semantics and consequently the homomorphism required by the principle of compositionality is warranted (Werning, 2005: 288).

The solution adopted in places is to follow Werning (2005) in adding the property of *non-hyper-distinctness* to the definition. That is given a grammar G and a set $GT(G)$ as before, there is a meaning function μ whose domain is $GT(G)$ which is non-hyper-distinct if there are grammatical terms $s, t \in GT(G)$ such that $s \neq t$ and $\mu(s) = \mu(t)$. In other words, our semantics not only blocks vacuity but now allows for synonymous expressions.

There are other such constraints on compositionality which also serve to strengthen the principle. For instance, Hodges identifies the 'Husserl property' which places a requirement of sameness of semantic category on sameness of meaning. As Hodges (1998:15) puts it

> (Husserl's principle) If two expressions have the same meaning then they have the same category.

There is some linguistic motivation for such a constraint since it is difficult to find expressions of natural language in which words of different categories can be substituted for one another *salve veritate*. Consider the following pairs of examples from English:

(2.4.1) John was thesis supervisor in the linguistics department.
* John was thesis supervising in the linguistics department.

(2.4.2) John was fast asleep after the earthquake.
John was fast sleeping after the earthquake.

[19]The general outcome may be stated roughly as 'anything goes' - even though adherence to the principle [of compositionality] often makes for elegance and uniformity of presentation. [...] we are entitled to conclude that by itself, compositionality provides no significant constraint upon semantic theory (van Benthem, 1984: 57).

There a slight difference in meaning in both sets of sentences, an ubiquitous phenomenon when the categories of words are changed. 2.4.1 marks the difference between a permanent property and a temporary one, while 2.4.2. contrasts a state with a process. Evidently this phenomenon is cross-linguistically verifiable, observe the Russian pair below:

(2.4.3) Džon byl rabotnik medlennyj
John was a slow worker.
Džon rabotal medlenno
John was working slowly.

A good semantics should be able to track the change of meaning in such cases. Adopting the Husserl principle is one way of achieving this goal.

There are also more formal constraints on compositional semantics. For instance, the "similarity of algebra" constraint suggested by Janssen (1986).

> A mapping is called a homomorphism if it respects the structures of the algebras involved. This is only possible if the two algebras have similar structure (Part 1: 21).

To some this constraint is too strong and leads to complications. One such complication is brought out by Hendriks (2001).[20] The similarity constraint requires that there are bijections from both the syntactic categories to the semantic types and from the syntactic operation indices to the semantic ones. However, in practice, these bijections do not always hold. Consider the semantic type $\langle e, \langle \langle e, t \rangle, t \rangle \rangle$ which denotes a two-place relation between individuals and first-order predicates such as can be found in phrases like "is a property of". It is not clear that this type can be straightforwardly linked to a syntactic category, thus the mapping from categories to types is not surjective. It can be shown to be non-injective as well in the cases where different syntactic categories correspond to the same semantic type. The same reasoning applies to the operations of the algebra. Consequently, this constraint is too strong.

There are, of course, many other constraints that one could place on the semantics such as computability and complexity constraints etc. Some can be more consistently motivated than others. The purpose of this section is not to provide a comprehensive overview of these possibilities but merely to indicate what a stronger compositionality principle entails.

2.4.4 Direct Compositionality

There is another debate, within the larger debate about compositionality, that has recently emerged and that debate concerns whether or not the principle involves a *direct* conception of compositionality. Direct compositionality pertains to a semantic analysis that requires that every constituent in a given expression receives a semantic value. This blocks various ways of underspecifying semantics such as Cooper Storage. These are typically strategies employed to delay the semantic analysis in an effort to resolve certain ambiguities or anaphoric and

[20]In section 2.4.6. I adopt a slightly weaker formulation of it.

scopal exigencies. But for a simple case of non-direct compositionality, consider the case of quantifier scope contrual in standard semantics:

> a verb phrase such as *saw everyone* fails to have a semantic interpretation until it has been embedded within the large enough structure for the quantifier to raise and take scope (e.g. *Someone saw everyone*) (Barker and Jacobson, 2007: 2).

Generative linguistics assumes a level of logical form in which interpretation is conducted, this too is not directly compositional. The bottom-line is that in directly compositional accounts, every single constituent needs to have a semantic value even before introducing further context. The way in which this is achieved is through the strong requirement that for every syntactic operation there is a corresponding semantic operation. This means that every syntactically computed expression has a meaning, i.e. no distinct level of representation through which interpretation is conducted. Other (popular) ways of putting this is that the syntax and semantics operate in tandem.

There are two ways to look at direct compositionality. One way is to view it as a strengthening of the principle along the lines of the other proposals in the previous section. This is the less interesting view. The other way of viewing this version of compositionality is as the intended account of what is meant by compositional semantics.

The rationale for this interpretation of the principle is that it involves the simplest conception of grammar and that it contains actual insight into human sentence processing. The strategy for proving this is usually to shift the onus to those objecting to the direct interpretation. The first argument runs roughly as follows (courtesy of Jacobson (2012)):

1. Any theory of natural language should have a compositional syntax (a recursive procedure for proving the grammaticality of expressions)

2. Any theory of natural language should have a compositional semantics (a system for predicting the meaning of larger syntactic structures from smaller ones)

3. The simplest way to satisfy both (1) and (2) is a version of compositionality which makes the two systems work in tandem and not separately.

Of course, one could dispute whether either or both of the premises are valid but the real question is assuming these premises does (3) follow? The answer is no. It is very hard to employ Ockam's Razor in the way that Jacobson wants to do so here, i.e. *in abstacto*. The matter is not as simple as comparing direct and indirect theories of compositionality. It has to be shown that in specific places where indirect methods are typically used for semantic analysis of phenomena, the same results can be achieved with direct analysis. Indeed, the motivation in most cases for indirect tactics is on a case by case basis. Thus, in certain cases the direct analysis might not be the simplest one. Therefore, I am not sure there is a legitimate application of simplicity considerations here. To illustrate this point, consider Montague Grammar. It contains 'Quantifying-In Rules' and syntactic substitution rules for anaphoric reference, these are indirect methods of providing semantic values. Yet on the whole, Montague Grammar is still considered to be directly compositional since there is a mapping between the syntactic and semantic rules ensuring that every expression has a meaning.

> We can call any theory which relaxes the prohibition against reference to internal structure in the syntax a theory with "weak direct compositionality" (Barker and Jacobson, 2007: 4).

This gradation do not make for a very strong case for global strong direct compositionality. If we are allowed to "relax the prohibition" on internal structure in some cases then what stops us from creating a representational level in order to resolve various syntactic concerns which we encounter and interpret others directly.

The second reason advanced in favour of direct compositionality is that it better accounts for language-users tendency to parse sentences incrementally. Garden-path phenomena among other things lead us to believe that this is indeed the case, at least for the most part. This seems like a more compelling line than the previous one. Yet it doesn't decide which brand of direct compositionality is best (Jacobson defines four) and the choice is non-arbitrary. Opting for one type as opposed to another could allow for more or less internal structures in the syntax (as we saw with Montague Grammar). In addition, levels of representation have been used in attempts at more cognitively realistic architectures for natural language processing and interpretation, consider Discourse Representation Theory. Nothing in the fact that human beings parse sentences incrementally necessitates that this process is done directly or compositionally for that matter.

At best, direct compositionality will be construed as a strengthening of the principle in its more common form with the amendment that a compositional semantics ought to be at least weakly directly compositional in some form.[21]

2.4.5 A Formal Description of Constituency

Providing a formal definition of constituency is not as obvious as one might expect. Various theories offer differing views of the notion. Is every part of a sentence a constituent? Or are constituents specific linguistic objects the nature of which is capable of precise definition? What counts as a constituent in a formal language? In what follows, I offer answers to these questions.

The mereology of the principle of compositionality is an important topic. I claim that the part-whole relation is what the principle is based on. In its most abstract form, the meaning of an expression is the sum of the meanings of its part and the way in which they

[21] It should be noted that in its strong form, direct compositionality places a very powerful constraint on the semantics. Not only in the case of scope contrual as mentioned before but elements such as particles and pronouns will have to be given immediate interpretations as well. Consider the English sentence *There he is, the man you are looking for.* The pronoun *he* would need to receive a meaning prior to and independently of its reference being determined. Then consider the Dutch sentence *Op de kamer is er een douche* (there is a shower in the room) where the use of the particle *er* is not an obvious semantic contribution.

are combined. Thus, the principle must involve a notion of meaning in terms of parthood such that another way of describing it is in terms of WC below

> WC : The meaning of the whole is the sum of the meaning of its parts and their method of combination

The picture of constituency that naturally emerges is that of a constituent being identical to a *part*. If this is the case, then one could define constituency in terms of the definition of parthood. In mereology (at least in accordance with Lesniewski), parthood is generally taken to have three formal properties:

1. Irreflexivity: $\neg Pxx$. Nothing is a part of itself.
2. Asymmetry: $Pxy \rightarrow \neg Pyx$. If x is a part of y, then y is not a part of x.
3. Transitivity: $(Pxy \wedge Pyz) \rightarrow Pxy$. If x is a part of y and y is a part of z then x is a part of z.[22]

According to this rationale any part of an expression would be a constituent of that expression. This quickly runs into triviality. Consider the sentence below:

(2.4.4) A man walked into the room with nothing good on his mind.

The way in which this would generally be compartmentalised into constituents in a neutral (as possible) theory of syntax is something like

> [a man walked][into the room][with nothing good][on his mind]

Ignoring hierarchical structure and relationships between these syntactic objects for the moment, we can see that the three properties are parthood are respected in this case. Which lends credence to the claim that constituents are just parts. But a problem arises when we realise that parthood doesn't end here and there are more elements of the sentence which fall within its definition, namely

> [a] [man] [walked][into] [the] [room][with] [nothing] [good][on] [his] [mind]

This situation is perfectly acceptable for a notion of parthood but it reduces the idea of constituency to naught.[23] You would be hard-pressed to find a linguist who holds that every part of an expression is a constituent. Furthermore, there are constituentless grammar formalisms (such as the one which we will discuss in Part III) which very obviously contain parts. Clearly there is something more to constituency than just parthood. In addition, the principle of compositionality cannot be reduced to the WC statement.

[22]In some literature, parthood is defined as a partial ordering, i.e. reflexive, antisymmetric and transitive. This allows a part to be a part of itself which when viewed from the point of view of set theory seems to invite inconsistencies.

[23]Of course, we could call these elements subparts but there is no principled distinction to be drawn between these elements and the elements before unless we smuggle in some intuitive notion of constituency.

Moreover, if a statement such as WC amounts to the claim that the whole is the *mereological sum* of its parts and their method of combination, then there is further evidence against the reduction. In Classical Mereology, an object a is the *mereological sum* of all the elements of some set X iff every element of X is an ingredient of a and every ingredient of a overlaps with some element $x \in X$, where *ingredients* and *overlap* are defined as follows:

1. x is an ingredient of y iff x is a part of y or $x = y$

2. x overlaps with y iff there is a z such that z is an ingredient of x and z is an ingredient of y

The problem is that meaning cannot be equivalent to mereological sum since the right hand side of the biconditional is not satisfied in general. To see this, let object a be the "meaning of the whole expression" and let the set X be "the meaning of the parts of the sentence". In many cases, the meaning of every element of the meaning of the parts of the sentence is not an *ingredient* of the meaning of the whole expression. In addition, every element of the parts of the total meaning needs to overlap with some part of the meaning of the parts of the sentence. Idioms are obvious counterexamples to this but notwithstanding idiomatic expressions (perhaps they can be treated as units), in many cases not all of the elements of sentences percolate up to the semantics of complex expressions, e.g. dummy verbs, particles etc.

Constituents seem to be "natural" groupings of linguistic material which act as units during syntactic processes. For instance, they can allow for movement and deletion,

(2.4.5) Into the room, a man walked with nothing good on his mind.

(2.4.6) A man walked ~~into the room~~ with nothing good on his mind

Whereas **Man a walked room into the with nothing good on his mind* doesn't work and **A man walked ~~into~~ the room with nothing good on ~~his~~ mind.* There is something intuitive about the groupings of words in a sentence, an intuition which guided a lot of early Phrase Structure Grammar. This is what any formal definition of constituency has to capture. Hodges (2012) offers such an account based on Bloomfield and Chomsky. It is given below.

Definition 7. Let a *constituent structure* be an ordered pair $(\mathbb{E}, \mathbb{F})$ where set $\mathbb{E}$ denotes the expressions and the elements of $\mathbb{F}$ are called the frames. This is the case if the following four conditions hold (expressions are denoted by e, f and frames are given by $F, G(\xi)$ etc).

1. $\mathbb{F}$ is a non-empty set of partial functions on $\mathbb{E}$.

2. Nonempty composition: if $F(\xi_1, ..., \xi_n)$ and $G(\eta_1, ..., \eta_m)$ are frames, $1 \leq i \leq n$ and there is an expression $F(e_1, ...e_{i-1}, G(f_1, ..., f_m), e_{i+1}, ..., e_n)$, then $F(\xi_1, ..., \xi_{i-1}, G(\eta_1, ..., \eta_m), \xi_{i+1}, ..., \xi_n)$ is a frame.

3. Nonempty substitution: if $F(e_1, ..., e_n)$ is an expression, $n > 1$ and $1 \leq i \leq n$, then $F(\xi_1, ..., \xi_{i-1}, e_1, \xi_{i+1}, \xi_n)$ is a frame.

4. Identity: there is a frame $1(\xi)$ such that for each expression e, $1(e) = e$.

This definition should allow us to define constituency precisely in the following way that respects its divergence from parthood and its idiosyncrasies without committing us to a specific theory of syntax.

Definition 8. An expression e is a *constituent* of an expression f if f is $G(e)$ for some frame G; e is a *proper constituent* of f if e is a constituent of f and $e \neq f$.

The notion of constituency seems to be captured by the definition but it depends to a large extent on the type of constituency posited by the language L. How we choose to fix the constituency structure for a language depends on the syntactic theory which we employ. Frames are (partial) functions which take expressions as input and yield expressions as output. A constituent is an expression which is contained in a frame of another expression and the frames are objects which allow for composition and substitution. This dovetails with our linguistic intuitions about constituents allowing for movement and deletion etc. It also gives us a good basis for distinguishing grammar formalisms which lack this grouping structure from those which possess it. Finally, we are in a position to define the principle of compositionality as a principle not just based on the vague term "component" as we did in Part I, but as a principle concerning expressions and their constituents,

> There is a meaning function μ such that for some expression f, $\mu(f) = \mu(e_1), ..., \mu(e_n)$ for all e iff e is a constituent of f and f is $F(e_1, ...e_{i-1}, G(e_{i+1}, ..., e_n))$.

In the next section, we discuss associated guiding principles which should be considered when providing for an account of the compositional semantics of formal and natural language.

2.4.6 The Principle of Syntactic Integrity

Formal syntax is an essential part of the homomorphism and thus the compositionality definition. The syntax of the formal language which we choose is what feeds directly into the relationship with the semantics such that the concept of compositionality can be applied. Thus, besides the requirements placed on the syntax by the formal algebraic concerns of the homomorphism there are other linguistically motivated principles which a good compositional theory of meaning should follow.

These principles can help guide the formation and evaluation of various proposals of compositional semantics for specific formal languages or grammar formalisms. In most cases, the syntax employed in a theory of compositional semantics is relatively naive. Thus, syntactic considerations are often guided by semantic ones. For instance, most theories of compositional semantics assume a context-free grammar for the syntax. Dowty (2006) criticises this tradition and offers insights into a novel methodology for compositional semantics, one which approaches a particular linguistic problem in three independent ways. Firstly, it considers the syntax-semantics interface which the problem entails independently of the syntax which is to be evaluated in purely syntactic terms and the semantics which is to be evaluated likewise. This paradigm shift advocates that results in natural language syntax and semantics share in each others advances and thus not develop with biases based on unconsidered assumptions. Towards these three goals, Dowty proposes three corresponding principles:

Compositional_Transparency: This is the degree to which a compositional semantics is obvious, simple, and easy to compute from a given syntactic structure

Syntactic_economy: This is conceived of as the degree to which the syntax is efficiently and easily (i.e. only as complex as necessary) formed in order to produce meanings compositionally

Structural_Semantic_economy: This is the semantic version of the previous principle, i.e. that the meaning operations used to build the meanings of complex expressions from the meanings of less complex constituents in the simplest way as possible.

These principles are to be viewed as useful criteria upon which to judge various theories of compositional meaning. All these criteria heavily favour simplicity over complex structures whenever possible. For example, in terms of the direct compositionality thesis, these principles can guide our decisions about whether certain linguistic constructions warrant directly compositional treatment or if the simplest solutions involve more complex semantic structure.

However, these principles although useful still manage to neglect a certain feature which I believe to be essential to any compositional analysis of a formal language and a natural language via such a formal language, namely that the semantics needs to respect the syntactic structure of the grammar formalism or formal language in question. I believe that this is in the spirit of the similarity constraint of Janssen discussed in section 2.4.3. It forces a certain sort of similarity between the syntax and semantics (and the associated algebras) without imposing the problematic strong bijection requirement between elements and operations of the respective algebras. In light of these considerations, I propose the following principle,

Syntactic_Integrity: The degree to which the semantic structures align with the syntactic structures to which they provide interpretations.

This is still quite vague. The idea is brought out in practice in section 3.4 but I will adumbrate the point here. For a simple example of this principle in action, consider Categorial Grammar. Let us assume that it contains an operation δ which concatenates two expressions a and b iff a belongs to sort $s \in S$ and b belongs to $s' \in S$ such that $s \neq s'$ and either s or s' is a primitive syntactic sort. The semantics for such a fragment would need to respect the syntactic combination such that the semantics composes two nonidentical elements at least one of which is a primitive semantic type, unless such an analysis comes at the cost of consistency or complexity.

The way in which I think this idea of guiding principles should be viewed is as a tableaux in Optimality Theory (OT). OT provides a framework for deciding between rival analyses of various natural language phenomena. Initially, conceived of as a tool for accessing phonetic structure, it is now widely used across linguistics within syntax, semantics and pragmatics. A standard tableaux is comprised of a violable ranked set of constraints (from a universal set of such constraints) and a candidate set of possible interpretations/phonetic structures/syntactic analyses etc generated by GEN (which is the set of such candidates) for a given input. The constraint set selects the optimal candidate from the set generated by GEN. An illustration of the procedure is given below for the ranking *Compositional Transparency, Syntactic Integrity*$\gg$*Syntactic Economy, Structural Semantic Economy*.[24] Different rankings will result in different optimal outcomes. In certain cases, there will be more than one optimal solution.

Syntax	Semantics	Comp.Trans	Syn.Int	Syn.Eco	Sem.Eco
Σ	μ_1	*	*		*
	μ_2	*		*	
$\Rightarrow$	μ_3			**	

2.5 Conclusion

In this part, we have shifted the focus from natural language to the formal languages we use to model its nature. The theme has been largely devoted to syntactic considerations. We then used these considerations in a formal description of the principle of compositionality. Modifications and clarifications were presented with relation to the principle. A key feature of its definition, namely constituency, was discussed and formalised. Finally, I offered another principle toward the aim of guiding the compositional semantics of a given formal language or grammar. The principle of syntactic integrity should serve as a guiding principle for the formation of the semantic analysis of the next part and the evaluation of alternatives already on the market.

[24]The symbol "$\gg$" denotes priority and "," denotes equal priority.

Part III

A Compositional Semantics for Dependency Grammar

In this part, we explore the grammar formalism known as Dependency Grammar (DG). First, I will outline the nature of this formalism and compare it to other formalisms such Context Free Grammar (CFG) and Tree Adjoining Grammar (TAG). Next, I will describe some of the advantages of this "constituentless" way of dealing with natural language syntax.

I then define a universal algebra to generate dependency structures and I attempt to provide an analogous semantic algebra both of which constitute a homomorphism between the term algebra of the former and the type-algebra of the latter. In terms of this homomorphism, I offer a rule-to-rule semantic analysis which takes dependency structures to semantic representations which are in turn interpreted in Montague Grammar.

3.1 Dependency Grammar

3.1.1 Elementary Structures and Constituency

Phrase Structure Grammar

A good tool for grasping the nature of a given formalism is to consider its elementary structures and basic operations. An additional aspect of these formalisms which is pertinent to the current discussion is constituent structure. For instance, if we are dealing with Phrase Structure Grammars or CFGs, the elementary structures are the nonterminal and terminal alphabets. The basic operations are the set of rewrite rules which define nonterminals in terms of combinations of terminals and nonterminals. Consider the following tree diagram in figure 3 below:

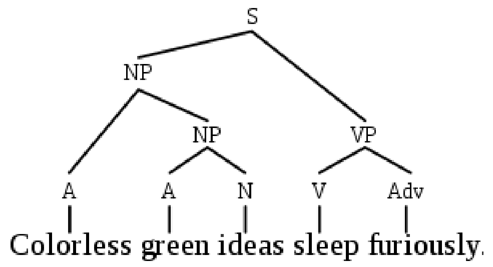

Fig.3

The basic operations or rewrite rules can be simply read off the tree. There are four such rules used to generate this tree.

1. $S \to NP, VP$
2. $VP \to V, Adv$
3. $NP \to Adj, NP$
4. $NP \to Adj, N$

Formally, phrase structure grammars are often modelled in terms of a quadruple $PSG = \langle N, T, P, S \rangle$ where N and T are the nonterminals and terminals respectively, P is the set of production rules (as above) and S is a unique start symbol such that $S \in N$. The elementary structures are N and T while P is the set of basic operations. The relationship between this grammar and the trees is one of derivation.

> Derivations in CFGs can be represented as trees: for each nonterminal node in the tree, the daughters record which rule was used to rewrite it (Rambow & Joshi, 1994:3).

Finally, a constituent in this type of grammar is that which falls on the right side of the arrow in a rewrite rule. In fact, this picture is quite misleading since constituent structure is meant to be captured by these rules and are not an offshoot of them. Thus, the constituents of a sentence are the phrases of which it is comprised. This is the crux of the Chomskyan paradigm in linguistics.

Tree Adjoining Grammar

Let us consider another formalism and its elementary structures, namely Tree Adjoining Grammar (TAG). TAGs were introduced to solve some of the problems of Tree Substitution Grammar (TSG). In the previous section, we considered trees to be derived objects, TSG is what we get when we take trees to be the elementary structures of the grammar. Below are three such structures (from Rambow and Joshi, 1994).

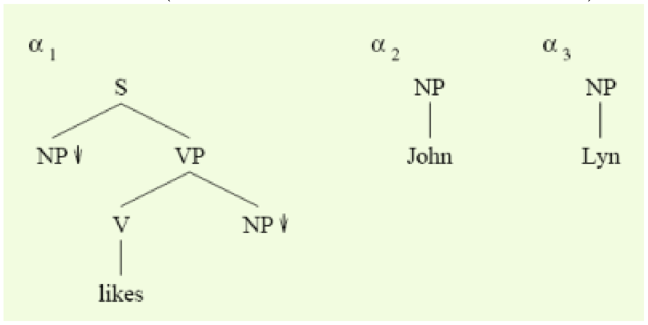

Fig 4.

There is only one basic operation in this grammar and that is substitution. This operation is indicated by the downward arrow in Figure 4. The NPs in α_2 and α_3 can be substituted into the structure in α_1 at any of the nodes marked with $\downarrow$. CFGs and TSGs are weakly equivalent. Unfortunately, TSGs have trouble generating certain iterative constructions prevalent in natural language, such as adverbial modification. Given the example above, it is not possible to modify the verb *likes* as there simply is no node in which to insert such a modifier. In the corresponding PSG this is a simple matter of adding the rewrite rule $VP \rightarrow adv, VP$ which can be iterated. This option is not available for the TSG.

Issues such as these led to the development of TAG. By adding another operation called adjunction to the grammar we can account for more natural language phenomena. The new operation is shown below.

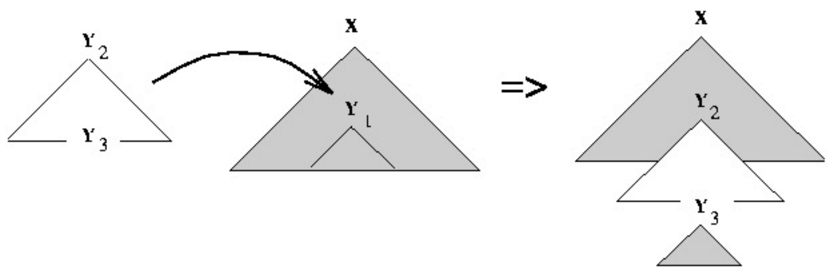

Fig 5.

The elementary structures remain the same between TSG and TAG. The only difference formally is that another basic operation has been added to the grammar. This operation allows nodes to be inserted internally into trees thereby solving the adverb insertion problem. However, adjunction enables TAG to go beyond the boundaries of CFGs. One consequence of this is that:

> TAGs are more powerful formally than CFGs, meaning that they can derive more complex languages than CFG. They are also more difficult to parse (Rambow and Joshi, 1994:6).

Context-sensitive grammar parsing is PSPACE-complete. However, this isn't such a serious concern since both CFGs and TAGs have the property of being able to be parsed in polynomial time. This moves TAGs out of the realm of Context Freeness in the Chomsky Hierarchy and into the area between this and context-sensitivity, i.e. mildly context-sensitive. "The class of string languages generated by TAGs contains the copy language, but unlike context-sensitive grammars, TAGs can be parsed in polynomial time" (Debusmann & Kuhlmann, 2008:6). We will discuss some advantages of moving away from context-free grammars for a correct description of natural language syntax in 2.2.4.

3.1.2 Lexicalization and Dependency Grammar

The lexicalization of a grammar involves associating every elementary structure with a lexical item or terminal node. More specifically, "we will call a grammar "lexicalized" if every elementary structure is associated with exactly one lexical item, and if every lexical item of the language is associated with a finite set of elementary structures in the grammar" (Rambow and Joshi, 1994:5). For instance, lexicalizing CFGs lead to TSGs. This precisely amounts to the claim that:

> If the underlying grammar is lexicalized, then there is a one-to-one correspondence between the nodes in the derivation tree and the positions in the derived string: each occurrence of a production participating in the derivation contributes exactly one terminal symbol to this string (Debusmann & Kuhlmann, 2008:6).[25]

Dependency grammar differs from the grammar formalisms discussed above in that it is already lexicalised since the elementary structures are the lexical items or more accurately

[25]This is how you can induce certain dependency structures from CFGs.

nodes labeled with terminal elements (anchors). This way of doing syntax was originally conceived of by Tesniere (although it shares certain aspects with pretheoretical insights into grammar):

> The sentence is an *organised whole*; its constituent parts are the *words.* Every word that functions as part of a sentence is no longer isolated as in the dictionary: the mind perceives *connections* between the word and its neighbours; the totality of these connections forms the scaffolding of the sentence. The structural connections establish relations of *dependency* among the words. Each such connection in principle links a *superior* term and an *inferior* term. The superior term receives the name *governor* (regissant); the inferior term receives the name *dependent* (subordonne) (1959).

From the above quotation we can glean the general structure of dependency grammar. We have already identified the elementary structures as nodes with terminal symbols (words) attached and the basic operation is governed by the dependency relation between words. The basic idea is that every word except the root in a sentence depends on another word. It helps to view this in more formal terms.

Traditionally, there are four essential properties to a dependency formalism.

Proposition 9. *$DG = \langle R, L, C, F\rangle$ such that:*
R is a set of dependency rules over auxiliary symbols C
L is a set of terminal symbols (lexemes or words)
C a set of auxiliary symbols (lexical categories)
F is an assignment function such that $F : L \to C$. (Hays 1964 and Gaifman 1965)

Unlike Context free grammars (CFG), Dependency grammars (DG) do not possess a non-terminal alphabet or a set of rewrite rules (although projective DG can be shown to be weakly equivalent to CFG). This produces a more flat structure to the syntactic analysis (see next section).

Proposition 10. *The set of rules are threefold:*

(1) $x(w_1, ..., *, ..., w_k) : w_1, ..., w_k$ are dependent on x.
(2) $x(*) : x$ is a leaf node
(3) $*(x) : x$ is a root node.

However, more needs to be said about the dependency relation DR. It is a proper subset of the product of the set of words or terminal alphabet and itself, i.e. $DR \subset W \times W$. It has three main properties or axioms (according to Robinson (1970)):

Proposition 11. *(1) Acyclicity: $\forall w_1 w_2 ... w_{k-1} w_k \in W : \langle w_1, w_2\rangle \in DR ... \langle w_{k-1}, w_k\rangle \in DR : w_1 \neq w_k$*
(2) Rootedness: $\exists ! w_1 \in W : \forall w_2 \in W : \langle w_1, w_2\rangle \notin DR$ and
(3) Single-headedness: $\forall w_1 w_2 w_3 \in W : \langle w_1, w_2\rangle \in DR \wedge \langle w_1, w_3\rangle \in DR \to w_2 = w_3$ (Debusmann, 2000).

From these properties, we can show that R is both irreflexive and asymmetrical. These properties were conceived of as treeness constraints on dependency graphs by Robinson (1970).

3.1.3 Dependency Structures

We now need to characterise the DG algebra such that it reflects the dependency structures generated by these properties. To sum up, (1) no word should depend on itself (not even transitively). (2) each word should have (at most) one governor, and (3) the dependency analysis should cover all words in the sentence.

Definition 12. A dependency structure can be identified as a labeled directed graph $G = \langle V, E \rangle$, with V as a set of nodes such that $V \subseteq \{w_0, w_1, ..., w_n\}$. E is a set of edges often labeled in terms of roles, so $E \subseteq V \times V \times Rol$. The dependency relation can then be defined for all $i, j \in V$ such that $i \to j \equiv (i, j) \in E$ and there is a path $i \to^* j \equiv i = j \vee \exists k : i \to k, k \to^* j$. A simple example of a directed graph is given below (fig 6).

The set of all such structures is given by DS. Furthermore, a DS structure can either be saturated or unsaturated, the set DS^+ is the set of saturated dependency structures i.e. all the nodes have the required values in the graph, and similarly the set DS^- is that set which consists of all unsaturated dependency structures with required nodes which are yet to be fulfilled. For instance, given a graph with a root node which is a transitive verb, it needs to have both an edge going to a subject noun and one going to an object noun, if either of these nodes are missing in the structure then that structure is unsaturated.

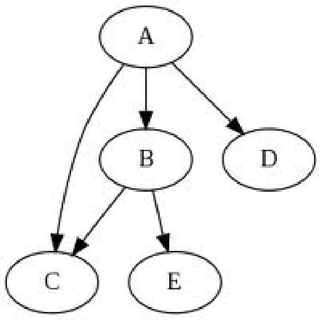

Fig. 6

Below are examples of various ways of representing dependency graphs. (a) is closest to the description above (without role labels).

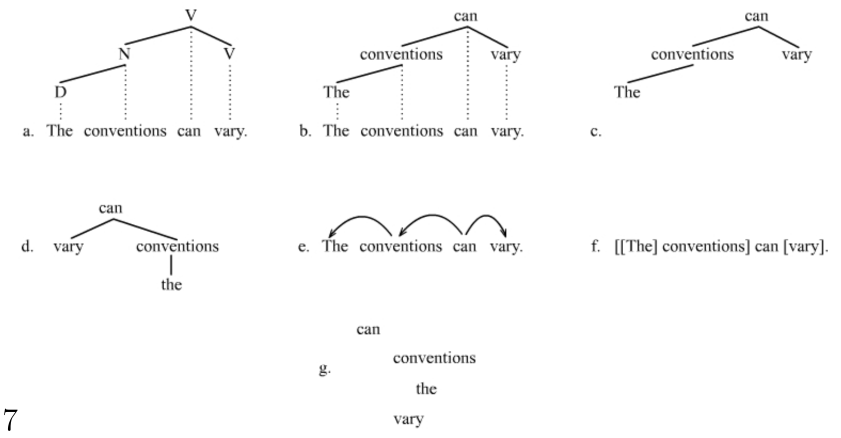
V
N
V
D
a. The conventions can vary.
can
conventions
vary
The
b. The conventions can vary.
can
conventions
vary
The
c.
can
d. vary
conventions
the
e. The conventions can vary.
f. [[The] conventions] can [vary].
can
g.
conventions
the
vary

Fig. 7

A labeled dependency analysis can be rendered as in figure 8 with labels either on the arrows themselves or below. The literature contains quite a few compatible representations of dependency analyses.

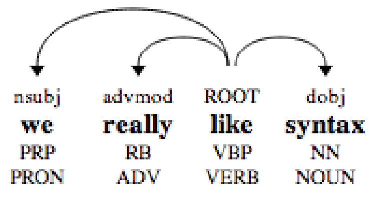

Fig. 8

3.1.4 Projective and Non-projective structures

I have not yet mentioned the last of Robinson's axioms which describes a condition called projectivity. The issue over whether or not DG requires this condition is controversial. For Robinson, a DS is only well-formed (or saturated in our sense) if it respects this condition:

Definition 13. Projectivity: If A depends directly on B and some element C intervenes between them (in the linear order of the string), then C depends directly on A or B or some other intervening elements (Debusmann, 2000).

In terms of directed graphs, this condition blocks crossing edges. Robinson compares the condition to the non-discontuity of constituents in immediate-constituent theory. The problem is that certain languages seem to require crossing edges in their analysis, especially the so-called free word order languages. Consider the German sentence below (from Duchier ESLLI 2000):

(3.1.1) Das Buch hat mir Peter versprochen zu lesen
The book has me Peter promised to read.
Peter promised me that he would read the book.

The DS for this sentence looks as follows.

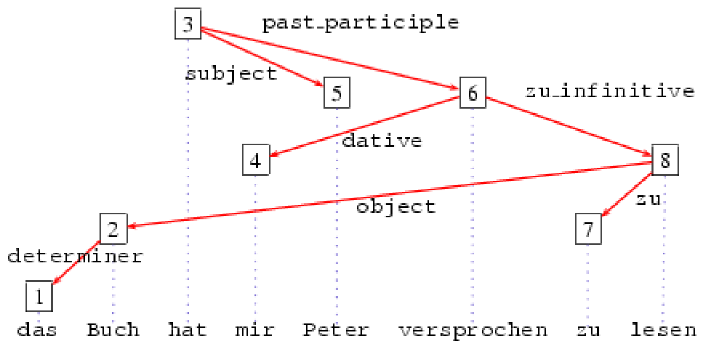

Fig. 9

Crossing edges are needed to account for such languages. Furthermore, from the work of Bresnan (1982) and Shieber (1985) we know that CFGs (even lexicalised ones) are not adequate for capturing the cross-serial dependencies of Dutch and aspects of Swiss German syntax (cf Section 2.2.4). Since projective dependency grammar is weakly equivalent to CFGs (or can be induced by lexicalised CFGs), a result proved by Hays (1964), we know that they too are inadequate for describing certain natural language phenomena.

Consider the following Russian sentence:

(3.1.2) Ya prishyel domoy, koguda oozye bilo tyemno

I came to home, when already it was dark

It was already dark when I came home.

This sentence is equivalent to the sentences below:

(3.1.3) Koguda ya prishyel domoy, bilo oozye tyemno

(3.1.4) Koguda bilo oozye tyemno, ya prishyel domoy

(3.1.5) Prishyel domoy ya, koguda bilo oozye tyemno

However, nonprojective dependency structures lead to increased complexity and parsing problems. “Insofar as computational linguistics is concerned, these rules [projectivity, scrambling etc.] are like any other rules involving transformations - they become unfeasible if one proceeds to implement efficient parsing algorithms” (Debusmann, 2000: 9). This is not necessarily devastating as nonprojective dependency structures fall into the camp of Mildly Context-sensitive formal languages discussed in Part II and there are efficient parsers for such languages. It is, however, quite beyond the scope of the current work to evaluate these issues further. Suffice to say, that any syntax algebra which aims at generating the gamut of dependency structures should account for both projective and nonprojective structures.

3.1.5 Why Dependency Grammar?

I will briefly list a few reasons for why the study of Dependency Grammar is interesting from a linguistic point of view and important for natural language syntax and semantics.

Despite certain formal similarities between Dependency Grammar and Phrase Structure Grammar, they represent very different ways of doing syntax from a linguistic point of view. DG analysis takes the relationship between words to be central while this is a derived concern in PSG which takes constituency or groupings of words to be of primary importance.

There are a number of advantages of adopting the dependency approach to natural language syntax. From a computational linguistic perspective, it provides more efficient parsing algorithms and allows for easier manual annotation of corpora (which increases the number of large corpora for such analysis). It is also more equipped to deal with free word order languages such as Russian and Czech which in some cases can not be dealt with at all by constituency approaches.

From a more theoretical perspective, it does not assume a universal syntactic level capable of explaining all languages (UG in generative linguistics) but does produce a uniform method of analysis applicable to many of the world's languages.

Initially, it was conceived of as a cognitive approach to grammar (by Tesniere, see above quote) which aimed to represent how the "mind perceives" various connections between words. Hudson (1993, 2007) argues for a similar psychological reality of dependency structures.

Another important aspect is "arguments that concern the advantages of a dependency-based structure for the interface with text (the linear rules) and for the interface with the meaning" (the syntax-semantics interface)" (Kahane, 2012: 258). These arguments are of direct importance here. It is often argued that dependency grammar allows for more transparent and apparent semantics. In fact, this claim will be investigated in depth in later sections and verified for certain dependency structures.

The lack of overt constituent structure, however, has cast doubt on DGs ability to generate a compositional semantics. As we saw in Part I and II, the principle of compositionality has many versions and controversies yet it is unified on the topic of constituency. Or at least the application of the principle has been almost exclusive conducted on the assumption of constituency. The uniqueness of this research is then to provide a compositional analysis of a grammar formalism in the absence of this assumption. This will be the task of the rest of this paper.

3.2 Compositionality applied

3.2.1 Dependency Algebra

The multi-sorted algebra or grammar for generating dependency structures is a 6-tuple given as:

Definition 14. Let the Dependency algebra be $DG = \langle E, W, C, Rol, F, R \rangle$ where E is the set of linguistic expressions generated by W the set of words or lexical entries (through R). C is the set of lexical categories (N,V, Adj etc). *Rol* designates the set of roles (subj, obj, vmod, nmod, arg1 etc). F is a function that maps elements of C to elements of W (i.e. projections which can also be seen as labels), $F : C \to W$. R is the set of operations that specify legal operations on the universe and generate the expressions in E from W, The next component which will be important for our formal specification of Dependency Grammar. It is a subclass of the Dependency algebra. This is the generating algebra which I will call dependency structures (DS). The finitely generating algebra is one which uses a subset of DG above to generate the entire carrier of the algebra.

Definition 15. Let $DS = \left\langle [DS^{+/-}], W, C, Rol, F, R \right\rangle$ be an algebra such that $DS^{+/-} \subseteq E$. Then $\left\langle [DS^{+/-}], W, C, Rol, F, R \right\rangle$ is the smallest algebra containing $DS^{+/-}$ and is called the algebra generated by $DS^{+/-}$. Furthermore, $\left\langle [DS^{+/-}], W, C, Rol, F, R \right\rangle = \left\langle E, W, C, Rol, F, R \right\rangle$ and thus $DS^{+/-}$ is the generating set for E. The DSs are identical to the dependency structures mentioned before and represented in Fig. 4.

The next thing to do is to define the signature or operations R of the grammar. The idea is that the rules should use all the elements of the sets of the universe and generate only legal structures in a dependency grammar. The general schema of the rules will take this form:

Claim 16. R is a indexed set of operations of this nature:
$R_{lex} = \left\langle \langle w_i, c, n, \langle w_j, .., w_n \rangle, c', \langle r_1, ..., r_n \rangle \rangle \rightarrow DS^{+/-} \right\rangle$ which takes $w_i \in W$ with label c in position n in a string to its governor w_j (the structure allows for multiple governors)[26] with label c' through a role r in a DS.

There are strictly speaking two types of operations which can be performed for each word, either the root operation or the dependent operation (given above).

Claim 17. The root operation is differently specified and states that a given w is a root word if it takes a set of dependents from a set of roles and categories,
$R_{root} = \left\langle \langle \langle w_i, .., w_n \rangle, \langle c_1, ..., c_n \rangle, \langle n_1, ..., n_n \rangle, \langle r_1, ..., r_n \rangle, w_i, \rangle \rightarrow DS^{+/-} \right\rangle$.

The operations are to be thought of as rule schemata which can represent a large number of specific rules. This algebra constitutes a fully abstract account of dependency grammar. Following Pagin and Westerstahl (2010), we do not apply our semantics directly to the expressions of the syntactic algebra but rather to their "derivation histories" or "analysis trees" represented in the term algebra (defined in section 2.4.1). Sentences can be identified with strings of words but the derivation histories are given by terms in the term algebra. We can derive a grammatical term algebra in which all the syntactic functions are defined for their arguments. This works as follows. Consider the following sentence.

(3.2.1) Some men cook.

Syntactically speaking, we get this sentence by applying the dependency operation in R of the syntax algebra to 'some', then to 'men' and then we apply the root operation to 'cook' such that $R_r(R_2(R_1(some))men)cook$. Now, there is a term t such that

$$t = R_r(R_2(R_1(some))men)cook$$

[26]There could be cases in which double dependence is warranted. Consider the cases in which the adjectives may depend on the noun and the verb, *John washed the clothes clean* where *clean* seems to depend on both *washed* and *clothes.* South African English allows for constructions such as *She watched the movie finished* in which *finished* has a similar double dependence.

R_n is a function that takes an element of E to other elements of E, namely its governors. For example, it takes *some* and yields as a value [[*some*]*men*], but $R_1(\textit{some}) \notin E$, only in $GT(E)$. Thus,

> Each term in GT_E corresponds to a unique string in E. Thus, there is a *string value function* V from GT_E to E. For a simple term like *most*, $V(\underline{\textit{most}})$=most, the corresponding expression (Pagin and Westerstahl, 2010: 5).

Technically, V is a homomorphism from the term algebra to the set of expressions. We let the grammatical term algebra $GT(E)$ be the domain of the meaning function. In other words, the meaning function maps only grammatical terms to their meanings. This procedure will be detailed in section 3.4.

Saturation constraints

The requirement of saturation on dependency structures allows us to specify the acceptable conditions of our grammar according to language specific constraints. In languages such as English the set of constraints will consist of word order constraints as well. Other so called free word order languages such as Russian and Czech will have relaxed or no word order constraints. The idea is that a given DS can only be saturated if it fulfills certain constraints.

The operations can be generalised for lexical categories through such constraints on the structures. Consider the case for nouns. DS^+_N requires that $R_{nn} = \langle\langle w_i, N, n, w_j, V, subj/obj\rangle \rightarrow DS^+\rangle$. This operation states that a noun has to depend on a verb in a given DS in order for that DS to be saturated. In this way, the constraint for adverbs is formally similar to that of nouns, $R_{adv} = \langle\langle w_i, Adv, n, w_j, V, vmod\rangle \rightarrow DS^+\rangle$. Rules for every lexical category can be specified in this way and thus we can generate an infinite set of dependency structures. It is important to note that unlike formalisms such as Categorial Grammar, rules for lexical categories cannot be derived compositionally from one another. The constraint on verbs can be given in terms of valency, i.e. $R_{Tv} = \langle Subj, N, Obj, N \rightarrow DS^+\rangle$.

There are also more formal constraints mentioned in terms of conditions on the dependency relation but these can be put in terms of constraints now.

Proposition 18. *(1) G is acyclic if* $\forall i, j \in V$*, if* $i \rightarrow j$ *then not* $j \rightarrow^* i$*, (2) G is single-headed if* $i \rightarrow j$ *then not* $k \rightarrow j$ *for any* $k \neq i$*. For saturation, connectedness is a good constraint,* $\forall i \exists j \in V(i \rightarrow j \vee j \rightarrow i)$*. Some grammars may even require projective structures (3) if* $i \rightarrow j$ *then* $i \rightarrow^* k$*, for any* k *such that* $i < k < j$ *or* $j < k < i$ *but as mentioned previously, we will not assume this here.*

We can now move on to a discussion of certain challenges which face the task of providing a straight Montagovian semantics for this type of syntactic algebra. After this, the notion of constraints resurfaces as we conceive of the semantic representation of dependency structures in terms of soluble constraint equations in a functional structure.

3.3 Challenges for a Type Semantics of Dependency Grammar

The first attempt at a compositional semantics for the dependency algebra should of course be a direct application of Montague grammar. The strategy would be to replace the usual categorial grammar in the syntax with a rule-to-rule translation from the dependency syntax. In this chapter, I argue that there are principled reasons preventing a Montague-style semantics for Dependency Grammar as it has been formulated here. I will focus on Extensional Montague Grammar (ignoring intensionsfor now) and show that Dependency Structures are unable to be represented in type theory without addressing these issues.

3.3.1 Extensional Montague Grammar

In traditional Montague Grammar, we start by generating the syntax by means of categorial grammar. The sentences of this categorial grammar are then compositionally translated into type theory. This is done by rule-to-rule translations from the rules of categorial grammar to the composition rules of the type semantics. There are three main stages in this process. I will present the process in turn through the use of a toy grammar which I will then use to show that there is no one-to-one mapping between the rules of this semantics and the rules of the dependency syntax described in the previous section. The details of the semantic analysis in this section is based on the proofs and definitions of GAMUT (1991). In the next section, I aim to resolve the issues brought out in this one.

Categories to Types

Categorial grammar consists of basic and derived categories. The basic categories are syntactic objects such as S for sentence, IV for intransitive verb, CN for common noun. From these categories through functional application, we obtain the derived categories α/β is a derived category given $\alpha, \beta \in CAT$. The convention in categorial grammar is to write in "result leftmost" notation "where the slash determines that the argument β is respectively to the right (/) or to the left (\) of the functor" (Steedman, 1998).

Thus, rules look like either (1) α/β, $\beta \Rightarrow \alpha$ or (2) β, $\alpha \setminus \beta \Rightarrow \alpha$. For instance, a term in categorial grammar could be derived as S/IV which takes an intransitive verb and generates a sentence or for a determiner the slightly more cumbersome $(S/IV)/CN$ which takes a noun and produces a term.

The first step in the compositional translation process involves defining a function that takes elements from the set of categories to elements in the set of types,

> $f : CAT \to T$ such that $f(S) = t$ and $f(CN) = \langle e, t\rangle$ and in general $f(\alpha/\beta) = \langle f(\beta), f(\alpha)\rangle$.

This function applied to the examples of categorial grammar above looks like $f(S/IV) = \langle f(IV), f(S)\rangle = \langle\langle e, t\rangle, t\rangle$ for general terms and for the determiners $f(S/IV)/CN = \langle f(CN), f(S/IV)\rangle = \langle\langle e, t\rangle, \langle\langle e, t\rangle, t\rangle\rangle$.

Translation of Lexicon

Lexical items in sentences are assigned their categorial representations and then translated into type theory expressions with lambdas. Back to terms and determiners, if we add transitive verbs we can compose entire sentences in Montague Grammar. I will provide a small lexicon with translations in this section.

Determiners: The lexicon is $LEX = \{every, a, some, the\}$, with the syntactic category $CAT = (S/IV)/CN$ (T stands for term defined above) and corresponding semantic type $T = \langle\langle e,t\rangle, \langle\langle e,t\rangle, t\rangle\rangle$. The translation of *every* is the expression $\lambda P\lambda Q\forall x(Px \to Qx)$ and the translation for a word such as *a(n)* is $\lambda P\lambda Q\exists x(Px \wedge Qx)$.

Nouns: $LEX = \{boy, girl, clown\}$, with the syntactic category CN and corresponding semantic type $\langle e,t\rangle$. Thus, the translation of common nouns is just the capital letters B, G, C.

Transitive_verbs: $LEX = \{likes, loves, gives\}$ with the syntactic category $IV/(S/IV)$ and corresponding semantic type $\langle\langle\langle e,t\rangle, t\rangle, \langle e,t\rangle\rangle$. The type theory expression for verbs is $\lambda X\lambda x X(\lambda y verbs(y)(x))$.

Of course, this is not an exhaustive list but it should serve us for the examples to come.

Rule-to-rule

For every rule of the syntax there is a corresponding composition rule for the semantics. Take the syntactic rule S3 which takes any CN α and yields a term every α. So we have the following semantic rule according to Montague Grammar:

T3: If $\alpha \in P_{CN}$ and $\alpha \mapsto \alpha'$, then $F_2(\alpha) \mapsto \lambda X\forall x(\alpha'(x) \to X(x))$

If the common noun is *boy*, then since *boy* $\mapsto$ *boy′* by the basic rule T1 and *every boy* is generated through the above rule which is translated into

every boy $\mapsto \lambda X\forall x(boy(x) \to X(x))$

The indefinite noun phrase is created through a similar rule to T3 (just the basic rule of translation of lexical items). Namely,

T3: If $\alpha \in P_{CN}$ and $\alpha \mapsto \alpha'$, then $F_4(\alpha) \mapsto \lambda X\exists x(\alpha'(x) \to X(x))$

Thus, resulting in the translation

a girl $\mapsto \lambda X\exists x(girl(x) \to X(x))$

Lastly, we have to derive a translation for the transitive construct *likes x.* The idea is that there is a rule S7 which takes a transitive verb and a term and creates an intransitive phrase which in turn can combine with another term to form a sentence. The rule looks like this

S7: If $\alpha \in P_{TV}$ and $\alpha \in P_T$, then $F_6(\alpha, \beta) \in P_{IV}$, and $F_6(\alpha, \beta) = \alpha\beta'$, where β' is the object from of β if β is a syntactic variable or else $\beta' = \beta$.

As before, there is a corresponding translation rule

T7: If $\alpha \in P_{TV}$ and $\alpha \in P_T$ and $\beta \mapsto \beta'$ and $\alpha \mapsto \alpha'$, then $F_6(\alpha, \beta) \mapsto \alpha'(\beta')$

This compositionally generates the translation of the English intransitive phrase *like a girl* to the semantic expression $\lambda X \lambda x X(\lambda y likes(y)(x))(\lambda X \exists y(girl(y) \wedge X(y))))$. After a series of beta-conversions, we arrive at $\lambda x(\exists y(girl(y) \wedge likes(y)(x)))$. This is now in the form of intransitive verbs in the syntax which can combine with a term to form a complete sentence through S2.

Every boy likes a girl $\mapsto \lambda X \forall x(boy(x) \rightarrow X(x))(\lambda x(\exists y(girl(y) \wedge likes(y)(x)))$ which is equivalent to $\forall x(boy(x) \rightarrow \exists y \forall z((girl(z) \wedge likes(z)(x)) \leftrightarrow y = z))$.

This constitutes a complete rule-to-rule translation of an expression in natural language to a semantic interpretation. The process is fully compositional.

In abstracto the homomorphism is established since for every syntactic rule

S_n α is a category (A/B), β is of category B $\Rightarrow F_k(\alpha, \beta)$ is of category A, where F_k is syntactic concatenation

There is a corresponding semantic rule

T_n α is a category (A/B), β is of category B, and $\alpha \mapsto \alpha'$ and $\beta \mapsto \beta' \Rightarrow F_k(\alpha, \beta) \mapsto F'_I[\alpha', \beta']$, where F'_I is functional application

3.3.2 Scope Ambiguity

It is often the case that many natural language sentences involving quantifiers and determiners have multiple readings. The type of sentence in the previous section is a famous example of such scope ambiguity. There are at least two possible readings of this sentence,

(3.3.1) $\forall x(Boy(x) \rightarrow \exists y(Girl(y) \wedge likes(y)(x)))$

(3.3.2) $\exists y(Girl(y) \wedge \forall x(Boy(x) \rightarrow likes(y)(x)))$

The first reading is the one which was compositionally translated in section 3.3.1. It states that every boy likes a possibly different girl. Example 3.3.2. is a reading which states that every boy or all the boys of a given domain like one and the same girl. The respective scopes of the universal and existential quantifiers mark the difference between these readings. Both are possible readings even though in many cases there are dominant readings such as 3.3.1. here. Still, our compositional semantics should be able to account for the different readings:

> The principle of compositionality requires that every (non-lexical) semantic ambiguity corresponds to a derivational ambiguity. Whenever a sentence has more than one meaning, there should be more than one way of constructing it (GAMUT, 1991).

In Montague Grammar, the alternative readings are generated by "rules of quantification" which is another method of sentence construction. The details are not important here except to say that the new syntactic rule is a schema for the generation of a number of rules. The idea is that we can create new sentences from the combination of a term and another sentence containing a syntactic variable if we substitute the term for a syntactic variable. The syntactic rule schema is given below:

> S8,n: If $\alpha \in P_T$ and $\phi \in P_S$, then $(\phi[he_n/\alpha])$ is of category S, where $\phi[he_n/\alpha]$ is the result of the following substitution in ϕ : (i) if α is not a syntactic variable he_k, then replace the first occurrence of he_n or him_n by α or an appropriate anaphoric pronoun and (ii) if $\alpha = he_k$ then replace every occurrence of he_n with he_k and him_n with him_k

The corresponding rule for the semantics is as follows:

> T8,n: If $\alpha \in P_T$ and $\phi \in P_S$ and $\phi \mapsto \phi'$ and $\alpha \mapsto \alpha'$, then $\phi[he_n/\alpha] \mapsto \alpha'(\lambda x_n \phi')$.

This indirect method of sentence construction allows us to create semantic expressions which belong to sets of properties referred to by the substituting term. The way this strategy resolves the scope ambiguity problem is by creating a lambda expression which corresponds to the property *every boy likes* him_4 in which (through T8) it is expressed that this property is in the set of properties specifying that exactly one girl has those properties.

There are other strategies for dealing with scope ambiguity in Montague Grammar, but the important thing is that they preserve the homomorphism between the syntactic rules and the semantic rules as the above method does.

3.3.3 Problems for Type-theoretic Dependency Semantics

Syntactic formation and Heads

The first problem for producing such type theoretic translations from dependency structures is that there is no analogous procedure to go from syntactic objects to semantic ones as in the previous sections. The rules of categorial grammar create constituent structures similar to the NP,VP,PP structures of Chomskian syntax. One major difference between this method of defining syntactic types and the dependency grammar method is that there is a patent lack of the notion of a head in the former but a central role of the head in the latter.

The operations in the dependency syntax take words as input and governors as output through the syntactic role of the heads. This blocks the type of semantics governed by functional application which is as flexible as syntactic concatenation. Since heads determine the specific syntactic combinatory possibilities, we cannot simply apply one category to the next via functional application in abstraction.

One fundamental difference is brought out in the way determiners are treated in the two theories. As we witnessed above, a common noun is the argument of a determiner phrase in both the syntax and semantics of categorial grammar. However, in dependency grammar, this situation is reversed and dependency formation rules will make the determiner subordinate to the noun because the noun is the head of the phrase and thus governs the determiner. In dependency grammar heads govern the composition process and there is no obvious way to incorporate this into the type semantics of Montague. Consider the transitive verb construction, the formation rule for a transitive verb is comparatively simple in Dependency Grammar, it takes two nouns through the subject and object roles respectively. The logical type is much more involved than this if we follow the above procedure and thus we loose the rule-to-rule translations.

The situation for modification is similarly problematic since adjectives and adverbs generally modify (depend on) their respective nouns and verbs while in type theory the order is reversed as the modifiers are the functors and the verbs and nouns the input. For instance, the functional type of an adjective CN/CN says nothing about the noun being the head of the combination and the adjective depending on the noun.

The result of the contrasting syntactic formation rules is that constituent structures are much more malleable to combination while dependency structures are more numerous. We can define new rules of syntactic combination at will in Categorial Grammar, this is not generally the case in Dependency Grammar. This point is related to the description of Dependency Grammar as a constituentless grammar formalism. As mentioned before, it is a core problem for compositional accounts of Dependency Grammar, since the lack of constituent structure obscures the part-whole relation required for compositionality. It is difficult to define the meaning of the whole in terms of its parts when the parts are numerous and not ordered by simple concatenation.

Scope Ambiguity again

Another serious problem is that maneuvers of the sort described in section 3.3.2. to account for scope ambiguity are not available to us. This is a significant charge against the compositionality of the system since the different readings are not spawned from different possible syntactic derivations. Let us consider the two readings of the quantified sentence from before,

(3.3.3) Every boy likes a girl.

(3.3.4) $\forall x(Boy(x) \to \exists y(Girl(y) \wedge likes(y)(x)))$

(3.3.5) $\exists y(Girl(y) \wedge \forall x(Boy(x) \to likes(y)(x)))$

According to the operations of the syntax, the first reading can be generated quite naturally in Dependency syntax. *Likes* is the root which takes two arguments in its saturation constraints, namely a subject and an object. In the DS, the role of *boy* is the subject and *girl* the object. More precisely,

Claim 19. $R_n : \langle\langle boy, CN, 2, like, verb, subj\rangle \to DS^+\rangle$
$R_{n'} : \langle\langle girl, CN, 5, like, verb, obj\rangle \to DS^+\rangle$

which together saturate the DS for the root verb *likes*. The issue is that there is no DS which naturally accommodates *girl* as the syntactic subject or *loving a girl* as the separate constituent and thus it is not possible to derive the second reading compositionally in this way. Montague's rules of quantification won't work in this case.

Non-projective structures

Non-projective dependency structures present another problem for type theoretic translations.

> Because of the flexibility of logical typing, there are various means of logical typing for many discontinuous constructions, such as topicalization, relativization, extraction, pied-piping, islands, etc. At the same time, till recently, there were no D-type [dependency type] calculi for discontinuous dependencies (Dikovsky).

One notable attempt at such a calculus is Categorial Dependency Grammar. This is an approach which aims at converting the head-driven nature of syntactic formation in DG into the canonical rules of CG. However, the approach itself is validation of the point that head-driven dependency analyses *simpliciter* are difficult to accommodate in type-theoretic semantics.

Moreover, there is no natural way to semantically represent the non-projective structures of pure DG. This forces the linguist to adopt a constituency/categorial reformulation of these structures in terms of discontinuous constituents and then use the tools which have been developed for semantic analysis the latter.

The problem with this methodology is that discontinuous constituency in phrase structure (or other constituency based grammars) is often cited as an analogous phenomenon to non-projective constructions in DG, but they are not always identical. Consider the common discontinuous construction type in English.

(3.3.6) John switched the light on.

(3.3.7) Mary broke the engagement off.

Neither of these constructions generate non-projective dependency analyses. Thus, the usual attempts at accommodating analogous structures (through movement or copy theory etc) in PSG will not translate directly into DG. And a semantic analysis based on such syntactic accommodation will be useless in the semantic analysis of non-projectivity. Therefore, conversion is not the answer. A semantics for DG needs to respect the idiosyncrasy of the formalism.

Subconclusion: Towards an alternative semantic characterisation

In this section, I have shown that there are principled reasons against the compositional treatment of dependency syntax in terms of type-theoretic semantics. Syntactic typing, insoluble scope ambiguity and non-projective dependency structures prevent the type of semantic translation schemata that Montague defined for categorial grammar. Perhaps at the base of these difficulties is the fact that the head-driven dependency relations which characterise the syntax is not easily representable in type theory, a formalism which better maps syntactic concatenation to functional application. Perhaps a succinct way of summing up this last chapter is by viewing these arguments as arguments which show that the principle of syntactic integrity is not respected in the standard analysis.

However, this does not mean that a standard analysis is impossible just that is requires some more ingenuity on the part of the semanticist. It is due to the above difficulties that I argue for a theory of meaning which can represent the semantic relations between syntactic objects in a more overt way, one which reflects the headedness of dependency analyses. Specifically, I look toward a method for representing the unique syntactic relations of dependency grammar in the semantics. The first step toward this goal is moving from Extensional to Intensional Type-theory. This and other matters are the topic of the next section.

3.4 A Resolved Montague Grammar for Dependency Structures

In this section, I adapt and extend the semantic treatment of F-structures in Lexical-Functional Grammar (LFG) due to Fenstad (1986) to DG as I have defined it here. Although, we analysis does depart quite radically from such treatments in general, it starts from a similar basis. However, before doing so, I will briefly outline the key elements of LFG which will be pertinent to my analysis. The main aim of this section is to provide a compositional semantics for dependency structures conceived as semantic structures themselves. This account respects the principles of syntactic and semantic economy as well as the principle of syntactic integrity (see section 2.4.6).

3.4.1 F-Structures in Lexical-Functional Grammar

Lexical-Functional Grammar (LFG) was developed to produce a model of natural language that achieved the twin goals of psychological reality and computational applicability. Asudeh and Toivonen (2009) describe the methodology harnessed to achieve these goals.

> A central idea of Lexical-Functional Grammar is that different kinds of linguistic information are modelled by distinct, simultaneously present grammatical modules, each having its own formal representation. The grammatical architecture of LFG thus postulates a number of simple data structures with mappings defining the relationships between structures.

There are two syntactic modules in LFG, C-structures and F-structures. C-structures are familiar from PSG and they generally contain only syntactic information such as constituency and subordination etc. F-structures, on the other hand, are meant to model grammatical functions. Their values are given by often complex feature structures which contain a considerable amount of information. Another way of putting this is that F-structures are sets of attribute-value pairs. A given f-structure is only well-formed if it adheres to consistency defined such that for every attribute in it in every f-structure, there is at most one value. In addition, the distinct modules of LFG take different elements as primitive. Consider the following examples.

(3.4.1) John resigned yesterday.

(3.4.2) Yesterday John resigned.

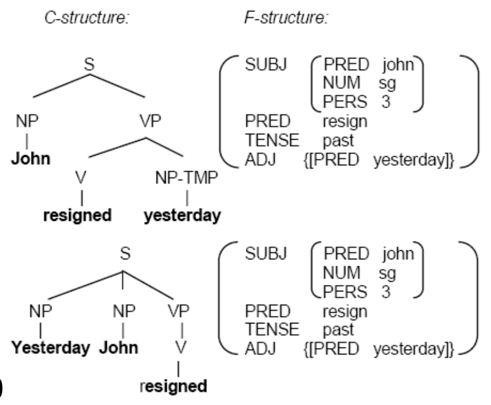

Fig. 10

The f-structures for both (3.4.1) and (3.4.2) are identical. For the most part, since C-structure captures the more superficial surface structural differences between languages while f-structures operate over "governable grammatical functions" which can be selected by predicates. This means that very different c-structures can possess the same f-structures and furthermore that any discoverable linguistic universals will be found in f-structures, if anywhere. The grammatical functions which are governable by selecting predicates are functions such as subj, obj, xcomp, comp etc. These are the primitive elements of the f-structures and there are other derived functions as well.

3.4.2 The Semantics

Situation Schemata and Functional DG

In this section, I provide a semantics for Dependency Grammar which is in part based a specific form of semantics designed for LFG type interpretations. In 'Situation Schemata and Systems of Logic related to Situation Semantics' (1986), J.E. Fenstad describes a method for generating a semantic interpretation from linguistic forms through insights from both LFG and situation semantics.

The methodology of this analysis is quite distinct from the usual compositional style semantics of Montague and others. Fenstad states it in this way:

> We would like to represent the constraints which are imposed on the interpretation of an utterance by its contextual and linguistic constituents through a cumulative system of constraint equations (1986: 93).

The idea is that we can systematically determine the meaning of utterances by solving these constraint equations consistently. He starts by defining a representation system called a situation schema. The intuition is that every expression is comprised of a *semantic predicate* which is related to a number of *actors* which play various *roles*. In addition, the predicate and its actors can be modified in myriad ways. The form of a situation schema for a simple sentence ϕ is:

Situation schema of ϕ : $\boxed{\begin{array}{l}\text{REL -}\\ \text{ARG 1 -}\\ \vdots \\ \text{ARG n -}\\ \text{LOC -}\end{array}}$ (Fenstad, 1986: 93).

Thus, the situation schema is a function with lists REL, ARG 1,...,ARG n, LOC and more. The value of the schema $SIT.\phi$ is given by the predicate and the roles of ϕ.

F-structure and Dependency Grammar have an important similarity. F-structures are designed to capture predicate-argument relations which are reflected in grammatical functions (situation schemata are supposed to be "refinements" of f-structures). Unlike context-free representations of natural language constructions which often obscure these relations, both f-structures and dependency graphs do not. In DG, this is due to the flat structure of the analysis, predicate-argument relations can be read off quite easily. In fact, this is one of the advantages of the formalism. Compare the ditransitive constructions in the sentences of dependency grammar in figure 11 versus the X-bar analysis of sentences in figure 12.

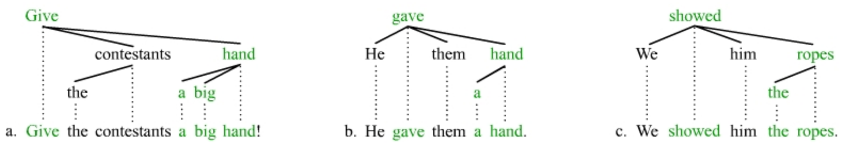

Fig. 11

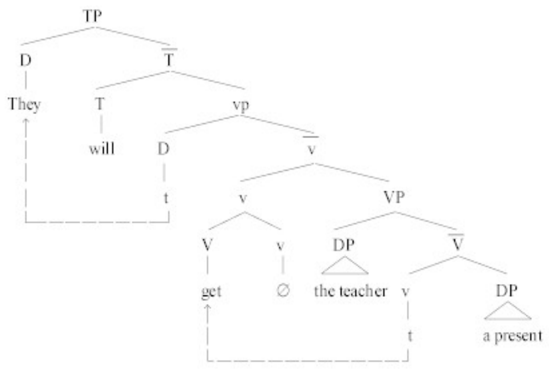

Fig. 12

In LFG and in the larger work by Fenstad, Halvorsen, Langholm and van Benthem (1985), there is considerable effort concerning the conversion of context-free grammar (c-structure) to situation schemata or f-structures.[27] My task is comparatively simple, since I argue that *DS*s need not be represented by situation schemata given the flatness of the formalism and its transparent predicate-argument structure. Furthermore, Dowty's principles and the principle of syntactic integrity require that we solve the issues mentioned in the previous section in a

[27]LFG converts linguistic forms into f-structures through a Correspondence or Parallel Projection Architecture.

way such that structure of the grammar is respected. The task after the conversion of the syntax to situation schemata is then dispensed with which allows dependency structures to be interpreted directly in standard semantics.

3.4.3 Montague Semantics for Dependency Grammar

The architecture of the entire system constitutes a Montague Grammar (with an added level of representation) which consists of the following components:

1. the syntactic algebra $DG = \left\langle E, W, C, Rol, F, R \right\rangle$ generated by $\left\langle [DS^{+/-}], W, C, Rol, F, R \right\rangle$
2. the type-logical algebra $TL = \langle (TL_t)_{t \in T}, (K_\delta)_{\delta \in \triangle} \rangle$
3. the semantic algebra $\mathcal{M} = \langle (M_t)_{t \in T}, (G_\delta)_{\delta \in \Delta} \rangle$ which is similar to TL
4. the interpretation homomorphism τ from TL to $\mathcal{M}$
5. the translation homomorphism *tr* from $T_{E,DS} = \left\langle (T_{E,DS,s})_{s \in S}, (R^T_\gamma)_{\gamma \in \Gamma} \right\rangle$ the term algebra of DG with respect to DS, to TL.

The picture above is parsimonious since it involves fewer steps from syntax algebra to semantics than analyses that make use of separate intervening semantic structures. Although it does share this methodology since there are semantic structures which are converted to the type-logical language, the main difference is that these structures are generated by the syntax algebra and not distinct levels of representation.

The chosen method is not uncommon in semantics, i.e. semantic interpretation through associated functional structures. The syntax algebra or rather dependency structures are depicted as functional structures that are then interpreted through an intermediate language. Much like the situation schemata of Fenstad or the semantic structures of Halvorsen (1983), DSs are acyclic graphs and this allows for the syntactic structures themselves to be fitting candidates for semantic analysis.

The steps involved in establishing the Montagovian type of syntax-semantic homomorphism at which we have been aiming are presented below:

Fig. 13

$$\begin{tikzcd}
\boxed{\text{Sentence of NL}} \arrow[d, Rightarrow] \\
\boxed{DS^+} \arrow[d, Rightarrow] \\
\boxed{\text{Formula of Type-Theory}}
\end{tikzcd}$$

This semantic analysis is segmented and the translation from a functional DS to a formula of intensional type-theory is often interceded by means of the specific semantic underspecification, e.g. the scope of quantifiers are represented as holes in a given DSs to be plugged in the type theoretic translation (details below). The actual intermediate language used to interpret these structures is not important, Fenstad used Situation Semantics and in principle this language can be dispensed with and a direct interpretation can be provided (however, certain issues mentioned in section 3.3.3 block this option here). For an example of the complete procedure, let the sentences ψ be

(3.4.3) John liked the car.

The syntax of this sentence can be represented as a *DS*: $[[john]like[[the]car]]$. My contention is that this same *DS* can also be represented as a functional structure, where the root word is the value of ROOT, the arguments of the verb the values of the argument lists with slots for specifiers and modification. Tense and aspect can be incorporated by use of the tense logical operators.

$$DS(\psi) = \boxed{\begin{array}{ll} \text{ROOT - } like' & \\ \text{ARG 1 - } & \boxed{\begin{array}{l}\text{IND - } \lambda PP(j)\\ \text{ROL - SUBJ}\\ \text{GOV - } like'\\ \text{CAT - CN}\end{array}} \\ \text{ARG 2 - } & \boxed{\begin{array}{ll}\text{IND - } car'(y) & \\ \text{DEP - } & \boxed{\begin{array}{l}\lambda P\lambda Q\exists x(\forall y(Py \leftrightarrow x=y)\wedge Qx)\\ \\ \text{ROL - Spec}\\ \text{CAT - Det}\\ \text{GOV - } car'\end{array}} \\ \text{ROLE - OBJ} & \\ \text{GOV - } like' & \\ \text{CAT - CN} & \end{array}} \\ \text{TENSE - } H & \end{array}}$$

In the semantic analysis that follows, I take *DS*s to formulas of intensional type-theory while respecting the structure of the syntactic operations of dependency grammar. In other words, we detail the procedure to get to the functional *DS* above and from there to the formula $like'(\hat{}car)(\hat{}j)$.

3.4.4 *DS*s as Semantic Structures

In this section, I will describe the architecture of the semantic *DS* or a dependency structure serving this dual purpose, I will also make comparisons between *DS*s as semantic structures and situation schemata as they are used to describe sentences of a grammar formalism. I hold that dependency structures themselves can be used to specify semantic information without the need for additional levels of representation. In formal terms, the functional *DS*s are restricted acyclic graphs. The attributes are selected from the set ROOT (matrix predicate), ARGi, ROL, GOV, CAT, DEP MODIFIER, TENSE (and other semantically relevant features such as ADJUNCT, NEG, etc). The values are either HOLE, LABEL, an expression of Intensional logic, a $sub - DS$ or a set of indices.

Consider the following ditransitive declarative sentence in English.

(3.4.4) b. The man will give the boy a gift
DS: $[[[the]man]will[give[[the]boy][a]gift]]$

Let ϕ be the English sentence in (3.4.4). The situation schemata are generated directly from the f-structures (which I omit here) and their grammatical functions.

$$SIT.\phi. = \boxed{\begin{array}{ll} \text{REL - give} & \\ \text{ARG 1 -} & \boxed{\begin{array}{ll} \text{IND - IND 1} & \\ \text{SPEC - the} & \\ \text{COND} & \boxed{\begin{array}{l} \text{REL - man} \\ \text{ARG1 - IND1} \\ \text{POL - 1} \end{array}} \end{array}} \\ \text{ARG 2 -} & \boxed{\begin{array}{ll} \text{IND - IND 2} & \\ \text{SPEC - the} & \\ \text{COND} & \boxed{\begin{array}{l} \text{REL - boy} \\ \text{ARG1 - IND2} \\ \text{POL - 1} \end{array}} \end{array}} \\ \text{ARG 3 -} & \boxed{\begin{array}{ll} \text{IND - IND 3} & \\ \text{SPEC - a} & \\ \text{COND} & \boxed{\begin{array}{l} \text{REL - gift} \\ \text{ARG1 - IND3} \\ \text{POL - 1} \end{array}} \end{array}} \\ \text{LOC -} & \boxed{\begin{array}{ll} \text{IND - IND 4} & \\ \text{COND}_{LOC} & \boxed{\begin{array}{l} \text{REL}_{LOC} \text{ - } > \\ \text{ARG1 - IND4} \\ \text{ARG 2 - 1} \end{array}} \end{array}} \end{array}}$$

In this situation schema, the value of REL is a ternary relation constant and the values of the other arguments are functional structures. Fenstad specifies LOC as having two structural relations, (1) $<$ which is temporal precedence and (2) 0 temporal overlap. This is meant to account for simple tenses.

He chooses to interpret ϕ through $SIT.\phi$ in accordance with situation semantics. The meaning of ϕ is $d, c||\phi||e$ where d is the utterance situation, the speaker's connection function is c and the described situation is e. This definition includes the so called relational theory of meaning.

The functions which correspond to the arguments are interpreted as generalized quantifiers or singular NPs. The LOC has a function as a value which anchors the described situation to a discourse location as given by the utterance situation d. Tense operates similarly by anchoring the temporal location of the situation schema. In Fenstad's terms (1986) there is a partial function g that does this anchoring in d if there is a $g(\ell) = \ell_d$ such that the infon $\ll <, g(IND.4), \ell_d; 1 \gg$ holds.

This is just a taste of the constraint-based semantic interpretation of a given situation schema. From the f-structures which often include complex feature structures and other syntactic information it may perhaps be necessary to introduce this additional layer of analysis. For our purposes, however, this complexity is not required. In addition, in section 3.4.5, I opt for an alternative interpretation to that offered by Fenstad in that I do not incorporate situation semantics in my analysis. I offer a more standard account in terms of intensional logic and a standard model theoretic interpretation of the type-logical formulas generated by the analysis.

Multiple dependency analyses (graphs) can map onto the same functional DS. In practice, this means that each node in a dependency graph is mapped onto a $sub - DS$. If the given DS is saturated, then all the dependents and roles have values. If not, we do not have a well-formed functional DS. The mapping is quite simple, the root of the DS is the semantic predicate and its dependents are the arguments. These arguments can be complex and contain their own dependents. Since there are no multiple levels in the dependency graph there is no need for a complex correspondence function as in LFG which specifies the f-structures for current nodes and their mothers in the c-structures. The resulting semantic structure is also an acyclic graph as is the case with dependency graphs.

Another important aspect of this semantic analysis is that it is order-free. As with Halvorsen's system (1983),

> The construction of semantic structures as defined here proceeds by successive approximation from any point in the sentence by way of symmetrical constraining equations (Kaplan and Bresnan (1982)). The algorithmic specification of the interpretation does not, therefore, impose any arbitrary constraints on the order in which the semantic composition should proceed.

This system allows us the freedom necessary to deal with scope ambiguity and headedness issues brought out in the previous section since we are free to compose the semantic relations as they fit into the functional structures in any order. This feature of the analysis is especially appropriate given the nature of statistical parsing algorithms in contemporary computational linguistics. Syntactic analyses are generated for sentences in terms of myriad dependency structures. Given the input of a sentence of natural language, multiple dependency graphs are generated with no specific order of combination. Thus, since there is no apparent "correct" way or order of structuring the dependency structures in the syntax, the property of order-freeness in the semantics is not an *ad hoc* convenience but another move toward respecting the syntax.

The semantic DS for ϕ above is given as follows:

$$DS\phi : \boxed{\begin{array}{ll} \text{ROOT - } give' & \\ \text{ARG 1 - } & \boxed{\begin{array}{ll} \text{IND - } man' & \\ \text{ROL - Subj} & \\ \text{GOV - } give' & \\ \text{DEP - } & \boxed{\begin{array}{l} \lambda P \lambda Q \exists x (\forall y (^{\vee} Py \leftrightarrow x = y) \wedge^{\vee} Qx) \\ \\ \text{ROL - Spec} \\ \text{CAT - Det} \\ \text{GOV - } man' \end{array}} \end{array}} \\ \text{ARG 2 - } & \boxed{\begin{array}{ll} \text{IND - } boy' & \\ \text{ROL - Ind.Obj} & \\ \text{GOV - } give' & \\ \text{DEP - } & \boxed{\begin{array}{l} \lambda P \lambda Q \exists x (\forall y (^{\vee} Py \leftrightarrow x = y) \wedge^{\vee} Qx) \\ \\ \text{ROL - Spec} \\ \text{CAT - Det} \\ \text{GOV - } boy' \end{array}} \end{array}} \\ \text{ARG 3 - } & \boxed{\begin{array}{ll} \text{IND - } gift' & \\ \text{ROL - OBJ} & \\ \text{GOV - } give' & \\ \text{DEP - } & \boxed{\begin{array}{l} \lambda Q \lambda P \exists x (^{\vee} Q(x) \wedge^{\vee} Px) \\ \\ \text{ROL - Spec} \\ \text{CAT - Det} \\ \text{GOV - } gift' \end{array}} \end{array}} \\ \text{TENSE - } G & \end{array}}$$

The procedure for compositionally translating such structures to formulas of intensional type-theory such as $give'(man', \hat{}\lambda X \exists x (gift'(x) \wedge^{\vee} X(x), \hat{}\lambda Y \exists y (boy'(y) \wedge^{\vee} Y(y))$ is given below.

The picture for getting from semantic DSs to intensional logic is roughly as follows $(a) \Longrightarrow (b)$:

(a) $\boxed{\begin{array}{ll} \text{Root } \alpha & \\ \text{Arg1 } & \boxed{\begin{array}{l} \zeta \\ \text{Dep -} \\ \kappa \end{array}} \\ \text{Arg2 } \xi & \end{array}}$ (b) $\alpha(\hat{}\xi)((\kappa)\hat{}\zeta)$.

3.4.5 Translation Procedure

There are certain guiding principles for the translation from the semantic DS to a formula in intensional logic. For instance, not all syntactic marking makes it up to the level of semantic translation or intensional logic. Only semantic forms are assigned semantic DSs and $sub - DS$s in the lexicon (refer to table 1 for this information). These structures or $sub - DS$s are given by constants of intensional logic or one of the other attributes mentioned in section 3.4.4

Table 1.

Elements	Type	Semantic Structure
do, there, it (non-anaphoric)	syn feature	$\emptyset$
Tense	Sem - no list	G, H, F, P
'man', 'woman', 'unicorn'	Sem - no list	$man', woman', unicorn'$
'a'	Sem - no list	$\lambda Q \lambda P \exists x (Q(x) \wedge Px))$
love, kick, run	Sem - list	$love', kick', run'$

Dependency graphs have a number of semantically irrelevant features, morphological features and syntactic features such as particles and dummy verbs etc. These elements are not translated into intensional logic. They serve no semantic purpose. Technically, they are still present in the functional structure but can be treated like "don't-care" terms of digital logic which have no effect on the output even if they are included in the input of the function. The semantic forms which do not have lists as values are given simple $sub - DS$s or formulas of intensional logic directly as semantic structures. The ROOT semantic form is a form with an argument list and is treated somewhat differently. Namely, the lexical forms which possess different argument-structures have the same semantic structure or DS. Consider the two sentences below.

(3.4.5) John kissed Mary.

(3.4.6) Mary was kissed by John.

The semantic structure of the root is $kiss'$ in both cases since $kiss \langle John, Mary \rangle$ is the predicate-argument structure of both sentences. $Mary$ may be the grammatical subject of (3.4.6) but it is not the semantic subject. "The designators that occur in the various argument positions simply indicate how the semantic structures of the grammatical argument of the PRED are the related to the semantic structure of the [ROOT]" (Halvorsen, 1983: 581). This is a nice feature of the system as explains the synonymy of active-passive pairs of sentences. In formal terms, our compositional semantics has the property of non-hyper-distinctness.

Technically, the application of the translation rules (below) introduce constraint equations. The resolution of the constraint equations (the resolution of such equations is presented in Kaplan and Bresnan 1982) allow the functional DS to be composed and the intensional logic formula to be derived. The constraint equations are guided by the translation rules which in turn are informed by the operations of the syntactic algebra.

Below are more specific rules for translating dependency syntactic structures and rules into semantic ones.

Basic Categories

In this section, we generate the semantic rules for each syntactic rule of DG. This process wil resemble the process for categorial grammar closely. We will start with the basic syntactic categories and their semantic type-translations as Montague did. In order to bring out the parallels in the formalism, we will represent these rules by abstracting from the roles and other factors. This reduces the rules to function-argument structure which was one of Frege's main contributions to compositional semantics (Partee, 2011: 16). So, there is a function f that takes syntactic categories to semantic types as with categorial grammar in section 3.1.1, $f(S) = t$, this time the mapping is from dependency categories to semantic types, $f : Cat \to Types$. The mapping is surjective since two syntactic categories can be matched with the same semantic type. The reverse is not always true. Consider the type for common nouns and intransitive verbs. The syntactic rules are as follows:

1. $CN = \langle NP \to DS^+\rangle = \langle e \to t\rangle$
2. $IV = \langle CN \to DS^+\rangle = \langle e \to t\rangle$

In both cases, a noun or object is required to saturate the DS which can be semantically interpreted as a function from entities to truth values. Thus, DS itself can be given a semantic value of t. Consider the rule for transitive verbs which takes terms and generates more complex syntactic objects from them.

1. $CN = \langle NPNP \to DS^+\rangle = \langle\langle\langle e \to t\rangle \to t\rangle, \langle e \to t\rangle\rangle$

In essence these rules take entities as input (here we understand verbs as events in a Davidsonian event semantics which are conceived to be unique individuals with space and time locations) and truth values as output. This analysis dovetails with Montague's proposal for construing transitive verbs as terms which apply to intransitive verbs. In other words, we have a function from sets of entities to sets of ordered pairs of sets.

The semantic type of completed dependency structures or sentences, i.e. ones in which all of the argument lists have values, is a truth value, $f(DS^+) = t$. In general the schema for the translation function is

> for a standard syntactic rule $f(\langle A \to DS^+\rangle) = \langle f(A) \to f(DS^+)\rangle$ where A is the dependent. And generally, $f(\langle A, B \to DS^+\rangle) = \langle f(A), f(B) \to f(DS^+)\rangle$.

Our order-free semantic composition allows for some freedoms in the ordering of arguments.

(3.4.7) Consider the sentence *Men love women.* Normally, the semantics of LOVE is a property of properties of sets of entities, i.e. a function from properties of sets of entities (with type $\langle\langle e \to t\rangle \to t\rangle$) to sets of entities (of type $\langle e \to t\rangle$). We will favour a semantic analysis of transitive verbs that better corresponds to dependency syntax. In terms of the syntax, LOVE takes two dependent NP's, $(NP, NP \to DS^+)$. This is equivalent to $(NP \to NP, DS^+)$ (given the free-ordering of arguments). Since NP has the type $\langle\langle e \to t\rangle \to t\rangle$, the complex syntactic object has the semantic form

$\langle\langle\langle e \to t\rangle \to t\rangle \to \langle\langle\langle e \to t\rangle \to t\rangle \to t\rangle\rangle$ through the function specified above. The corresponding formula is $\lambda X \lambda x X(\lambda y love'(y)(x))$.
MEN and WOMEN have the syntactic category CN and the semantic type $\langle e \to t\rangle$ which is equivalent to the formula $\lambda x(MEN'x)/\lambda y(WOMEN'y)$.
The last part involves applying the dependent semantic objects to their governor, i.e. $[(men)(women)Love]$.
Therefore,

$$\lambda x^{\langle e\to t\rangle}(MEN'x), \lambda y^{\langle e\to t\rangle}(WOMEN'y) \to \lambda Z^{\langle\langle e\to t\rangle\to t\rangle}\lambda x^{\langle e\to t\rangle}Z(\lambda y^{\langle e\to t\rangle}LOVE(y)(x)) = \lambda x^{\langle e\to t\rangle}(MEN'x) \to \lambda y^{\langle e\to t\rangle}(WOMEN'y), \lambda Z^{\langle\langle e\to t\rangle\to t\rangle}\lambda x^{\langle e\to t\rangle}Z(\lambda y^{\langle e\to t\rangle}LOVE(y)(x))$$

Determiners and modifiers of various sorts will be dealt with differently in accordance with the composition rules of the dependency syntax and since this process has been determined to be head-driven in the syntax, our semantics needs to incorporate a notion of headedness as well.

Head Semantics

The difference between this way of representing the syntax-semantics interface and the previous categorial grammar based method is that we have to incorporate a notion of headedness into the semantics. Instead of the constituents of the categorial grammar, we posit that in every sentence there are syntactic heads which become semantic heads in the semantic structure or semantic DS such that modification is applied to the heads where and when appropriate. This process takes the form of a translation rule.

Head_Rule: If $\zeta \in DS^{+}$and $\xi \in DS^{+}$ such that $\zeta \sqsubseteq \xi$[28], then $f(\zeta) \sqsubseteq f(\hat{}\xi)$.

Headedness is not the most transparent concept in theoretical syntax and it is often gestured at in vague terms. However, dependency grammar offers a intuitive way of defining heads directly in terms of governance. The above rule introduces a notion of semantic governance or at least a functional approximation of the governance relation present in the syntax. The semantic combination operates according to whether or not a given word or phrase is governed by another. As we have seen, nouns are applied to the verbs which govern them (and similarly in the root rule). Since the 70's and the work of Robinson 1970 the equivalence between constituency-based grammars and dependency ones has been well-known.

> A selection of one immediate head per constituent induces a unique DT [dependency tree] by the following induction: $C \subsetneq C' \Rightarrow root(ImmHead(C')) \to^{*} root(ImmHead(C))$ (Dikovsky, 2012).

Thus, this equivalence shows that dependency structures are produced when heads are selected in constituency structures. From here we can also provide an alternative treatment of determiner and modification composition, one which is more compatible with dependency syntax.

[28]where "$\sqsubseteq$" means ζ is governed by ξ

Let us start with determiner phrases. In all cases, determiners are governed by the nouns which they specify. However, in categorial syntax the noun is an argument of the determiner phrase. We do not necessarily require that the latter be reversed. Thus, determiners should always be derived after the $sub - DS$ containing the dependent clause to the root. This can be viewed in a situation in which a $sub - DS$ such as the one below is present:

$$\boxed{\begin{array}{ll} \text{IND - } man'{\downarrow_i} & \\ \text{ROL - Subj} & \\ \text{GOV - x} & \\ \text{DEP - } & \boxed{\begin{array}{l} \uparrow_i \lambda P \lambda Q \exists x(\forall y(Py \leftrightarrow x = y) \wedge^{\vee} Qx) \\ \\ \text{ROL - Spec} \\ \text{CAT - Det} \\ \text{GOV - } {}^{\wedge}man' \end{array}} \end{array}}$$

This means that, in general, dependents are added to the heads and determiners are added last in this process (indicated here by indexed up and down arrows). In constituency based grammars, heads are defined according to phrases. For instance, a VP may be the head of a given S, while a V may be the head of the VP etc. In dependency based grammars, the governors can fulfill this role. This is a “bottom up” approach to semantic composition and it is not unprecedented.

> Every phrase has a unique category, called its head, with the property that the functional features of each phrase are identified with those of its head. The head category of a phrase is characterised by the assignment of the trivial function-equation and by the property of being a major category. The output of each procedure is constructed by the subprocedure corresponding to the head (Frey and Reyle, 1983).

In this work, the authors provide an implementation for LFG in Prolog where the semantic representation is done through DRT. The process here is analogous. Dependents are applied in order to the head or in this case governors of the DS. In the above case, we have the determiner 'the' applied to 'man' such that $the'(\lambda x {}^{\wedge} man')$. Since we were dealing with intensional logic, we can state this as modification (and dependence in general) is applied to the intension of man' or the governor more generally. Similarly for other dependents of this sort. Therefore we can provide three semantic rules corresponding to the syntactic rules which apply to determiners *every*, *the*, *a*:

T2: If $\zeta \in DS_{CN}$ and $\zeta \mapsto \zeta'$, then $f_2(\zeta) \in DS_T$ and $f_2(\zeta) \mapsto every'\lambda x{}^{\wedge}\zeta'$

T3: If $\zeta \in DS_{CN}$ and $\zeta \mapsto \zeta'$, then $f_3(\zeta)$ and $f_3(\zeta) \mapsto the'\lambda x{}^{\wedge}\zeta'$

T4: If $\zeta \in DS_{CN}$ and $\zeta \mapsto \zeta'$, then $f_4(\zeta)$ and $f_4(\zeta) \mapsto a(n)'\lambda x{}^{\wedge}\zeta'$

These translation rules ensure that *every*, *the* and *a* are applied to the common noun in question and not the other way around such that, for instance, formulas like $every'(\lambda x{}^{\wedge}man')$

is the semantic counterpart of the universally quantified sentence. The same procedure applies to adjectives and adverbs which are applied to their respective nouns and verbs. The underlying combination rule is

HR': If $\zeta \in DS_{CN/IV}$ and $\xi \in DS_{Det/Adj/Adv}$, then $f'(\zeta, \xi) \in DS^{+}$ and $f'(\zeta, \xi) = \xi(\hat{}\zeta)$.

The headedness in these structures can be viewed as an alternative to constituency in phrase structure (or c-structures of LFG). It represents a way of organising the semantic composition by determining which elements get applied to other elements.

(3.4.8) Let us consider a simple determiner phrase such as *a man*. The syntax of such phrases is derived by taking a determiner and a noun and generating an NP, i.e. $NP = D, CN \rightarrow DS^{+}$. The difference in the analysis lies in the place that determiner attribution takes in the semantic composition process, i.e. determiners are added last to their respective nouns. The NP containing a determiner and noun pair still retains its type as $\langle\langle e \rightarrow t\rangle \rightarrow t\rangle$ on this account. A of semantic type $\langle\langle e \rightarrow t\rangle \rightarrow t\rangle$ is added to *man* of type $\langle e \rightarrow t\rangle$ after *man* is added to the semantics of its governor/s. Therefore, $\lambda x^{\langle e\rightarrow t\rangle}(man'x), \exists x^{\langle\langle e\rightarrow t\rangle\rightarrow t\rangle}(a'(x)) \rightarrow \lambda Z^{\langle e\rightarrow t\rangle}\exists x^{\langle\langle e\rightarrow t\rangle\rightarrow t\rangle}(man'(x))$.

3.4.6 Resolving Scope Ambiguity

The next hurdle to overcome is the problem of scope ambiguity. There are a few options for dealing with this problem given the mixed methodology used in this analysis. Underspecification is one common technique for dealing with a wide range of ambiguities (both lexical and structural) in natural language semantics without necessarily altering anything at the syntactic level. It represents another in a series of departures from the Montagovian ideal of syntax-semantics interface taken in this research. The option of quantifying rules etc. are available here but there are reasons to avoid this approach. Representing the scope ambiguities through alternative syntactic derivations can led to an explosion of ambiguity and the need for innumerable alternative syntactic configurations. See Bunt and Muskens (1999) for a proof of this based on the ambiguity in an average Dutch sentence. Motivation for syntactic alteration is often *ad hoc* and not syntactically well-motivated. In the case of DG this is even more so since the same dependency graph can generate a number of semantic ambiguities.

Importantly, the underspecification approach does not automatically entail a departure from the principle of compositionality (for the debate see section 3.4.7). Semantic underspecification is basically an intentional omission of linguistic information from semantic description. The underlying idea is to postpone semantic analysis until it can be executed in such a way that various ambiguities can be resolved. In other words,

> The key idea of underspecification is to devise a formalism which allows to represent all logical readings of a sentence in a single compact structure. Such a formalism allows one to preserve compositionality without artfully casting pure semantic ambiguities into syntactic ones...(Lesmo and Robaldo, 2006: 550).

This process amounts to a type of storage of interpretations without checking for consistency. At a later stage these interpretations are pulled out or extracted and interpreted in parallel. A given semantic representation can be underspecified in one of two ways.

> 1. atomic subexpressions (constants and variables) may be ambiguous, i.e. do not have a single value specified as their denotation, but a range of possible values;
> 2. the way in which subexpressions are combined by means of constructions may not be fully specified (Bunt, 2007: 60).

These paths specify constraints on representations of meaning and are often viewed as meta-representations which display all the representations that satisfy the set of constraints, i.e. all the possible readings of an expression. This fits well with DG since there is not always the possibility of syntactic recombination.

Thus, the functional *DS*s should not be thought of as representing single expressions but rather sets of sub-expressions which in turn represent the meanings of the different parts of the given sentence possibly containing items corresponding to (1) and (2) above. This marks another departure from Fenstad in the characterisation of situation schemata.[29] We will formally represent the UDS (underspecified dependency structure) for an utterance ϕ as a pair:

> $UDS.\phi = \langle E_U, C_U \rangle$ where E_U is a set of expressions and C_U is a set of constraints on the admissible ways of combining the subexpressions in E_U.

The way in which we will deal with scope ambiguity is to omit the scope relations in the functional *DS*s. We will follow Bos (1995) in introducing holes and labels into the representation. Labels $L_1, L_2, ...L_n$ are associated with each element of E_U. If a label L_1 consists of two subexpressions which are joined by a construction k, then $L_1 : k(L_2, h_1)$ where L_2 is the first of subexpressions and the second is an unknown entity called a hole "i.e. a variable that ranges over the labels of the subexpressions in E_U" (Bunt, 2007:64).

The HOLE variables are plugged by means of operations which replace the holes with subexpressions. The procedure is as follows. Labels are our constants and holes are the variables over these constants conceived as arguments over certain operators which possess scope. Constraints are defined to show us how we can reorganise the elements in such a way that they cover every possible reading of a given sentence. We now introduce our arsenal of labels and holes to the structure of the DS. ℓ_i represents a label with index i, h_i is the representation of a hole with index i and we write $\ell \leq h$ for ℓ is in the scope of h. Labelling is represented as $\ell_i : \phi$ for a formula ϕ.

Consider the DS for an example above with an additional argument, *John likes the car and not the van.* I have labelled the elements according to this key:

[29]For Fenstad situation schemata were *in situ* representations intended for use at some intermediate stage of semantic interpretation and resolved at a later stage. This approach also favoured by Alshawi and van Eijck's 'quasi-logical forms' (1987) and Schubert and Pelletier's 'conventional translations' (1982) is somewhat dated though.

Label	Element
$\ell_1 : L$	$like'$
$\ell_2 : J$	$\lambda PP(j)$
$\ell_3 : h_1$	$\lambda P \lambda Q \exists x(\forall y(Py \leftrightarrow x = y) \wedge Qx)$
$\ell_4 : C$	car'
$\ell_5 : V$	van'
$\ell_6 : h_2 \wedge h_3$	conjunction "and"
$\ell_7 : \neg h_4$	negation "not"

We then introduce h_0 as a variable for "narrow scope". Thus, we have our pluggings (which are mappings from holes to labels). "A plugging is a bijective assignment function, with the sets of holes as scope and the set of labels as range" (Bos, 1995). The constraints we have are that "car" and "van" should be in the scope of their respective determiners ($\ell_4 \leq h_1$, $\ell_5 \leq h_1$), while the latter should also be in the scope of the negation ($\ell_5 \leq h_4$) and finally that each conjunct should be in the scope of the appropriate side of the conjunction ($\ell_4 \leq h_2$, $\ell_5 \leq h_3$). There is only one possible plugging for the above sentence, as we would expect, since there are no ambiguities:

$$P_1 : \{h_0 = \ell_3, h_2 = \ell_4, h_3 = \ell_5, h_0 = \ell_7\}$$

The pluggings can be represented by an object language such as intensional logic as is the case here. This is the same strategy which we will employ to resolve scope ambiguities. Furthermore, we will incorporate labels and holes directly into our functional *DS*s. Consider the sentence below

(3.4.9) Every boy loves some girl.

We can represent this sentence in a *DS* with labels and holes in the place of certain forms. We start with the basic *DS* for the above sentence (excluding tense considerations for present purposes).

DS(3.4.7):

ROOT - $love'$
ARG 1 - IND - boy'
DEP - $\lambda P \lambda Q \forall x(Px \rightarrow Qx)$

ROL - Spec
CAT - Det
GOV - boy'
ROLE - SUBJ
GOV - $love'$
CAT - CN
ARG 2 - IND - $girl'$
DEP - $\lambda P \lambda Q \exists x(Px \wedge Qx)$

ROL - Spec
CAT - Det
GOV - $girl'$
ROLE - OBJ
GOV - $love'$
CAT - CN

The structure can be coded with the labels and holes for each of the lexical items and sentential operators, creating something of the sort below:

$DS'(3.4.7)$

ROOT - $\ell_1 : L$
ARG 1 - IND - $\ell_2 : B$
DEP - $\ell_3 : h_2$

ROL - Spec
CAT - Det
GOV - ℓ_2
ROLE - SUBJ
GOV - ℓ_1
CAT - CN
ARG 2 - IND - $\ell_4 : G$
DEP - $\ell_5 : h_3$

ROL - Spec
CAT - Det
GOV - ℓ_4
ROLE - OBJ
GOV - ℓ_1
CAT - CN

Once again, we introduce variables h_0 for "widest scope" and h_1 for "narrow scope". Firstly, in defining the constraints, we would want *boy′* to be under the scope 'every', i.e. $\ell_2 \leq h_2$, and *girl′* to be under the scope of 'some', i.e. $\ell_4 \leq h_3$. Then either 'every' itself is $\ell_3 \leq h_0$ or $\ell_4 \leq h_1$ and either 'some' is $\ell_5 \leq h_0$ or $\ell_5 \leq h_1$ for the respective scopes of the quantifiers. Hence, there are two pluggings for this DS:

$$P_1 : \{\ell_3 \leq h_0, \ell_2 \leq h_2, \ell_4 \leq h_3, \ell_5 \leq h_1\}$$
$$P_2 : \{\ell_5 \leq h_0, \ell_2 \leq h_2, \ell_4 \leq h_3, \ell_3 \leq h_1\}$$

The two pluggings correspond to the two possible readings for the quantified sentence. The translation into intensional logic needs to be conducted in accordance with the scope readings of the pluggings. Thus,

Claim 20. P_1 corresponds to $\forall x(Boy'(x) \rightarrow \exists y(Girl(y) \wedge love'(y)(x))$ where 'every' has wide scope and P_2 corresponds to $\exists z \forall y((Girl(y) \wedge \forall x(boy(x) \rightarrow love'(x)(y))) \leftrightarrow z = y)$ where 'some' or the existential quantifier has wide scope.

In this way, we have a solution to the problem of scope ambiguity. One which applies directly to our syntactico-semantic structures and falls in line with the order-free composition property of the structures themselves. This strategy is not only effective as a tool for resolving quantifier scope ambiguities but all kinds of natural language scope ambiguities including those involving disjunction and negation. In the next section, I extend this treatment to the often problematic case of "floating quantifiers".

Extension: Floating Quantifiers

There is a special case of expressions involving quantifiers which operates quite differently to the norm. These are the so-called floating quantifiers such as *all*, *both* and *each* which can fulfill various positions in sentences and completely optional in most cases. Consider the three sentences involving *all* below:

(3.4.10) We should all have been there for that event.

(3.4.11) We all should have been there for that event.

(3.4.12) We should have all been there for that event.

The quantifier *all* is omissible in every sentence without semantic effect. Hoeksema (1996) suggests that the reason for this property is that "the quantifier *all* is neither in an argument position, nor is it selected by any expression within the sentence". There are a number of semantic analyses of floating quantifiers most of which define them differently from normal quantifiers in order to capture (1) their semantically inert natures and (2) their unique flexibility of movement. These twin syntax-semantic goals make the phenomenon of floating quantifiers especially hard to explain. Sytacticians have gone from the traditional position of viewing these elements as floating off of certain NP-hosts, viewing them as inert and postulating movement of the hosts in the deep structure (Sportiche 1988) to treating them as predicate modifiers (Roberts 1987, Fukushima 1991, Van der Does 1992). On the semantics side, they have been treated as everything from operators (Van der Does 1992) to constituting their own linguistic categories (Hoeksema 1996). All of these theories have difficulties attached to them. I hold that the reason for these difficulties is brought about by attempting to treat these lexical items as syntactically or semantically different from regular quantifiers in some basic way either assigning them distinct syntactic categories or by type-shifting of some sort. I offer an analysis with does neither.

Floating quantifiers are exactly the same as regular quantifiers except that they have null-scope, i.e. they are dummy quantifiers. From a syntactic point of view we are largely in the clear since given the nature of the syntax of dependency grammar, we are not compelled to account for movement directly. First, we should treat floating quantifiers as normal quantifiers which are dependent on nominal (or pronominal) arguments of verbs. Movement considerations are prompted mostly by phrase structure, a formalism which requires one to account for elements which can occupy different positions in different phrases. No such requirement arises in dependency grammar, a word can be dependent on another no matter where it is positioned in the sentence. This situation is similar to the syntax of adverbial elements such as *often* which are also allowed somewhat free movement. In either case, dependency grammar does not need to offer an account of this movement since the dependency relation can range over various positions and need not individuate the constituents of phrase structure analyses.

Secondly, in terms of semantics, the quantifiers are the same type as regular quantifiers of their respective sorts. For instance so-called floating *all* is the same semantic type as the universal quantifier on my view, and *none of them* as the negation of the existential quantifier (in sentences such as *The children were none of them very impressed with the party*). The only difference is that when we define the pluggings for a given DS containing a floating quantifier, we assign it null-scope such that $\emptyset \leq h_{FQ}$ where the label for the floating quantifiers are $\ell_{FQ} : h_{FQ}$. Thus, there is no need for type-shifting or changing the lexical categories of these items, since they neither make it into the intensional logic (or whichever object language the DS is translated into) nor therefore the semantic interpretations of sentences. This explains there optional nature and their free movement in the most parsimonious way.

Negative Concord: A case for Afrikaans Negation

Negative concord languages (NC hereafter) are often cited as contravening the principle of compositionality of meaning as more than one negative element often does not change the negative meaning of a sentence. Here, I investigate one such language, a Dutch-descended language spoken in Southern Africa, Afrikaans. This phenomenon is a departure from what we would expect given a first order logical understanding of negation. From a compositional point of view, negation would correspond to sentential negation $\neg$ and thus more than one negation results in a positive sentence as *per* double negation. In fact, many languages do operate in this way, we call these Double Negation languages (DN hereafter), English, Dutch and German are such languages. Consider the following examples:

(3.4.13) I didn't not go to work today.

DN: I went to work today.

-

(3.4.14) Ik ben niet van plan om je niet vertellen

I am *not* of plan to you *not* to tell

DN: I am going to tell you.

-

(3.4.15) Niemand wird nicht zu dem Treffen heute Abend

Nobody will *not* to the meeting tonight

DN: Everyone is going to the meeting tonight.

Multiple negative elements produce a positive sentence as expected in these examples. In (3.4.13), the combination of an n-word *niemand* and the German negative marker *nicht* generates a positive sentence. It is important to note that it is not only negative markers that can carry a negative feature but also n-words or negative indefinites which correspond to $\neg\exists x$ such as *nobody*, *nowhere*, *nothing* etc. In NC languages this is quite different as two negative elements are sometimes required for one negative reading. Consider the Spanish and Russian examples below:

(3.4.16) María no puede encontrar a nadie

Maria *not* can find to *nobody*

NC: Maria can't find anyone.

-

(3.4.17) On nikogda ne znaet, kuda idti

He *never not* know, where to go

NC: He never knows where to go.

Similarly, Afrikaans has a double negative construction in which the negative marker *nie* is reiterated after the verb. Thus, it has two negative markers both with the form *nie* which roughly correspond to *not* in English. Both negative markers are required for the negation of most sentences in Afrikaans. In addition, Afrikaans has a number of n-words such as *geen* 'none, not any', *geeneen* 'not one', *geensins* 'by no means, not in any way', *nerens* 'nowhere', *niemand* 'no-one, nobody', *niks* 'nothing', *nooit* 'never'. The second *nie* is often considered a scope marker which marks the scope of the negation. Consider (3.4.16) and (3.4.17) below:

(3.4.18) Hy is nie 'n goeie mens nie.

He is *not* a good person *not*

He is not a good person.

-

(3.4.19) Sy het nie die kans gesien nie maar haar ouers het.

She did *not* the chance saw *not* but her parents did

She did not see the chance but her parents did.

In (3.4.16), the entire sentence is negated while in (3.4.17) only a constituent is negated and this is marked by the position of the second *nie.* However, there are instances in which only one *nie* is required. (3.4.18) is a general situation in which the sentence only consists of a subject and finite verb in the active voice and (3.4.19) is an interrogative construction. Neither construction allows for the scope marker or second *nie.* This also confirms the generalisation that "every negative sentence, regardless of whether it contains an n-word or a negative marker, ends with the (extra) negative marker nie" (Zeijlstra, 2009)

(3.4.20) Ek ken hom nie.

I know him *not*

I don't know him.

-

(3.4.21) Ken jy hom nie?

Know you him *not*

You don't know him?

I do not think that these cases refute the generalisation that Afrikaans negation requires two negative elements as they could be accounted for by the interference of other linguistic phenomena.[30]Importantly, constructions involving n-words also require the second instance of the negative marker *nie* and are even required in constructions similar to (3.4.17) and (3.4.18) above. However, interestingly two n-words generally produce a double negation reading in Afrikaans. Examples of all these constructions are given below:

(3.4.22) Riaan gaan nooit daarna toe nie.

Riaan goes *never* to there *not*

NC: Riaan never goes there.

-

(3.4.23) Die kinders is nerens nie.

The children is *nowhere not*

NC: The children are not anywhere.

[30]Cancellation of the second *nie* could be due to euphony problems such that the second *nie* is present just not realised in the surface structure. Two *nie*'s never directly follow one another in Afrikaans (unlike the possibility of two *not*'s in English litotes constructions).

-

(3.4.24) Niemand het niks gekoop nie

Nobody have *nothing* bought *not*

DN: Everybody bought something.

It seems that in such cases the second *nie* is not marking the scope of the negation since it is not strictly necessary for this purpose but is still required by the syntax. This is a peculiarity of Afrikaans, one which is in need of explanation. Furthermore, as (3.4.22) shows, double negation is possible in Afrikaans and it can be produced when two n-words are both present in a given sentence.

The way in which I propose to deal with this case of negation is inspired in part by Zeijlstra's (2007) treatment of the differing negation features of n-words and negative markers. It starts with the claim that "NC is analyzed as an instance of syntactic agreement between one or more negative elements that are formally, but not semantically negative and a single, potentially unrealised, semantically negative operator" (Zeijlstra, 2007: 1). In accordance with this notion and a distinction argued by Giannakidou (2000), Zeijlstra differentiates between two types of NC languages, Strict NC and Non-strict. The former states that every n-word requires a negative marker and the latter requires that only postverbal n-words co-occur with negative markers (preverbal n-words are banned from this). He offers Czech (and some other Slavic languages) as evidence of the existence of Strict NC while Romance languages such Italian are cases of Non-strict NC languages.[31]

The treatment of Afrikaans which I propose differs radically from Zeijlstra's in that he takes Afrikaans to be the missing link in a typology of natural language negation. We agree that NC agreement is between one formally and semantically negative operator and one or more non-semantically negative elements. Where we are in disagreement is the claim that since we have DN languages in which both negative markers and n-words carry [iNeg] features, Strict NC languages in which none do and Non-Strict NC languages in which only negative markers carry this feature, there is one more logical possibility and that is a group of languages in which the negative markers carry [uNeg] and are thus non-semantically negative while the n-words carry [iNeg]. Zeijlstra proposes that a certain variety of Afrikaans fills this gap. Indeed, this analysis does account for the examples above if we allow for the covert operator in (3.4.18) and (3.4.19). However, it not only requires the postulation of a panacea in terms of a covert negative operator (tied heavily to the Minimalist underpinnings) but

[31]The analysis of these languages is described in terms of the Agree operation and Feature checking within the Minimalist framework. However a detailed discussion of this goes well beyond the purview of the current work. In brief, Zeijlstra defines negative concord as a type of Agree relation between a formally semantically interpretable [iNeg] feature and at least one uninterpretable [uNeg] feature. Thus, NC languages can contain elements which only look negative but actually bear the [uNeg] feature. In other words, some negative elements on the surface can be semantically non-negative in this theory. In addition, this Agree relation is a Multiple Agree relation which means that multiple [uNeg] elements can be c-commanded by one element bearing [iNeg] in the feature checking. Finally, it is argued that in grammatically justified situations, a covert [iNeg] can be assumed to c-command any overt [uNeg] and "of course, language-specific properties determine whether this non-realisation possibility is actually employed" (Zeijlstra, 2007: 5).

it also fails to predict the correct reading of the following types of sentences (admitted by Zeijlstra in personal correspondence),

(3.4.25) Niemand het nie die werk voltooi nie

Nobody has *not* the work finish *not*

DN: Nobody didn't complete the work.

In addition, it neglects the fact that the *nie* seems to have a strong negating force in sentences such as (3.4.18) and (3.4.19) as well. To claim that the marker is semantically vacuous seems counterintuitive. Morever, covert semantic material is hard to argue for given the kind of syntax-semantics interface which I have been proposing here.

A sketch of the procedure which I propose is as follows: following the underspecification strategy of the previous section, we will differentiate between the two *nie*s of Afrikaans.[32] In terms of the DS containing these elements, we assign null scope to the second nie (or nie_2) and wide scope to both n-words and the negative marker nie_1 (or place the verb within the scope of the negation) which results in DN when placed in the same expression and accounts for all of the other data above when not. Both *nie*s are syntactically dependent on the verb but only one of them takes semantic scope over it. Therefore, the second negative marker is semantically vacuous and does not feature in the semantic analysis of the sentences of Afrikaans. This account deals with all of the cases and does not fall prey to the problems of Zeijlstra's analysis.

3.4.7 Compositionality and Underspecification

Although underspecification has become a ubiquitous technique is formal semantics, it is not settled whether or not the principle of compositionality is strictly adhered to by this technique. Given the analysis of dependency grammar above in terms of an underspecified Montague Grammar, a central question beckons, namely how compositional are underspecified structures (if at all)? Naturally, I will argue that functional dependency structures are indeed compositional, specifically in the weak direct sense of the term. Let us return to our definition of compositionality from the first part.

Principle: The meaning of a complex expression is a function of the meaning of its components and their method of combination.

The way in which we chose to interprete this principle was through the means of a homomorphism between the syntactic algebra and the semantic algebra. Specifically, for every rule of the dependency syntax there is a semantic rule providing the meaning of the syntactic expression produced by the rule. We encountered certain problems, however, when we attempted to account for quantifier scope. This led us to adopt an alternative strategy to the one employed by Montague, namely underspecifying the information presented in the

[32]This distinction is linguistically motivated. See Biberauer (2008a) for four differing characteristics of the two *nie*s.

dependency structures. There are two possible ways in which this technique could lead to a violation of the principle. (1) The meanings of complex terms are not determined by the meanings of their component parts or (2) the method of combination used does not maintain structural similarity between the algebras or more generally does not establish the appropriate relationship between the syntax and semantics.

It should be admitted that this alternative strategy does affect our original conception of compositionality. We are no longer assigning the meaning of components (constituents) in isolation and then using these assignments to generate the meaning of more complex expressions. Underspecification interferes with this process as the assignment of basic meanings is delayed (or stored) for later resolution. The important thing though is surely that the meanings of complex expressions are built up from the meanings of their components, a property which is still preserved during the resolution of underspecified structures. Therefore, the controversy comes down to whether the "method of combination" is the "intended method" for compositional analysis.

This objection stems from the fact that underspecification (through either enumerative or constraint-based approaches) involves taking functional structures as the input of semantic interpretation and not surface syntax. Indeed this is a worry since such a strategy could miss the point of a compositional syntax-semantics interface. The problem is that this is too general. If we do not consider the specific surface syntactic structures in our semantic analysis then it seems that we are not giving a semantics for that formalism specific. Consider, if both a phrase-structure tree and a dependency graph can be mapped onto the same functional structure. This structure is then interpreted in the semantics without any relation to the original syntactic configuration. Thus, we no longer have a direct homomorphism between the syntactic algebra and the semantic algebra, in fact the specific syntactic structure does not make much of a difference at all.

Moreover, the step from functional structures to semantic output is also problematic. As Fenstad claimed for his situation schemata, they can be interpreted through various semantic theories based on the proclivities of the semanticist. This means that neither the syntax nor the semantics is of particular importance for an account of the syntax-semantics interface given functionally underspecified structures, which is absurd. Using underspecified structures in this way is too general and does not provide us with an account of the syntax-semantics interface, which compositionality is meant to be. Therefore, it indeed does not provide the compositional analysis at which we are aiming.

The above is a serious concern for semantic underspecification techniques in general. However, this is not to say that these concerns cannot be mitigated by specific implementation which aims to avoid these problems. In this thesis, I have undertaken such a task.

The uniqueness of the current approach is that it renders the surface syntactic structures to functional structure conversion in such a way that the syntactic relations of dependency grammar are preserved in the functional structures.

In the present framework, it is not an arbitrary matter what the surface syntax is since it is from this syntax that the functional structures are generated. This process is a specific one, i.e. it would not be the same for a syntactic algebra based on phrase-structure or categorial grammar. The functional structures are composed of the root and dependency rules of the syntax algebra. The translation rules provide the guidelines for the semantic output of these structures specifically. But the translation rules too are married to the head semantics ordered by the governance relation of the syntax.

The point at which the underspecification enters into the analysis is no longer arbitrary and it aims to represent purely semantic ambiguities without resorting to altering the syntax. But at each step of both creating the underspecified structures and interpreting them, we work towards to goal of representing the surface syntactic formalism through a compositional semantics.

The analysis does not constitute a strongly direct compositional system since individual components (words) are not interpreted in isolation. However, as was noted in section 2.4.4, the requirement of direct compositionality is quite strong and not well justified in many cases. Our system does allow for synonyms [33] and thus is not a trivial account of the syntax-semantics interface and importantly it is an account of this interface in the intended sense. This latter claim is brought out by the fact that any alteration in the syntactic algebra will have a domino effect on the rest of the processes involved in interpretation, i.e. the nature of the functional DS and the eventual intensional logic translation.

3.4.8 Advantages of this Framework

In this part, I have detailed the process of taking terms from the dependency algebra and representing them in a dependency structure that includes both syntactic and semantic information, namely a functional DS. I have then offered a Montague Grammar which takes rules generated by the dependency grammar syntax and maps them to rules of the semantic algebra. The dual nature of the functional/semantic DSs allows for the process of semantic underspecification to solve many of the problems of the previous section without committing us to an additional layer of representation thereby preserving the rule-to-rule translations favoured by Montague.

I have shown that this system is powerful enough to account for various scope ambiguities as well as offer treatments of floating quantifiers and negative concord. In addition, it respects the principles of section 2.4.6 especially the principle of syntactic integrity. Semantic composition rules operate in connection with syntactic formation rules. This framework is both syntactically and semantically economical as well.

As for compositionality, we have a weak directly compositional system (as per Montague Grammar) with the property of non-hyper-distinctness (allows for synonyms) and the Husserl property (sameness of meaning implies sameness of semantic type). As we have seen, not every element of the syntax contributes to the semantic output. However, the rule-to-rule translations ensure that the homomorphism between the syntactic and semantic algebras remains in place.

There are two salient contributions to the literature on compositionality present in this research. The first is the introduction of the functional treatment of the generating algebra of

[33]The non-hyper distinctness property was evinced with the passive-active semantic equivalence above.

the syntax. This methodology takes the raw syntactic structures or dependency structures to be semantic structures used for interpretation as well. Through this device, various semantic phenomena can be explained and accounted for in a way that respects the architecture of the syntax. It also allows us to benefit from the powerful technique of underspecification in the semantics, a mechanism which is usually unavailable to a proponent of a strongly direct rule-to-rule semantics.

The second contribution is a fully compositional account of the semantics of a grammar formalism which lacks constituent structure. The process of semantic composition had to be altered and replaced with features that mirror the composition of syntactic objects in dependency grammar. This led to the development of a "Head Rule" and the incorporation of semantic heads which govern composition to a certain degree. Head semantics replaced constituency as the tool for semantic composition (although they need not be incompatible). In this way, I have also addressed the concerns for a compositional treatment of dependency grammar which I presented in the previous section. As such, I have extended the definition of compositionality to include languages which lack constituents, an extension which could enrich the debate on the nature of the principle itself.

The semantic analysis in this paper has taken the form of intensional logic, but this is a choice not a requirement. As we saw with Fenstad's situation schemata, our functional structures can be interpreted in terms of a variety of different semantic theories, including situation semantics. We are not married to a specific semantic theory or Montague Grammar for that matter. The idea of functional *DS*s is compatible with many and varying semantic analysis, which is an advantage of any system, provided that these analyses respect the syntactic structures of the interpreted formalism.

I will highlight some more advantages of this framework in my description of some of the problems facing its competitors.

3.4.9 Other Frameworks

I have already compared the analysis in this paper to pure Montague Grammar and to the pure underspecification semantics of Halvorsen and Fenstad. In this section, I will match the system described above to other theories which possess the common goal of providing a semantics for dependency grammar.

The Semantics of Word Grammar

Hudson (1984, 1991, 1998) offers an dependency based grammar formalism which takes the dependencies between pairs of words to be central. Word Grammar (WG) is considered to be a branch of cognitive linguistics. WG considers only words as syntactic units and dependencies between these words as constituting sentence structure. In general, word grammar is given a strong lexical semantics in terms of Frame Semantics (Fillmore 1982). This semantic formalism interprets individual words through a network of related words and concepts. Words relate to frames of knowledge related to concepts surrounding the use of that word.

However, the semantics of WG in terms of Frame Semantics does not pretend to be compositional. Whatever its advantages, they fall outside of the scope of this work.

Categorial Dependency Grammar

Categorial Dependency Grammar (Dekhtyar and Dikovsky 2008, Dikovsky 2004) does aim to provide a compositional account of dependency grammar syntax. Although it is similar in some ways to the analysis above, e.g. it includes underspecified semantic representation and a rule-to-rule translation from syntax to semantics, it fails to do so in a way that respects the nature of dependency syntax. By enforcing categorial rules on dependency grammar, we move toward a constituency based account of grammar. In these theories, dependencies are expressed in terms of classical categorial grammar with the addition of polarized valencies.

From this point, the semantics might become more standard, whether we use Descriptive Semantics as in Dikovsky (2012) or older formalisms to interpret the syntax since categorial grammar-like composition is assumed. Not only does this account fail to respect the integrity of the syntax but it also involves additional levels of semantic representation in terms of Descriptive Semantic Representations (DSRs) to which the account in this paper is not strictly committed.

Extensible Dependency Grammar

Debusmann and Kuhlmann (2008) propose a radical approach to the exploration of dependency grammars. XDG, as it is called, consists of "multigraphs" which are tuples of dependency graphs which share the same sets of nodes and a relational syntax-semantic interface. This is a multidimensional model-theoretic framework in which the XDG grammar narrows down the set of candidate grammars that eventual act as the models of the grammar itself. Dependency structures themselves are defined algebraically. The dependency algebra is generated through order-annotations which are interpreted as composition operations on the structures. As for the semantics, these multidimensional structures are interpreted through a relational interface which constrains the relation between the syntax and semantics through dominance constraints (which falls within the underspecification genre of semantic techniques), unlike accounts which derive the semantics from the syntax, as I have done.

The spirit of this account seems to be geared toward parsing efficiency and computational implementation. In so far as it achieves these goals, it also veers away from the principles discussed in this research. Both the syntactic and semantic structures are quite complex, involving multiple levels of analysis. Dependency multigraphs are so complex at times that they no longer possess the simple flat structure initially favoured by dependency syntacticians. Of course, this is not a knock down argument against this theory, it merely illustrates the divergent underlying motivations for this particular account of the syntax-semantics interface of dependency grammars.

3.5 Conclusion

This thesis has drawn together evidence and arguments from philosophy, linguistics and logic to lend insight into the topic of compositionality, its nature and implementation. The first part was devoted to describing the debate which has captured the contemporary field of natural language philosophy and linguistics. I discuss some of the foundations for the view that natural language is compositional and dismiss them in turn.

The next part opens up the compositionality debate in the arena of formal language theory. I argue that there are reasons based on complexity and the legitimacy of infinity considerations which make for a more compelling evidence in favour of compositional semantics for formal languages. I also define more precisely what the definition of compositionality entails for formal languages used to model natural language.

In the final section of the thesis, I offer a modified Montague Grammar for one such formal language, namely dependency grammar. This formalism is especially interesting since it lacks constituency, a property which forms part of most definitions of compositionality. The methodology I used is mixed. I utilise the flat structure of dependency syntactic analysis to represent the syntax-semantics interface in terms of functional dependency structures which are the generated by the syntactic algebra. I then interpret these structures through a series of rules, constraint equations and underspecification techniques into intensional logic. The result is a compositional system which can account for both the syntax and semantics of various natural language phenomena from scope ambiguity and floating quantifiers to negation and negative concord. This framework has many advantages over other accounts of the semantics of dependency grammar, which are detailed in section 3.4.7.

References:

Asudeh, A., and Toivonen, I. (2009). 'Lexical Functional Grammar'. *The Oxford Handbook of Linguistic Analysis.* Oxford University Press.
Barker, C., and Jacobson, P. (eds). (2007). *Direct compositionality.* Oxford: Oxford University Press.
Barwise, J., and Cooper, R. (1981). 'Generalized quantifiers and natural language.' *Linguistics and philosophy* 4.2: 159-219.
Biberauer, T. & S. Cyrino. (2009). 'Negative developments in Afrikaans and Brazilian Portuguese'. Paper presented at the 19th Colloquium on Generative Grammar (Vitoria-Gasteiz, Spain).
Biberauer, T. And Zeijlstra, H. (2009). 'Negative Concord in Afrikaans: filling the typological gap', lingbuzz/000830
Bos, J. (1996). *Predicate logic unplugged.* Univ. des Saarlandes.
Bresnan, J., et al. (1987) 'Cross-serial dependencies in Dutch.' *The formal complexity of natural language.* Springer Netherlands. 286-319.
Bunt, H. (2007). 'Semantic underspecification: Which technique for what purpose?.' *Computing Meaning. Springer Netherlands.* 55-85.
Bunt, H, and Muskens, R. (1999) 'Computational semantics.' *Computing Meaning.* Springer Netherlands. 1-32.
Chomsky, N. (1957). *Syntactic Structures.* The Hague.
Chomsky, N. (1995). *The minimalist program.* Cambridge, Mass.: MIT Press.
Chomsky, N. (2010). 'Some simple evo-devo theses: how true might they be for language?' In R.K. Larson, H. Yamakido, & V. Deprez (eds.), *Evolution of Human Language: Biolinguistic Perspectives.* Cambridge: Cambridge University Press.
Cooper,R. (1975). 'Montague's Semantic Theory and Transformational Syntax'. PhD Thesis, University of Massachusetts, Amherst.
Davidson, D. (1957). *Inquiries into Truth and Interpretation: Philosophical Essays.* Clarendon Press.
Debusmann, R. (2000). 'An introduction to dependency grammar'. *Hausarbeit fur das Hauptseminar Dependenzgrammatik* SoSe 99. Univeristat des Saarlandes.
Debusmann, R, et al. (2004) 'A relational syntax-semantics interface based on dependency grammar.' Proceedings of the 20th international conference on Computational Linguistics. Association for Computational Linguistics.
Debusmann, R., and Kuhlmann, M. (2010) 'Dependency grammar: Classification and exploration.' *Resource-Adaptive Cognitive Processes.* Springer Berlin Heidelberg. 365-388.
Dekhtyara, M., and Dikovsky, A. (2004). 'Categorial dependency grammars.' Nantes University.
Dikovsky, A. (2012). 'What Should be a Proper Semantics for Dependency Structures'. Nantes University.
Dowty, D. (2007) 'Compositionality as an empirical problem.' *Direct compositionality* 14: 23-101.
Duchier, D (2001) 'Lexicalized syntax and topology for non-projective dependency gram-

mar'. In: Joint Conference on Formal Grammars and Mathematics of Language FGMOL'01, Helsinki.
Gaifman, H (1965). 'Dependency systems and phrase-structure systems'. *Information and Control* 8. 304-337
Groenendijk, J., and Stokhof, M. (1991) 'Dynamic predicate logic.' *Linguistics and philosophy* 14.1: 39-100.
Groenendijk, J. and Stokhof, M. (2005). 'Why compositionality?' In Carlson and Pelletier (2005), 83–106, URL http://staff.science.uva.nl/˜stokhof/papers/wc.pdf.
Fenstad, J.E. (1986). 'Situation schemata and systems of logic related to situation semantics.' *Studies in Logic and the Foundations of Mathematics* 120: 91-104.
Fillmore, C. (1982). 'Frame Semantics' in Linguistics in the Morning Calm. Anon (ed.), Seoul: Hanshin; Linguistic Society of Korea. 111-138.
Fodor, J. (1975). *The language and thought.* Harvard University Press.
Frey, W., and Reyle, U. (1983). 'A prolog implementation of lexical functional grammar as a base for a natural language processing system.' *Proceedings of the first conference on European chapter of the Association for Computational Linguistics.* Association for Computational Linguistics.
Halvorsen, P. (1982) 'Lexical-functional grammar and order-free semantic composition.' *Proceedings of the 9th conference on Computational linguistics*-Volume 1. Academia Praha.
Halvorsen, P. (1983) 'Semantics for lexical-functional grammar.' *Linguistic Inquiry* 14.4 (1983): 567-615.
Hays, D. (1964). 'Dependency theory: A formalism and some observations'. *Language* 40. 511-525
Hendriks, H. (2001), 'Compositionality and model-theoretic interpretation,' *Journal of Logic, Language, and Information*, 10: 29–48.
Hockett, C. (1968). *The State of the Art.* Mouton and Company, the Hague.
Hoeksema, J. (1996) 'Floating Quantifiers, Partitives and Distributivity.' *Partitives: studies on the syntax and semantics of partitive and related constructions* 14: 57.
Hodges, W. (1998). 'Compositionality is not a problem.' In *Logic and Logical Philosophy* - Volume 6, 7-33.
Hodges, W. (2001). 'Formal features of compositionality.' *Journal of Logic, Language and Information* 10.1: 7-28.
Hodges, W. (2012). 'Formalizing the relationship between meaning and syntax.' In Werning, M., Hinzen, W, and Machery, E, (eds). (2012). *The Oxford handbook of compositionality.* Oxford University Press.
Hudson, R. (1984). *Word Grammar.* Oxford: Blackwell.
Hudson, R. (1998). *English Grammar.* Oxford: Blackwell.
Jäger, G., and Rogers, J. (2012). 'Formal language theory: refining the Chomsky hierarchy.' *Philosophical Transactions of the Royal Society B: Biological Sciences* 367.1598: 1956-1970.
Janssen, T. (1996). 'Compositionality.' University of Amsterdam.
Janssen, T. (2012). 'Compositionality: its historical context.' In Werning, M., Hinzen, W, and Machery, E, (eds). (2012). *The Oxford handbook of compositionality.* Oxford University

Press.
Joshi, A., and Schabes, Y. (1991) 'Tree-adjoining grammars and lexicalized grammars.' University of Pennsylvania.
Kahane, S. 'Why to choose dependency rather than constituency for syntax: A formal point of view.' University Paris Oest Nanterr & CNRS.
Kamp, H. (1981). 'A theory of truth and semantic representation'. In: J.A.G. Groenendijk, T.M.V. Janssen, and M.B.J. Stokhof (eds.), *Formal Methods in the Study of Language.* Mathematical Centre Tracts 135, Amsterdam. Pp. 277-322.
Kaplan, R., and Bresnan, J. (1982). 'Lexical-functional grammar: A formal system for grammatical representation.' *Formal Issues in Lexical-Functional Grammar*: 29-130.
Karttunen, L. (1986). 'Radical lexicalism'. CSLI.
Kay, P and Michaelis, L. forthcoming. 'Constructional Meaning and Compositionality'. In C. Maienborn, K. von Heusinger and P. Portner (eds.), *Semantics: An International Handbook of Natural Language Meaning.* Berlin: Mouton de Gruyter.
Lahav, R. (1989). 'Against compositionality: the case of adjectives.' *Philosophical studies* 57.3: 261-279.
Lambek, J. (1958). 'The mathematics of sentence structure.' *The American Mathematical Monthly* 65.3: 154-170
Langendoen, D. and Postal, P. (1985). *The Vastness of Natural Languages.* Basil Blackwell.
Lesmo, L, and Robaldo, L. (2006). 'Dependency tree semantics.' *Foundations of Intelligent Systems.* Springer Berlin Heidelberg. 550-559.
Luuk, E. and Luuk, H. (2011). 'Natural language - no infinity and probably no recursion'.
Mohri, M., and Sproat, R. (2006). 'On a common fallacy in computational linguistics.' *SKY J Ling* 19: 432-439.
Montague, R. (1970c). 'Universal Grammar'. *Theoria*, 36: 373–398. Reprinted in Thomason (ed.) 1974, pp. 7–27.
Montague, R. (1973) 'The proper treatment of quantification in ordinary English.' *Approaches to natural language.* Springer Netherlands. 221-242.
Pagin, P., and Westerståhl, D. (2010) 'Compositionality I: Definitions and variants.' *Philosophy Compass* 5.3: 250-264.
Pagin, P., and Westerståhl, D. (2010) 'Compositionality II: Arguments and problems.' *Philosophy Compass* 5.3: 265-282.
Partee, B. (2011). 'Formal Semantics: Origins, Issues, Early Impact,' *The Baltic International Yearbook of Cognition, Logic and Communication* 6: 1 - 52
Pelletier, F. (1994) 'The principle of semantic compositionality.' *Topoi* 13.1: 11-24.
Pinker, Steven. (2010) *The language instinct: How the mind creates language.* HarperCollins.
Pullum, G. and Scholz, B. (2010). 'Recursion and the infinitude claim'. In Harry van der Hulst (ed.), *Recursion in Human Language (Studies in Generative Grammar 104), 113-138. Berlin: Mouton de Gruyter.*
Rambow, O., and Aravind J. (1997) 'A formal look at dependency grammars and phrase structure grammars, with special consideration of word-order phenomena.' *Recent trends in meaning-text theory* 39: 167-190.

Robinson, J. (1970). 'Dependency structures and transformational rules'. Language 46. 259-285

Sag, I, Wasow, T and Bender, E. (2003). *Syntactic Theory: a formal introduction, Second Edition.* Chicago: University of Chicago Press.

Shieber, S. (1987) 'Evidence against the context-freeness of natural language.' *The Formal complexity of natural language.* Springer Netherlands, 1987. 320-334.

Stabler, E. (1999). 'Formal grammars'. In Robert A. Wilson and Frank C. Keil, (eds.), *The MIT Encyclopedia of the Cognitive Sciences* , 320–322. Cambridge, MA: MIT Press.

Steedman, M.(1992). 'Categorial grammar.' University of Pennsylvania.

Stojanovic, I. (2011). 'Situation semantics.' *Introduction to the Philosophy of John Perry* (ed.) Newen, A., van Riel, R.

Szabó, Z (2007). 'Compositionality.' *The Stanford Encyclopedia of Philosophy* (Fall 2013 Edition), Edward N. Zalta (ed.), forthcoming URL
= <http://plato.stanford.edu/archives/fall2013/entries/compositionality/>.

Szabó, Z. (2012). 'The case for compositionality.' In Werning, M., Hinzen, W, and Machery, E, (eds). (2012). *The Oxford handbook of compositionality.* Oxford University Press.

Tesniere, L.(1959). *Elements de Syntaxe Structurale.* Editions Klincksieck, Paris.

Tiede, H.J., and Stout, L.N. (2008). 'Recursion, Infinity and Modeling'.

Tomasello, M. (2000). The item-based nature of children's early syntactic development. *TRENDS in Cognitive Sciences* - Vol. 4, No.4 *Relation*, 10(1.90), 4440.

Van Benthem, J., Groenendijk, J., de Jongh, D., Stokhof, M., and Verkuyl, H. (1991). *Logic, Language and Meaning, Volume II: Intensional Logic and Logical Grammar.* University of Chicago Press.

van Eijck, J. and H. Kamp. (1997). 'Representing discourse in context'. In J. van Benthem and A. ter Meulen (eds.), *Handbook of Logic and Language.* Amsterdam: Elsevier Science, 179-237.

Werning, M. (2005). 'Right and wrong reasons for compositionality.' *The compositionality of meaning and content* 1: 285-309.

Werning, M., Hinzen, W, and Machery, E, (eds). (2012). *The Oxford handbook of compositionality.* Oxford University Press.

Zeevat, H. (1989) ‘A compositional approach to Discourse Representation Theory,’ *Linguistics and Philosophy*, 12: 95–131.

Zeijlstra, H. (2012). 'There is only one way to agree'. Paper presented at GLOW 33 in Wroclaw

Zeijlstra, H. Expressing Negation, in preparation.