Universitat Politècnica de Catalunya
Departament de Ciències de la Computació

Algebraic Dependency Grammar

Carles Cardó

PhD Thesis
in Computing

Directors
Glyn V. Morrill
Oriol Valentín

Barcelona, January 2018

PhD thesis in Computing
Algebraic Dependency Grammar
Carles Cardó
`cardocarles@gmail.com`
(versió final, 11/1/2018)

Universitat Politècnica de Catalunya,
Departament de Ciències de la Computació
`http : //www.cs.upc.edu`

Nota impremta: primera pàgina en color, la resta blanc i negre (amb grisos)
Encuadernació: 170mm × 240mm (d'un A4 escalar a 92%)

Endreça i Agraïments

Aviat farà quatre anys que vaig anar un tarda —xafogosa, ho recordo— al despatx del Glyn Morrill per consultar la possibilitat de fer el doctorat. Va venir a dir-me que un doctorat és una cosa llarga i feixuga, i deia bé. Més tard em va presentar el que seria el meu altre director de la tesi, l'Oriol Valentín. Ara a punt d'acabar el viatge, he de dir que des del primer minut tots dos han esdevingut uns consellers infal·libles. El Glyn, amb la seva eloqüència minimalista feta de paraules meticulosament endreçades i d'efecte persistent. L'Oriol, amb el comentari agut, exacte i enlluernador. La pluja i el llamp fan la tempesta perfecta —excusi el lector la literatura baratíssima. El primer de tot que els he d'agrair és haver acceptat el tema de la meva tesi, la gramàtica de dependències, que, bo i sent cosina de la seva investigació, la gramàtica categorial, la parafernàlia matemàtica n'és del tot diferent. De fet els he d'estar agraït per moltes altres coses, com ara les incontables hores de correcció, però el que més em satisfà és saber que he guanyat dos amics.

També voldria fer arribar la meva gratitud al J. L. Balcázar i al R. Gavaldà per haver-me ajudat amb les presentacions. A la L. Wilson per ajudar-me a millorar el meu anglès pedestríssim i al J. Campos per ajudar-me a tocar el "piano".

En el terreny personal acuso amb un dit inquisidor dos vells amics. D'ells és la culpa que el meu cap hagi estat tres anys traient fum com una cafetera. Primer la S. Montero, perquè va tenir la paciència de corregir-me, per amor a l'art, un lleugeríssim llibre d'unes quatre-centes pàgines. Gràcies a aquest treball, que em va fer de carta de presentació, vaig poder accedir al programa de doctorat. Només recordar-li que un llamàntol viu feliç al fons del mar esperant el seu fatal destí. Segonament acuso el F. Moncunill. A ell li dec una simple empenteta, no res, un copet a l'esquena donat amb gràcia i al moment escaient: —escolta, i per què no fas el doctorat —va dir-me— ja que t'agrada tant el tema? El cuc em va rosegar i vet-ho aquí. Com que sé que li plau sempre de llegir l'última paraula dels llibres, li estalviaré feina i li diré que l'última d'aquest és "mollusks". Apa, a veure com es lliga això amb la lingüística matemàtica.

Em queda el deute més gran amb la M. Maluenda. Companya de viatge, gràcies per la paciència, pels ànims i per escoltar-me infatigablement. Has demostrat que tendeixes a l'∞ quan jo tendeixo a 0. Finalment la meva endreça als companys silents. A mon pare, de qui encara recordo i esgrano paraules. A dues bèsties que habiten la part més simple de la bi-jerarquia, la Greta i el Dido. I també a l'àlber, que sotjant per la finestra tot confegint la tesi, l'he vist mudar la fulla tres voltes.

Carles Cardó
Aiguamúrcia, juny del 2017

P.D.:

Many thanks to the three anonymous referees for their opinions, advice, suggestions and references. Their help has contributed greatly to the improvement of this thesis.

Carles Cardó
Aiguamúrcia, desembre del 2017

Abstract

We propose a mathematical formalism called *Algebraic Dependency Grammar* with applications to formal linguistics and to formal language theory. Regarding formal linguistics we aim to address the problem of grammaticality with special attention to cross-linguistic cases. In the field of formal language theory this formalism provides a new perspective allowing an algebraic classification of languages. Notably our approach suggests the existence of so-called anti-classes of languages associated to certain classes of languages.

Our notion of a dependency grammar is as of a definition of a set of well-constructed dependency trees (we call this *algebraic governance*) and a relation which associates word-orders to dependency trees (we call this *algebraic linearization*).

In relation to algebraic governance, we define a *manifold* which is a set of dependency trees satisfying an agreement condition throughout a *pattern*, which is the algebraic form of a collection of syntactic addresses over the dependency tree. A boolean condition on the words formalizes the notion of agreement.

In relation to algebraic linearization, first we observe that the notion of *projectivity* is quintessentially that certain substructures of a dependency tree always form an interval in its linearization. So we have to establish well what is a substructure; we see again that patterns proportion the key, generalizing the notion of projectivity with recursive linearization procedures.

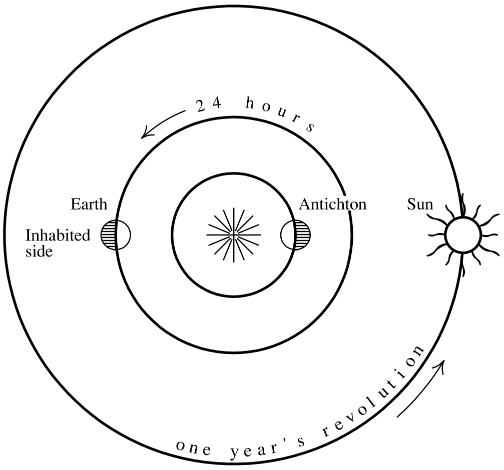
Combining the above modules we have the formalism: an *algebraic dependency grammar* is a manifold together with a linearization. Notice that patterns sustain both manifolds and linearizations. We study their interrelation in terms of a new algebraic classification of classes of languages.

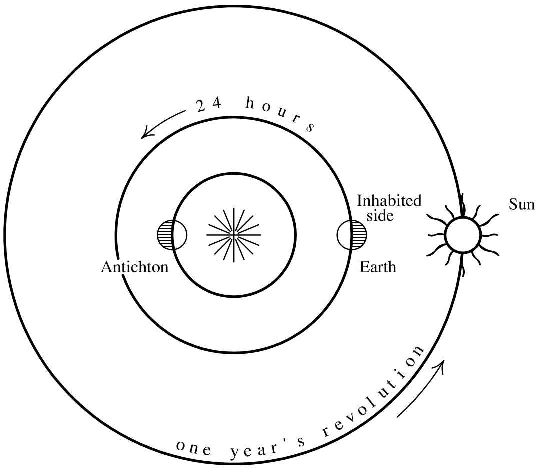
We highlight the main contributions of the thesis. Regarding mathematical linguistics, algebraic dependency grammar considers trees and word-order different modules in the architecture, which allows description of languages with varied word-order. Ellipses are permitted; this issue is usually avoided because it makes some formalisms non-decidable. We differentiate linguistic phenomena structurally by their algebraic description. Algebraic dependency grammar permits observance of affinity between linguistic constructions which seem superficially different.

Regarding formal language theory, a new system for understanding a very large family of languages is presented which permits observation of languages in broader contexts. We identify a new class named *anti-context-free languages* containing constructions structurally symmetric to context-free languages. Informally we could say that context-free languages are well-parenthesized, while anti-context-free languages are cross-serial-parenthesized. For example copy languages and respectively languages are anti-context-free.

Algebraic Dependency Grammar

Carles Cardó





(. . .) and also invisible to us was another planet, Antichthon, or Counter-Earth, for this revolved within our orbit, and also in twenty-four hours. It was added to the system, because the addition of Earth as a heavenly body spoiled the sacred number of seven, but by adding Antichthon, and counting the star-sphere as another, the total was brought up to ten, another sacred number.

Dante and the Early Astronomers
Mary Acworth Orr (1807)

Contents

II THEORY 53

Part I

INTRODUCTION

1

Introduction

This first chapter constitutes a guide to reading the thesis. We present a historical review of formal linguistics and its main concepts. We introduce the basic features of the family of dependency grammar and we review recent work in dependency theories.

The purpose of this thesis is to introduce a new dependency based formalism from an algebraic perspective which aspires to account for natural languages transversally; however we will see that this has also ramifications in the panorama of formal languages. We expose in brief the elements and the architecture of the model. The most important contributions are presented together with the structure of the chapters.

In order to display linguistic examples in the future we need to fix some conventions for dependency analysis and the notation for syntactic functions. Finally we summarize the necessary mathematical concepts and fix also notations to understand subsequent chapters.

1.1 Some Topics on Formal Linguistics

Before we consider some basic topics on dependency grammar, let us zoom out and comment some general aspects considered significant in formal linguistics and that will arise throughout the thesis. By way of introduction we make a rapid historical review.

1.1.1 The Productive Decade: A Very Short Note on the Chronology of Formal Linguistics[1]

In the mid-twentieth century linguistics was shaped by two formal theoretical approaches. Linguistics was a field that had previously been formed by linguistic structuralism, philosophy, logic and, in general, mathematics in the first half of the century.

[1] In this note we just relate innovations and contributions in formal syntax and semantics, not other linguistic issues such as phonology or pragmatics. Further, we just review the last half of the XX century, and the XXI century. For a more detailed chronology see (Kruijff, 2002).

The common denominator of the innovations was the application of formal tools to analysis of the mechanisms of natural language.

On the one hand in 1957 N. Chomsky wrote his *Syntactic Structures* (Chomsky, 1957). Today the imprint of that work is still palpable in linguistics and computer science. His hierarchy was one of the first road maps in formal languages, and its four classes are the four cardinal points with respect to which theorists still orient themselves. It is widely assumed that the class of natural languages inhabits a level between the class of context-free languages and the context-sensitive languages.

Chomsky has followed a long trajectory: from the early rewriting systems, through *Transformationalism* (TG), *Principles and Parameters* or *Government and Binding* (GB), and finally to the *Minimalist Program* (MP) in the 1990's.[2] Over this time linguistics has accumulated a number of derivative theories such as *Generalized Phrase Structure Grammar* (GPSG) or its successor, *Head-Driven Phrase Structure Grammar* (HPSG), among others. Regardless of the details these approaches are frequently called *constituency grammars* because of the way one groups words of a sentence.

On the other hand, in the beginning of the 1950's Y. Bar-Hillel picked up an old and simple idea from the logician K. Adjukiewicz (1935) and invented a combinatorial system, or as he named it, *a quasi-arithmetical notation for syntactic description*, (Bar-Hillel, 1953). Just a few years later J. Lambek independently connected this idea to logic in (Lambek, 1958). This formulation captured a grammar elegantly in a few deductive rules giving what is now named *Lambek Calculus* (L). Nevertheless this attempt did not account for discontinuities of natural language. Recently that problem was addressed preserving the initial elegance of Lambek Calculus by Morrill, Valentín and Fadda (Morrill et al., 2011) in the form of the *Displacement Calculus* (D). This research continues actively.[3]

However, by way of a third route, at the same time that Chomsky and Lambek produced their early works, there was published posthumously the *Éléments de syntaxe structurale* of L. Tesnière (1959). His style falls short of formality, but contains a deep and interesting intuition: the *stemma*, an unfortunate denomination which nobody would use again (in modern literature, *dependency tree*).[4] This notion was a very compact and succinct way to represent syntactic structure. Unfortunately his work remained unnoticed, perhaps overshadowed by the other authors, but the ideas it contained flourished later, being transformed into what is now called *Dependency Grammar*.

Intermittently in the following years several authors formally developed the first steps of the *Éléments*. Regarding the weak generative capacity Gross (1964), Hays (1964), Gaifman (1965) and Robinson (1970) established the equivalence of certain *dependency systems* with context-free languages. Since context-free grammars are

[2] Chomsky talks simply of *generative grammars* for any of his works, cf. (Chomsky, 1995).

[3] See Morrill (2007) for a chronicle of type logical grammar.

[4] The dependency based approach enjoys its own prehistory. See (Osborne and Kahane, 2015) for some examples of rudimentary dependency structures well before Tesnière.

considered insufficient for natural languages (in fact Chomsky already claimed this insufficiency from his first works); this could be one of the reasons why dependency grammar had been temporarily sidelined.

More in relation to linguistic issues, there appeared the stratificational theories versus the first mono-stratal *stemmas*: the *Functional Generative Grammars* (FGD) of Sgall et al. (1969) and later the *Meaning-Text Theory* (MTT) of Gladkij and Mel'čuk (1975), amongst others.

The new millennium has revitalized and exploited the Tesnerian ideology with works which will be commented later, when we explicate our contributions to this third route. All in all, the 1950's in the past century was a productive decade.

1.1.2 Some Topics of Formal Linguistics

Finiteness and Decidability

A *formal grammar* $\mathscr{G}$ is a well-defined and finite formalism describing fundamental phenomena of linguistics such as grammaticality, semantics, first language acquisition, second language acquisition or existence of universal principles, apart from computer applications. Perhaps, the most basic of these fundamental problems is that of grammaticality, which consists in establishing a set of grammatical strings, $\mathscr{L}(\mathscr{G}) \subseteq \Sigma^*$.

By *finite* here we mean a mathematical system (a finitely generated algebraic structure, a finite axiomatic system, an algorithm or any other mathematical object finitely described) able to capture the infinite combinatorial capacity of language, i.e. $\mathscr{L}(\mathscr{G})$, using a finite set of resources. This constitutes a widely accepted general principle for formal linguistics which goes back to the XIX century, and perhaps before, when the philosopher and linguist, Humboldt, stated that languages "make infinite use of finite means", *apud* (Chomsky, 1965).

But strict finiteness, as stipulated above, is a too vague and weak condition. *Decidability* is a more reasonable feature which a linguistic formalism should enjoy.[5] This is based in the human skill to decide when a sentence is grammatically correct and when it is not, which establishes a well-delimited set of grammatical occurrences or strings.[6]

There are two possible manners of formulating this problem: *universal recognition* vs. *fixed language recognition*:

- (*Universal*) Given a grammar $\mathscr{G}$ (in some grammatical framework) and a string x, is $x \in \mathscr{L}(\mathscr{G})$?

- (*Fixed Language*) Given a string x, is $x \in L$, where L is some independently specified set of strings L?

[5] Sometimes the constraint of finiteness in the sense of Humboldt is such a weak hypothesis that the term "finiteness" is almost taken as synonymous of decidability.

[6] This is an idealization, because when one tests speakers with confusing sentences, the frontier of acceptance turns out fuzzy and there seems to appear a grammatical gradient.

The problems are distinct. The first contains two variables: the grammar and an input string. The second contains just one variable, the string. This means that the fixed language recognition problem can employ any technique to solve the membership. There arises the question which of them is most adequate for linguistics. Universal problems study entire grammatical families and they consider the worst case grammar as input and hence they are harder than fixed language problems.

Speakers of a language do not need to have knowledge of a wide class of grammars, but just one grammar and, by using some internal brain mechanism, are able to develop fast processing. So, it could seem that fixed language problems are more pertinent for linguistic inquiry.[7]

Notwithstanding, finiteness and decidability are still weak conditions. In a majority of cases a hearer can decide quickly the grammaticality of a sentence. This leads some authors to consider the hypothesis that natural languages are recognizable in polynomial time in the length of the sentence. But there is no longer total consensus here. We resume this issue later.

Phrase Structure Grammars

Early computational approaches to linguistics came from the direct manipulation of strings of symbols. Based on previous notions of the logician E. Post and the mathematician A. Thue, Chomsky introduced a family of rewriting grammars, now sometimes called *phrase structure grammars*, consisting in: a finite set of *non-terminal* symbols or *variables*, $P, Q, R, S, \ldots$, with a distinguished *start* symbol; a finite set of *terminal symbols* or *letters*, $a, b, c, \ldots$; a finite set of *rewriting rules* or *productions* consisting in expressions as $X \rightarrow Y$, where X and Y are strings of letters and variables. A rule can be applied over a string of letters and variables when the scheme described by X occurs in the string. Then we can substitute X by Y in the string. A string of letters is *derivable* if we can obtain it by applying rules from the start symbol. According to the form of the productions Chomsky separated four classes of grammars:

> Type 3 (or *regular grammars*, RGG): when the X only contain a variable and Y is of the form aP, where a is a letter and P is a variable.[8]
>
> Type 2 (or *context-free grammars*, CFG); when the X only contains a variable.
>
> Type 1 (or *context-sensitive grammars*, CSG); when the X and Y are of the form aPb.
>
> Type 0 (or *unrestricted*); there is no constraint.

[7] For further discussion see e.g. (Barton et al., 1987).

[8] Or of the form Pa, but all the rules in the same form. It is provable that the same classes of languages are obtained.

The derivational history of a string in the case of type 2 grammars can be encoded by a tree. In this formalism the syntactic structure (the derivational tree) is the byproduct of a computation. This contrasts with other formalisms where the computational protagonist is the syntactic structure itself, as in the case of dependency grammars.

Weak Capacity

The *weak capacity of a grammar* is the set of sentences licensed by the grammar, and, by extension, the *weak capacity of a class of grammars* is the class of languages generated by the grammars. Every kind of Chomskian grammar generates a class of languages: *regular languages*, RG (which coincide with the class of languages recognizable by *automata* or the class of languages definable by *regular expressions*); *context-free languages*, CF; *context-sensitive languages*, CS. In particular unrestricted grammars generate the class of *recursively enumerable languages*, $\mathsf{RecEnum}$, i.e. those languages the strings of which a Turing Machine can enumerate. The four formalisms yield an increasing hierarchy of classes of languages:

$$\mathsf{RG} \subsetneq \mathsf{CF} \subsetneq \mathsf{CS} \subsetneq \mathsf{RecEnum}.$$

By an empirical idealization we can assume a class NT of all possible natural languages. There arose the question of where the class NT of natural languages is placed. Early on theorists realized that context-free grammars are too inexpressive, but the convincing proof did not arrive before 80's. A paper of Shieber (1985) closed a long debate about the context-freeness of natural languages initiated in the 50's, (Chomsky, 1956).[9] Using elementary techniques of formal language theory, Shieber showed that certain sentences of Swiss-German could be reduced to a non-context-free language. In the opposite side, context-sensitive grammars were too powerful. These grammars were proved to be very near to Turing machines, and these last are clearly excessive for natural language.[10] The scenario was then:

$$\mathsf{RG} \subsetneq \mathsf{CF} \subsetneq \mathsf{NT} \subsetneq \mathsf{CS} \subsetneq \mathsf{RecEnum}.$$

Under the suggestive title "Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions" Joshi (1985) proposed guiding conditions which an adequate formalism for natural language should satisfy (of course subsuming context-freeness): "1. limited cross-serial dependencies, 2. constant growth, 3. polynomial parsing". Kallmeyer (2010) interprets these conditions as properties of classes of languages: a set $\mathcal{L}$ of languages is *mildly context-sensitive* iff

[9] See also the paper of Pullum and Gazdar (1982), for a state of the art in the debate just a few years before the paper of Shieber (1985).

[10] More exactly, context-sensitive grammars are equivalent to *linear bounded automata*, i.e. Turing machines restricted to a bound portion of tape, see (Hopcroft and Ullman, 1979).

(a) $\mathsf{CF} \subseteq \mathcal{L}$;

(b) $\mathcal{L}$ can describe cross-serial dependencies: there is an $n \geq 2$ such that $L_{k\text{-copy}} = \{x^k \mid x \in \Sigma^*\} \in \mathcal{L}$ for all $k \leq n$;

(c) the languages in $\mathcal{L}$ are polynomially parsable, i.e. $\mathcal{L} \subseteq \mathsf{PTIME}$;

(d) the languages in $\mathcal{L}$ have the *constant growth property*.

The first point is clear. Regarding the second point the definition only requires generation of the $L_{2\text{-copy}}$ language, but permits that all the k-copy languages could be generated. Regarding time complexity, here PTIME is the class of languages with fixed language recognition problem solvable in polynomial time. Although some linguists do not insist on this requirement, computationally it is a desirable property. The last condition means informally that the lengths of strings of languages in $\mathcal{L}$ cannot be too far apart. Weir (1988) defines the *constant growth property* as follows: a language L has the *constant growth property* iff there is a constant $c_0 > 0$ and a finite set of constants $C \subset \mathbb{N}$ such that for all $x \in L$ with $|x| > c_0$, there is a $x' \in L$ with $|x| = |x'| + c$ for some $c \in C$.

There is another notion of growth which is alleged for natural languages. We set $\Sigma = \{a_1, \ldots, a_k\}$. The *Parikh mapping* is the mapping $p : \Sigma^* \longrightarrow \mathbb{N}_+^k$ defined by $p(x) = (|x|_{a_1}, \ldots, |x|_{a_k})$. A set in $\mathbb{N}_+^k$ is *semi-linear* iff it is a finite union of sets of the form $\{p_1A_1 + \cdots + p_nA_n + B \mid p_1, \ldots, p_n \in \mathbb{N}_+\}$ for some $n \geq 0$ and $A_1, \ldots, A_n, B \in \mathbb{N}_+^k$. A language $L \subseteq \Sigma^*$ is *semi-linear* iff $p(L)$ is a semi-linear set. This is more restrictive notion than the Weir one, see (Kallmeyer, 2010).

With Joshi's conditions there commenced a difficult search to establish the precise place of natural language, and a discussion ensued on the four above points. Several formalism were proposed: *indexed grammars* IG, Aho (1968);[11] *linear indexed grammars* LIG, Gazdar (1988)[12]; tree adjoining grammars TAG, Joshi (1985); *q-linear context-free rewriting systems* q-LCFRS, Vijay-Shanker et al. (1987); *q-multiple context-free grammars* q-MCFG, Seki et al. (1991). See Fig. 1.1, which contains a diagram of inclusions for these formalisms, with some examples of specific languages (inclusions are strict). Some of these formalisms are equivalent, for example linear indexed grammars are equivalent to tree adjoining grammars, and q-LCFRS are equivalent to q-MCFG, for any q. And some of the formalisms are subsumed by others, for example linear context-free rewriting systems are more expressive than linear indexed grammars.

As pointed out by Kallmeyer (2010), LCFRS and MCFG are important for natural language processing. They can account discontinuity phenomena and are weakly equivalent to other more linguistic-based formalisms, such as *minimalist grammars*

[11] Although indexed grammars were proposed before the paper of Joshi we commented here according to its relation with the other grammars.

[12] See also (Weir and Joshi, 1988).

MG, (Stabler, 1997), the formalization of the Chomskian ideology in the *minimalist program*, (Chomsky, 1995); or *finite-copying* LFG *(lexical functional grammar)*, (Kaplan and Bresnan, 1982).

Some works suggest that natural languages are beyond LCFRS. Becker et al. (1992) shows that *scrambling in German* can be reduced by regular intersections and homomorphisms to the p-counting language $L_{p\text{-count}} = \{a_1^n \cdots a_p^n \mid n \in \mathbb{N}_+, a_1, \ldots, a_p \in \Sigma\}$ for a sufficiently large p, which, using a pumping lemma by Seki et al. (1991), shows that natural language cannot be described by any LCFRS.

Recently Salvati (2011) proved the surprising result that the language $L_{\text{mix}} = \{x \in \{a, b, c\}^* \mid |x|_a = |x|_b = |x|_c\}$ can be generated by a 2-multiple context-free grammar. That language is supposed to be an extreme case without linguistic relevance, whereby it should not be derivable by a constrained formalism for natural languages. [13]

Some authors discuss the property of growth. Boullier (1999) considers the Chinese number language which exhibits a non-constant growth property, and hence non-mild context sensitivity. This lead him to define *range concatenation grammars* RCG, Boullier (2004). Another phenomenon which is alleged to be beyond the constant growth property is the genitive construction of Old Georgian, Michaelis and Kracht (1997). The accumulated genitive infixes grow quadratically.

Furthermore, another construction in Yoruba, a Nigerian language, suggests that natural languages would tolerate exponential growths. Parallel Multiple Context-Free Grammars PMCFG allows to construct the language $L_{\text{expo}} = \{a^{2^n} \mid n \geq 0\}$; see Clark and Yoshinaka (2012).

All these phenomena are argued to be truly syntactic constructions (actually, Old Georgian is a dead language and there are no speakers to contrast the hypothesis). However as Clark and Yoshinaka (2012) comment: “There is a broad consensus that natural language string sets are semilinear, and so attention has focused largely on properties of formalisms that generate semilinear languages. However there are a number of cases where linguistic data suggest that there are richer processes involved, processes that either require or might benefit from a more powerful formalism.”

Structural Adequacy and Strong Capacity

The considerable efforts focusing on weak capacity of natural languages sometimes make us forget the question of the structural adequacy of grammar formalisms. The

[13] See (Bach, 1981, 1988; Salvati, 2011; Kanazawa and Salvati, 2012).

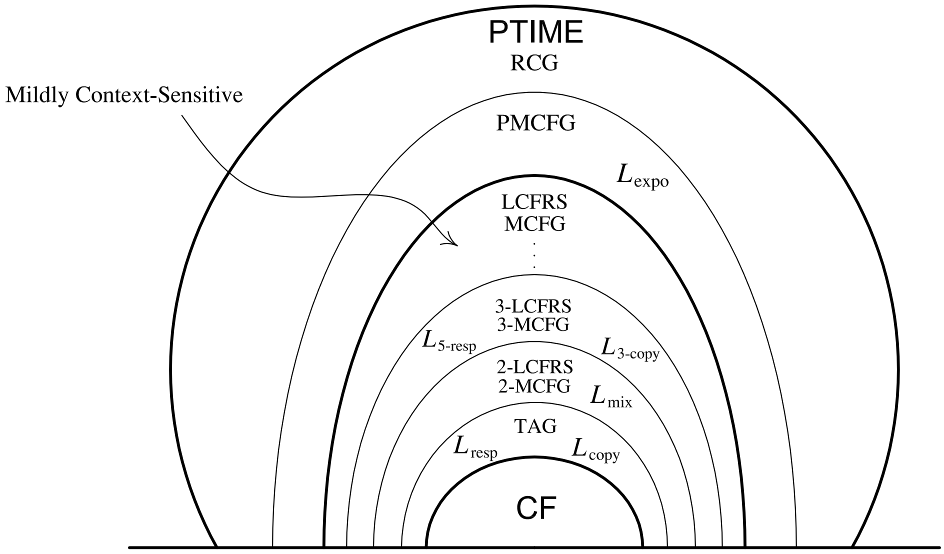


Figure 1.1: hierarchy of grammars (and classes of languages) in relation to mild context-sensitivity. Figure adapted from (Kallmeyer, 2010).

following context-free grammar:

$$\begin{cases} S \longrightarrow NP_{\text{singular}} + V_{\text{singular}} + NP \\ S \longrightarrow NP_{\text{plural}} + V_{\text{plural}} + NP \\ NP_{\text{singular}} \longrightarrow \text{Mary, John, } \ldots \\ NP_{\text{plural}} \longrightarrow \text{girls, boys, } \ldots \\ V_{\text{singular}} \longrightarrow \text{eats, grows, } \ldots \\ V_{\text{plural}} \longrightarrow \text{eat, grow, } \ldots \\ NP \longrightarrow \text{apples, tomatoes, } \ldots \end{cases}$$

generates sentences like *John grows tomatoes* with a correct agreement of number between the subject and the verb. Under the point of view of weak capacity there is nothing to say. In Chomskian terms, it is said that the grammar is a purely *generative* engine, where "generative" means enumerative.

However if we address our interest rather to obtain linguistic explications, then the above grammar seems tricky or artificial; we tend to interpret the variable symbols with some syntactic meaning, but in fact computationally they are simply auxiliary symbols. Notice that in the derivation the number of subject and the verb is decided before the actual subject and verb. Furthermore, one can design a context-free grammar to make long distance agreement by introducing more auxiliary variables. But this rewriting style approach is linguistically misleading because it multiplies the number of variables and rules.

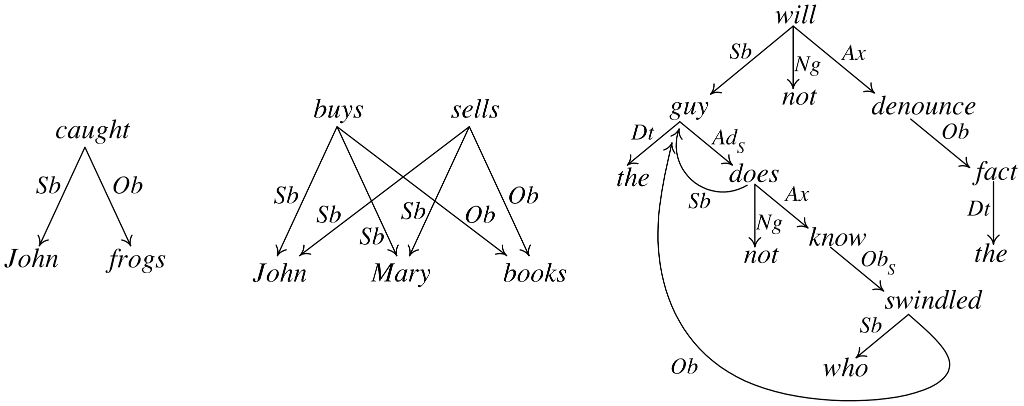


Figure 1.2: a sample of dependency structures.

Chomsky introduced the concepts of *strong* and *weak* capacity of a grammar, (Chomsky, 1965). The strong capacity is defined by the set of parse structures which a grammar generates, and similarly the strong capacity of a class of grammars is the class of sets of parse structures, see (Bach and Miller, 2003). This permits differentiation of two grammars yielding the same language but with different structural devices. Unfortunately, while for weak capacity we can compare classes of languages from different formalisms, strong capacity depends on the format of the syntactic structure which only allows us to compare grammars inside a same formalism.

In sum, the syntactic structure is also an element of study in itself, even more in dependency grammar. This appreciation is limited by the obvious practical fact that syntactic structures are invisible, while string sentences are not. So the work of giving an adequate structural description of natural language is harder than that of giving a mechanism which proportions the correct sentences as mere strings.

1.2 Dependency Grammar

Dependency based approaches assume as a foundational notion that there is a sort of linguistic structure called *dependency structure*. Very informally a dependency is a triplet formed by two words and a function or role which relates them. Graphically:

$$\mathit{caught} \xrightarrow{\mathit{object}} \mathit{frogs}$$

It is said that the word “caught” *governs* the word “frogs”. Joining together a set of dependencies we have a dependency structure. So for dependency grammarians the linguistic structure is a relational system. Fig. 1.2 shows a sample of dependency structures.

Dependency structures are very versatile objects which according to their topology can be used for different purposes: for a semantic representation, for a syntactic repre-

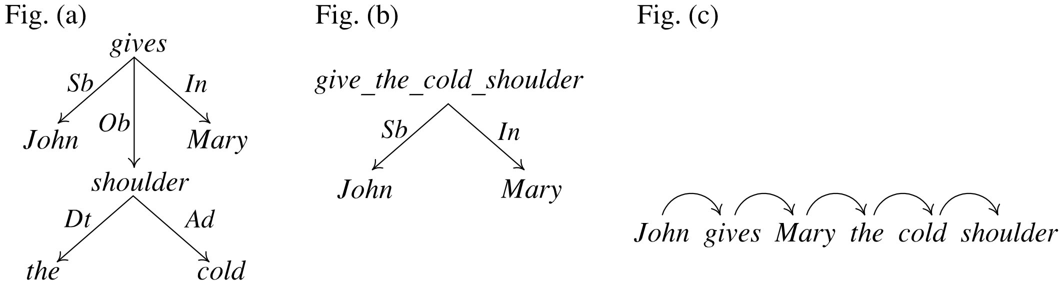


Figure 1.3: (a) syntactic representation for the sentence *John gives Mary the cold shoulder*; (b) semantic representation; (c) representation of the word-order.

sentation or even for a word-order representation; see Fig. 1.3 for some examples. Thus a dependency grammar is able to use different modalities of dependency structures orchestrated according the convenience of each author.

In order to discuss general concepts let us fix a very short glossary. We just comment a few words of widespread usage; the more specific glossary will be introduced in the course of the text.

By *word-order* we mean a string of words, or more abstractly all those syntactic aspects concerning the order of words in a sentence. For a formal notion of word order we will use the term *linearization*. By *syntactic functions* or *grammar roles* we mean arguments, syntactic arguments or semantic arguments, or more in general any label on the edges on a dependency structure. In the old Tesnèrian denomination: *actants*. The relation of *domination* is the reflexive and transitive closure of the governing relation. Some authors call *dependency structure* a dependency tree together a word order.[14] Here we prefer to use this terms for a general dependency set (without word-order) where its graphical representation can be a general graph. When the graph is tree-shaped we call it a *dependency tree*.

1.2.1 The Parameters of Dependency Grammar

The family of dependency models shows differences from the smallest cog to the most general architecture. For a brief exposition on the various models see e.g. (Debusmann, 2000). We make a list of variables to highlight the differences between models. We follow and adapt with some changes the exposition from Nivre (2005).[15]

> **Stratification.** This is an intuitive concept and there is no consensus on what it is. Stratum, level or layer means a component in the architecture of the model. The *Elements* of Tesnière was mono-stratal: his *stemma* was the centre of the

[14] For example (Kuhlmann, 2013).

[15] List of abbreviations: FGD, Functional Generative Grammar; MTT, Meaning Text Theory; TDG, Topological Dependency Grammar, XDG, eXtensible Dependency Grammar.

theory. However his approach left issues without treatment. FGD presented two components: a syntactic representation and a *tectogrammatical* representation. MTT exhibited a multi-stratal architecture, including structures for deep phonetics, surface morphology, deep morphology and semantics. The notion of stratum suggests a linear organization of the layers. The idea is still employed in TDG which is bi-stratal. In the XDG framework layers are not linearly allocated, so it would be more adequate to speak of modules (or *dimensions* in the XDG denomination).

Nodes Another variable, linguistic rather than structural, concerns nodes. Most theories agree that nodes on dependency trees must contain single words. However, in one direction, some approaches pad out nodes with phrases, and in the other direction, some authors decompose words in lexemes granting a deeper representation.

Grammar Roles. This issue is closely related to the architecture. Mono-stratal models must establish a set of grammar roles for the whole theory, while complex models are able to use several groups depending on the nature of each layer.

Shape. This concerns the shape of the syntactic structure. Many theories accept only tree-shaped structures. However XDG considers also directed acyclic graphs for semantic representations, see Fig. 1.3(c).

Projectivity. This is, perhaps, the most important issue in dependency grammar. A linguist draws dependency trees following the grammar roles (which sometimes is a very intuitive procedure) and then adds a linear order. Linguists speak of "continuity" to express that this relation must not break the syntactic substructures in the linear word order. There are several equivalent formalizations of this idea, for example: subtrees in the dependency tree must become continuous intervals in the linear order. Nonetheless, projective structures do not cover the totality of the sentences of natural language. For this reason other wider kinds of projectivity have been proposed.

1.2.2 Recent Work in Dependency Grammar

Topological Dependency Grammar (TDG) is a proposal of Duchier and Debusmann (2001). An analysis in TDG consists of two dependency structures. The first one, called the *syntax tree* or ID tree, contains a dependency tree capturing the syntactic dependencies (edges are labeled as *Subject*, *Object*, . . .) and a non-projective word-order (i.e.: admitting crossing edges in its representation). The second one, called the *topological tree* or LP tree, contains a dependency tree where the edges are now labeled by semantical roles; it also contains a projective word-order. In addition the LP tree contains a partial order on the tree. So this model can be considered bi-stratal (where every layer is a dependency structure). A number of constraints (captured by *principles*) on some parameters like *valencies, barriers, climbings, etc*, which act simultaneously on both layers license analysis.

Duchier and Debusmann and others have been developing these ideas in recent years: (Debusmann and Duchier, 2002*a*), (Debusmann and Duchier, 2002*b*), (Duchier, 2004), (Debusmann, Duchier, Koller, Kuhlmann, Smolka and Thater, 2004). In the later papers more structures (for example semantic directed acyclic graphs) have been added to account for richer phenomena. In the successor theory *eXtensible Dependency Grammar* XDG (Debusmann, Duchier and Kruijff, 2004), layers are not distributed sequentially and are renamed *dimensions*. This extension allows semantics to be tackled; for example multiple structures allow the scope ambiguity of quantifiers to be dealt with. See the evolution in (Debusmann et al., 2005), (Debusmann, 2007*a*), (Debusmann, 2007*b*).

In relation to semantic representation, recent works have introduced a module called *Abstract Meaning Representation* (AMR) which proportions a very deep semantic form which is not obtained directly from the word-order, (Tosik, 2015; Flanigan et al., 2014; Koller, 2015).

Some other recent works are focused not on the general architecture of a grammar, but on the dependency structures in themselves, as in for example (Kahane, 1997; Gerdes and Kahane, 2015; Kahane and Mazziotta, 2015). These works test the limits of the capability of syntactic representation of dependency trees and word-order and investigate variations on the theme.

Work of (Kuhlmann and Nivre, 2006; Kuhlmann and Möhl, 2007; Kuhlmann, 2010) investigates how rewriting grammars, such as TAG or LCFRS, can be translated to a dependency framework, via *lexicalization*. A grammar is lexicalized if each rewriting rule contains a non-terminal symbol (called the *anchor*). For example every CFG can be lexicalized via the Greibach normal form. A derivation in a lexicalized grammar is interpreted as inducing a set of dependencies.

Recently dependencies have been attracting attention thanks to the existent treebanks which permit the computational treatment and metric analysis of natural language.[16] However these trends go far from the idea of a underlying grammar and try to explain language from the direct observation of data. It is thought that such approaches are going to be more and more influential in coming years and probably the more classical point of view of language as a grammar and these empirical approaches will coexist.

[16] Consider in this connection the contributions made in the *International Conference on Dependency Linguistics* (Depling 2013, 2015, 2017).

1.3 The Formalism in a Nut-Shell

In the course of the following chapters we will introduce a new dependency based grammar formalism. We give here an overview which can be used as a reading guide. Fig. 1.4 shows the necessary elements in the apparatus and the organization. We explain them succinctly (key concepts are in boldface, secondary concepts, in italic).

1.3.1 Building Blocks: Syntagmata and Patterns

First of all we present the basic tool kit. Syntagmata and patterns can be considered as the bricks and mortar in order to construct the modules of the formalism. Syntagmata are a form to capture algebraically a dependency tree. We set a vocabulary $\Sigma = \{John, hunts, frogs, \ldots\}$ and a set of syntactic functions $\zeta = \{Sb, Ob, \ldots\}$. A **syntagma** is a mapping $S : \zeta^* \longrightarrow \Sigma \sqcup \{0\}$ where 0 is a null word without phonetic realization. The free monoid ζ^*, the elements of which are called *loci*, represent syntactic addresses. For example the locus $Dt{\cdot}Sb \in \zeta^*$ is read as “the determiner of the subject”. The mapping S assigns words of Σ to syntactic addresses in ζ^*. If we make null all the loci except a finite set then this captures a dependency tree.

In order to capture syntactic constraints we will need to control these syntactic addresses in some way: patterns do this. We consider the a times Cartesian product of the monoid ζ^*, that is $(\zeta^*)^a$, with the product taken component-wise. This is newly a (non-free) monoid the elements of which can be though as vectors of syntactic addresses (or loci). The whole vector represents a link among the components. A **pattern** is a product of *constants*, singletons of $(\zeta^*)^a$, and *submonoids* of $(\zeta^*)^a$. So patterns are certain subsets of vectors of addresses.

1.3.2 Modules: Algebraic Governance and Algebraic Linearization

Our notion of a dependency grammar is as of a set of well-constructed syntagmata or dependency trees (we call this *algebraic governance*) and a relation which associates linear orders to each dependency tree (we call this *algebraic linearization*).

We establish the algebraic governance part. Well-constructed dependency trees are captured by a manifold through the patterns. More specifically, a **manifold** is a set of syntagmata satisfying an agreement condition throughout a pattern. We will see that the well-construction of a syntagma can be established by agreeing its addresses and, furthermore, these agreements (vectors of syntactic addreses) form a pattern. We only need to formalize the notion of agreement which is simply made by a **boolean condition** on the words.

Now we consider algebraic linearization. First we notice that the notion of projectivity can be reread as that the substructures of a dependency tree always form a prosodic interval in its linearization. So if we take this as a lemma we only have to establish well what is a substructure. In order to do this we will see again that patterns proportion the

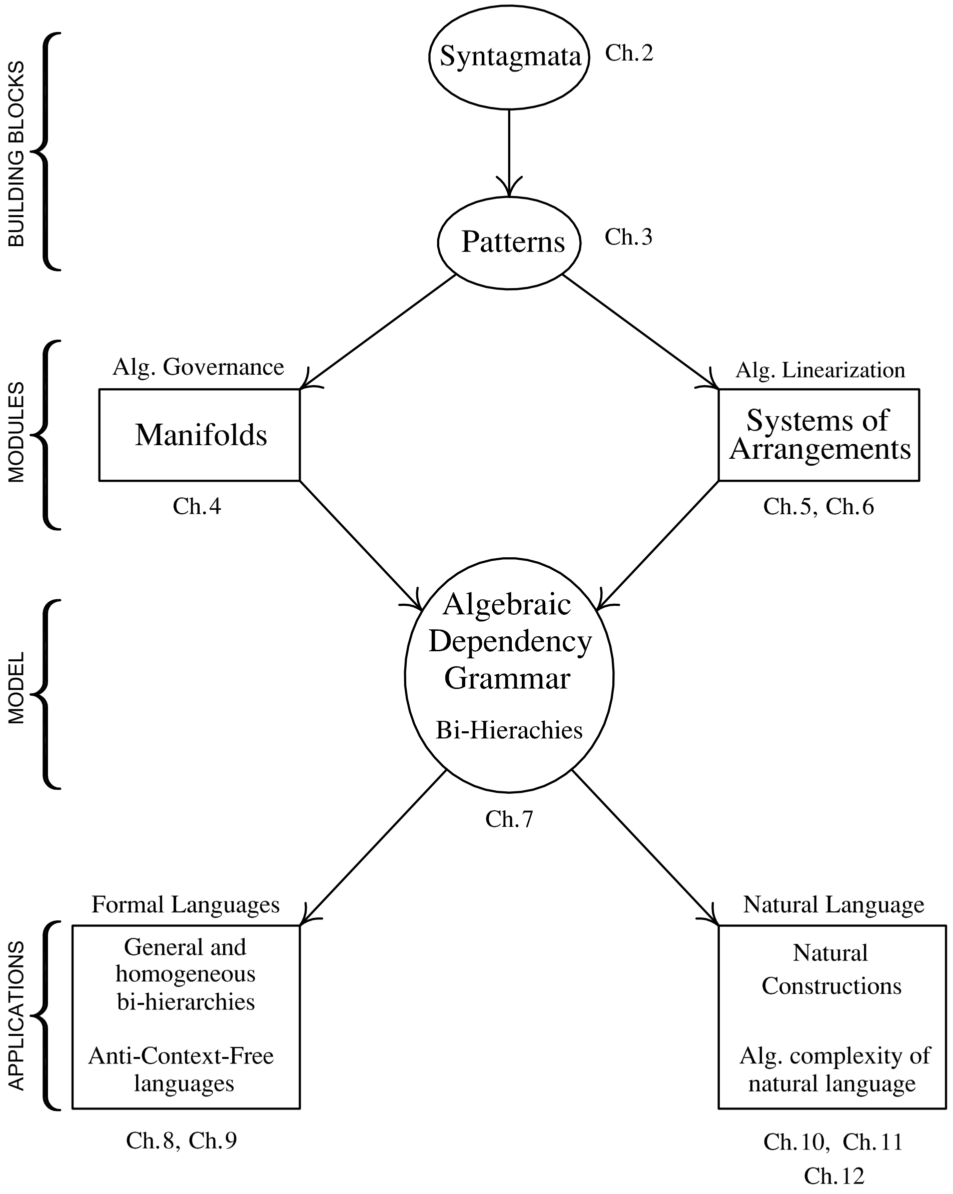


Figure 1.4: organization of the model and the thesis by concepts (alg.=algebraic).

key: a **subsyntagma** is a substructure of the syntagma induced by a pattern. Thus we obtain a generalized notion of projectivity.

However this does not tell us how we have to linearize a syntagma. We need a further element. We will notice that a linearization must behave recursively. If we decompose a syntagma in subsyntagmata and we specify how these subsyntagmata must be ordered (this is an **arrangement**) and we repeat this action for each subsyntagma we will have established a recursive procedure. This is formalized as a system of recurrence equations which we call **systems of arrangements**.

1.3.3 The Model: Algebraic Dependency Grammars and the Bi-Hierarchy

Joining the above modules we already have the formalism: an **algebraic dependency grammar** is a manifold together with a linearization given by a system of arrangements.

Notice that patterns sustain both manifolds and linearizations. Since these parts have the same mathematical essence it will be natural to study their interrelation. More specifically, a monoidal pattern is a product of constants and submonoids. So every pattern has at least one description in terms of these basic classes. If we use the symbol $\mathbf{k}$ for constants and the symbol $\mathbb{M}$ for submonoids, a pattern which is a product of constant by submonoid by submonoid by constant by constant by submonoid can be described comfortably as: $\mathbf{k}\mathbb{M}^2\mathbf{k}^3\mathbb{M}\mathbf{k}$. In this way we will be able to write for example:

$$^{\mathbb{M}\mathbf{k}^3}\!\diagup_{\mathbf{k}^2\mathbb{M}^2}$$

to represent the class of languages which have manifolds using patterns of the form $\mathbb{M}\mathbf{k}^3$ and which are linearized using patterns of the form $\mathbf{k}^2\mathbb{M}^2$. The higher the complexity of this pair of descriptions, the higher is the complexity of the related languages. Since we have in fact two modules in the theory we have two classificatory axes. We call this a **bi-hierarchy**.

1.3.4 Applications: Formal Languages and Natural Languages

Once we have defined all the concepts to construct our machinery we will be able to put it to work and check its capabilities. The above described bi-hierarchy is a broad hierarchy of languages (which we call the **general bi-hierarchy**). However we can use more refined systems of classification, and thus more refined bi-hierarchies. Notably there is the **homogeneous bi-hierarchy**. In this bi-hierarchy we have two basic **types** to describe patterns, namely: $\mathbf{k}$ for constants and $\mathbb{H}$ for those monoids the compoments of which are all equal. We are going to classify several classical formal languages in the homogeneous bi-hierarchy. By way of a short sample:

$$\mathsf{CF} = {}^{\mathbf{k}^2\mathbb{H}}\!\diagup_{\mathbb{H}\mathbf{k}}, \quad \text{copy language is in } {}^{\mathbb{H}\mathbf{k}^2}\!\diagup_{\mathbb{H}\mathbf{k}}.$$

The bi-hierarchy enjoys interesting properties. The main one, called *bi-symmetry*, states that if we take the reversed words $X^{\mathfrak{R}}, Y^{\mathfrak{R}}$ we obtain the same class:

$$^{X}\!/_{Y} = {}^{X^{\mathfrak{R}}}\!/_{Y^{\mathfrak{R}}}.$$

All these properties are going to suggests that there is a hidden class, which we can name *anti-context-free* and which we notate $-\mathsf{CF}$ given by the class $\mathbb{H}\mathbf{k}^2/\mathbb{H}\mathbf{k}$. Such languages contain constructions symmetric to context-free languages.

Regarding natural languages we will firstly analyze several constructions involving manifolds and linearizations. Although we are not going to construct a complete grammar for a natural language we will claim the viability of this showing some complex linguistic phenomena. We study them attending structural facts and we see that linguistic phenomena can be classified according to certain fragments of an adequate classificatory monoid. Secondly we reanalyze the constructions attending to just the weak capacity, and we show that the semi-linear constructions can be reduced to the class $\mathbf{k}^2\mathbb{H}/\mathbb{G}\mathbf{k}\mathbb{G}$. We show that this class contains context-free languages, anti-context-free languages (and hence all the copy languages), that they are semi-linear and decidable. We also see that the manifolds defining languages in this class are locally definable sets of syntagmata, and hence they are decidable in linear time.

1.3.5 The Model in the Context of Dependency Based Formalisms

The algebraic dependency grammar proposal matches some of the characteristics of known models and differs from other ones. Let us review the parameters in relation to our proposal.

> **Stratification.** Since we have decided to separate the dependency structure from the word-order, algebraic dependency grammar is bi-modular: *algebraic governance* + *algebraic linearization*.
>
> **Nodes.** In this aspect we follow the majority: one-node-one-word, although syntagmata and manifolds allow other possibilities and we do not exclude other analyses.
>
> **Grammar Roles.** We will use ordinary syntactic functions like: subject, object, indirect object. See section §1.5 and Appendix B.
>
> **Projectivity.** Our model incorporates projectivity amongst other kinds of linearizations. We will see that projective linearizations are given by patterns of the form $\mathbb{M}\mathbf{k}$, which are only one of many possibilities: $\mathbf{k}, \mathbb{M}, \mathbf{k}\mathbb{M}, \mathbb{M}\mathbf{k}^2, \ldots$.

1.3.6 Structure of the Thesis

The thesis has a tripartite structure: I *Introduction*, II *Theory*, and III *Applications*, the contents of each part of which have been already commented. The previous Fig. 1.4 incorporated the chapters dealing with each item. We have deferred to three appendices contents on which the main reading does not depend. Appendix A examines the question of coordination, which is a controversial point in dependency grammar. Since we are going to implement some fragments of natural language, it is necessary to configure a system of syntactic functions, which is done more systematically in the Appendix B. We have reserved the Appendix C for some algorithmic questions concerning manifolds.

1.4 Mains Contributions of this Thesis

Let us highlight the main contributions of the thesis. We separate two domains: mathematical linguistics and formal language theory.

Mathematical Linguistics

- Algebraic dependency grammar **considers trees and word-order different modules in the architecture**. By treating linearizations separately from trees we can describe languages with highly free word-order. Systems of arrangements permit us to do this compactly.

  We allow **the possibility of ellipses**. This issue is usually avoided because it makes some formalisms non-decidable. However under certain conditions ellipses can be appropriate and can be tractable.

- By introducing an adequate bi-hierarchy, we **differentiate structurally linguistic phenomena by their algebraic description**. Local constraints, long distance agreement, non-semi-linear constructions, projectivity, movements, extrapositions or nested and cross-serial dependencies each have a characteristic algebraic description.

  Algebraic dependency grammar permits **observance of affinity between linguistic constructions** which seem superficially different. For example we see that the family of West German languages share a same scheme of linearization.

Formal Language Theory

- **A new system for understanding a very large family of languages** is presented. The classical Chomsky hierarchy and the successive hierarchies appended along the decades are linear in their conception. Bi-hierarchies emerge as more intricate organizations which permit observation of languages in broader contexts.

- We discover **a new class named anti-context-free languages**, $-\mathsf{CF}$, which shares some properties with context-free languages but which occupies a symmetric place in the homogeneous bi-hierarchy with respect to its homologue. We prove that $-\mathsf{CF}$ contains only semi-linear languages and that all the copy and respectively languages are anti-context-free. Informally we could say that CF are languages which are well-parenthesized, while $-\mathsf{CF}$ are languages exhibiting analogous cross-serial dependencies.

There are two aspects which will not treated in the thesis. For reasons of space we will not address semantics, although some suggestions are made tangentially. Parsing is a fundamental part of any formal linguistic model. Unfortunately we cannot enter into the parsing issue itself. We will show, however, that the universal problem for algebraic dependency languages is decidable.

1.5 Styles and Notation of Dependency Analyses

Styles of Dependency Analysis

Dependency analyses are very intuitive in the majority of cases. Frequently the structure is suggested by the semantic reading of the sentence.[17] However there are some tricky cases for which linguists do not agree on a unique solution. The main such cases are:

- What must be the structure for coordination? Or even, are bare dependencies capable of capturing coordination?
- Assuming that dependency structures suffice for coordination, where must coordinators be placed in them?
- Where must prepositions be placed?
- Where must auxiliary verbs be placed?

This thesis does not propose to solve these questions incontestably. We are more interested in developing the underlying mathematics of dependency than in finding evidences for one or other style of analysis. These issues are more purely linguistic than mathematical. Nevertheless in order to display examples we need to adhere to certain conventions. For this reason let us comment the last points.

- Coordination is perhaps the more delicate and controversial of these points and we dedicate some space to discuss the pro- and contra-arguments in Chapter 2 and in Appendix A. In brief, we accept than dependency trees suffice and we assume a style *à la* Mel'čuk for coordination, i.e. the conjuncts are chained through a coordination function, see (Melčuk, 1988).

[17] Here by "analysis" we mean the general style of analysis, not the specific parsing of a sentence.

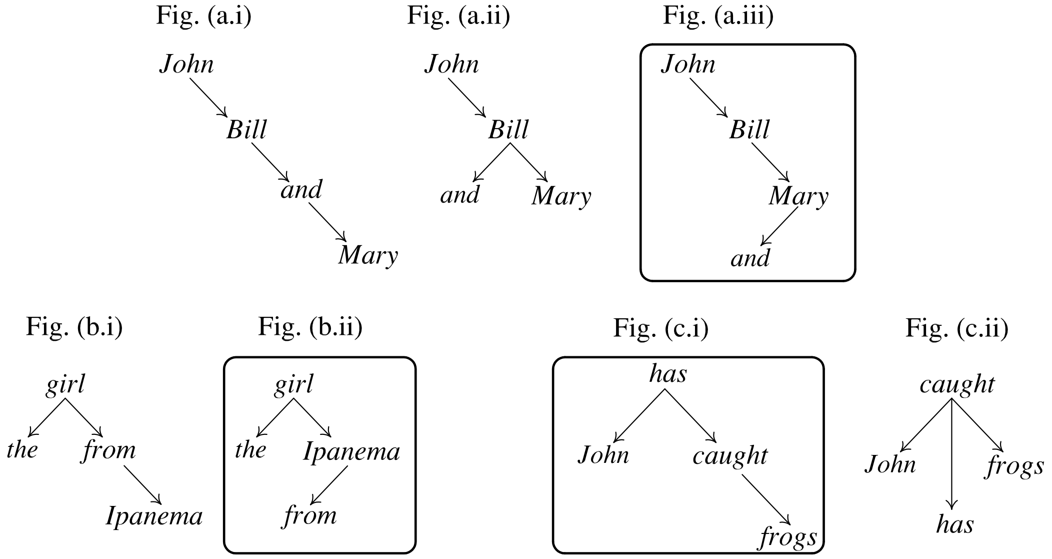


Figure 1.5: (a) some possible positions of the coordinator; (b) possible positions of the preposition; (c) possible positions of the auxiliary verbs. The enclosed dependency structures correspond to the style chosen in the thesis.

- However once we have assumed the last point, we can still place the coordinators in several places. See Fig.1.5(a.i), (a.ii) and (a.iii). We will take the bottom position, Fig. 1.5(a.iii).[18] See details in Appendix A.

- Basically there are two possible solutions: either the preposition dominates the rest of the phrase, or the preposition is governed by the noun; see Fig. 1.5 (b.i) and (b.ii). The second is the style adapted by The Stanford NLP Group (2017), although Groß and Osborne (2015) and Osborne (2015) give good evidence in defense of the other option. We will take the second, (b.ii), however we feel ourself very tempted to follow the arguments of Groß and Osborne.

- Usually the auxiliary verbs are placed at the top as in Fig. 1.5(c.i) and so they govern the full verb. The other solution consists in understanding auxiliaries as complements of the full verb as in Fig. 1.5(c.ii). This issue is related to the last point, in fact Groß and Osborne advocate solutions such as (c.i), while the Standford group, (c.ii). Here we follow the most widespread solution (c.i).

Of course the list of tricky cases does not end here. For example where must a floating quantifier be placed? Or, must interrogative particles be moved in the analyses? We address such issues in Chapters 10, 11 and 12.

[18] Although Mel’čuk uses the variant (a.i).

Remark 1.1. None of the conventions assumed above need to be taken definitively. Our notion of grammar will be capable of accommodating the several possible alternatives.

Syntactic Functions Notation

Regarding notation for syntactic functions, there exists a growing necessity to unify a nomenclature for syntactic functions in the treebanks, for example (The Stanford NLP Group, 2017) or (Charles University, Czech Republic Faculty of Mathematics and Physics, 2017). Unfortunately these notational systems are inconveniently long for an algebraic manipulation. In addition we will stipulate a structure on the functions which is not shared by these treebanks. In consequence we develop our own particular notational system. The Appendix B contains tables and lists of examples for consultation at any moment of the reading. Nevertheless for most of the examples it is not necessary to know all the system. The following shorter list covers the majority of cases.

Sb — *Subject function* introduces the subject argument of a verb when it is dominated by a noun.

Ob — *Object function* introduces the object argument of a verb when it is dominated by a noun.

In — *Indirect function* introduces the indirect object argument of a verb when it is dominated by a noun.

Pd — *Predicate function* or *attribute function* introduces the attribute of a copulative verb like *to be*.

Ax — *Auxiliary function* introduces a verb which is governed by an auxiliary verb.

Ob_S — *Sentential or Propositional subordinate object function* introduces a subsyntagma representing a subordinate clause which plays the role of a propositional object. The functions Sb_S, In_S are similar.

Dt — *Determiner function* introduces the determiner which is governed by a noun.

Ad — *Adjective function* introduces an adjective which is governed by a noun.

Nc — *Noun complement function* introduces a noun complement mediated by a preposition.

Co — *Coordination function* chains conjuncts.

1.6 Mathematical Notation and Background

We have decided to maintain throughout the text a high level of formality. So we have tried to prove detailedly all the mathematically significant statements. This entails that some intuitive results have a non-short proof, which can discourage the reader. In order to facilitate a general reading we have advertised those parts of the text which are not fully necessary to understand the thesis as a whole. The symbol ♣ indicates that the item (section, theorem, proof, . . .) can be skipped.

Algebraic style tends to introduce objects according to some preliminary universal property which licenses the concept and then a posterior proposition tells how to obtain a set-theoretical representation of the object. This contrasts with the computer science style which introduces directly the set-theoretical representation.

We introduce basic mathematical notions and fix notations. Some of these items will be defined again in the course of the thesis. We have tried to preserve the most widespread mathematical notation. Subsections titled as "specific notation" are necessary for understanding concepts. For deeper exposition of the concepts we refer the reader to the cited bibliography.

1.6.1 Set Theory

Conventional notation

We use the symbols and operators $\in, \varnothing, \cup, \cap, \subset, \subseteq$ in the usual sense. Sometimes we will emphasize $A \subset B$ as $A \subsetneq B$. $A \sqcup B$ means $A \cup B$ but in addition this notation indicates that the union is disjoint $A \cap B = \emptyset$. $A \setminus B$ stands for the difference of sets. $A - x = A \setminus \{x\}$. The *power set of* A is the set of all subsets of A and it is notated $\wp(A)$. The *cardinality* of a set A is notated $|A|$.

Given a set A, we notate $A \times B$ the *Cartesian product* consisting of all the ordered pairs of elements of A and of B which we notate (a, b). $A^n = A \times \cdots \times A$ where the product is repeated n times. We call *vectors* of length n elements $(a_1, \ldots, a_n) \in A^n$. We call *components* every a_i for $i = 1, \ldots, n$. We omit parenthesis in contact with other superscript operators such as $A^{*n} = (A^*)^n$.

A binary relation π between the sets A and B is a subset $\pi \subseteq A \times B$. A relation f is a *mapping* provided that for every $x \in A$ there is at most one $y \in B$, called the *image*, such that $(x, y) \in f$.

Mappings are notated as $f : A \longrightarrow B$. The image of an element $x \in A$ is notated $f(x)$. The *domain* of f is the set $\mathrm{dom}(f) = \{x \in A \mid \exists x \in B \text{ such that} f(x) = y\}$. The *image* is the set $\mathrm{im}(f) = f(A) = \{y \in B \mid \exists x \in A \text{ such that } f(x) = y\}$. A mapping is said to be *total* when the domain is A, otherwise it is said to be (properly) *partial*. If we do not say anything we presuppose that the mapping is total. The *identity mapping* on set A is the total mapping $id_A : A \longrightarrow A$ given by $id_A(x) = x$.

The *canonical set extension* of an operator is the extension element-wise. For example $f(A) = \{f(a) \,|\, a \in A\}$ or $A \cdot B = \{a \cdot b \,|\, a \in A, b \in B\}$.

A mapping $f : A \longrightarrow B$ is *injective* when for any $x, y \in A$ $f(x) = f(y) \implies x = y$. f is *surjective* or *epijective* when $f(A) = B$. f is *bijective* when it is injective and surjective.

Specific notation

When all the n components of a vector are equal we write $(a)_n = (a, \ldots, a)$. We call *homogeneous power* the set $(A)_n = \{(a)_n \,|\, a \in A\}$. Do not confuse $(A)_2 = \{(a, a) \,|\, a \in A\}$ with $A^2 = \{(a_1, a_2) \,|\, a_1, a_2 \in A\}$. In this notation parentheses will always be maintained in order not to confuse the homogeneous power with an ordinary subscript. However we avoid the second pair of parentheses in expressions such as $((A)_2)^*$ for which we will simply write $(A)_2^*$.

$\mathbb{N}$ is the set of natural numbers, which must not to be confused with the blackboard notation for the types in the classificatory monoid $\mathbb{M}$. We assume by convention that 0 is not a natural number. In general the subindex + in X_+ adds to the set X an extra element which depends on the context. Thus, the following expressions will be very frequent: $\mathbb{N}_+ = \mathbb{N} \cup \{0\}$, $\Sigma_+ = \Sigma \cup \{0\}$ and $\zeta_+ = \zeta \cup \{1\}$. $\{a\}_+ = \{a, 0\}$ (or more informally $a_+ = \{a, 0\}$). We must not confuse this notation with $\Sigma^+ = \Sigma^* \backslash \{0\}$, see later §1.6.5. For *singletons*, i.e. sets with only one element, we frequently write a instead $\{a\}$.

1.6.2 Algebra[19]

Algebras

An *operation of arity t* is a mapping $+ : A^t \longrightarrow A$. An *algebra* is a set A together with some operations notated $(A; +_1, \ldots, +_n)$. As is usual we will simply notate A the algebra when the operations are given by the context.

Let $(A; +_1, \ldots, +_n)$ and $(B; \times_1, \ldots, \times_n)$ be two algebras such that the arity of each $+_i$ and $\times_i$ coincides and is t_i. A *morphism of algebras* is a mapping $f : A \longrightarrow B$ such that for each operation $+_i$ and $\times_i$, $i = 1, \ldots, n$ we have that:

$$f\big(+_i\,(a_1, \ldots, a_{t_i})\big) = \times_i\big(f(a_1), \ldots, f(a_{t_i})\big),$$

where t_i is the arity of each operation. A morphism of algebras is said to be a *monomorphism* when it is injective, *epimorphism* when it is surjective, *isomorphism* when it is bijective. When two algebras are isomorphic we will notate $A \cong B$.

[19] References: (Grätzer, 1968), (Sankappanavar and Burris, 1981), (Lang, 2004), (Sakarovitch, 2009), (Basart, 2003).

Given an algebra $(A; +_1, \dots, +_n)$, a *subalgebra* is a subset $B \subseteq A$ such that all the operations restricted to B are closed. Every subalgebra is trivially an algebra $(B; +_1, \dots, +_n)$.

Imposing conditions on the operations (associativity, commutativity, . . .) we obtain several kinds of algebras. Some examples which we are going to introduce later are: semigroups, monoids, semilattices, lattices, boolean algebras, ...

Relational Algebras

A *relational algebra* is a generalization of algebras in the sense that mappings are simply relations. So a relational algebra is a set A with some relations $(A; \pi_1, \dots, \pi_n)$, $\pi_i \subseteq A^{t_i}$ for any $i = 1m, \dots, n$. Given two relational algebras $(A; \pi_1, \dots, \pi_n)$ and $(B; \sigma_1, \dots, \sigma_n)$, a *morphism of relational algebras* is a mapping $f : A \longrightarrow B$ such that for each relation π_i, $i = 1, \dots, n$ we have that:

$$(a_1, \dots, a_{t_i}) \in \pi_i \implies \big(f(a_1), \dots, f(a_{t_i})\big) \in \sigma_i,$$

where t_i is the arity of each relation π_i and σ_i. A morphism of algebras is said to be a *monomorphism* when f is injective, and *epimorphism* when it is surjective. However we have to note that the notion of isomorphism of relational algebras is stronger than simply algebras. A mapping $f : A \longrightarrow B$ is a *isomorphism of relational algebras* when it is bijective and:

$$(a_1, \dots, a_{t_i}) \in \pi_i \iff \big(f(a_1), \dots, f(a_{t_i})\big) \in \sigma_i.$$

When two relational algebras are isomorphic we will notate $A \cong B$.

One of the most immediate relational algebras are graphs. A *directed graph* is a pair (V, E) , $E \subseteq V \times V$, where V are called *vertices* and E *edges*. We will suppose that notions such as *path, cycle, forest, tree, . . .* are well known. There is abundant literature on the issue.

More in general imposing conditions on the relations (reflexivity, transitivity, . . .) we obtain several kinds of relational algebras. Some examples which we are going to introduce later are: graphs, preorders, partial orders, automata, ...

1.6.3 Order Theory[20]

Orders

A *preorder* is a set X together a binary relation $\leq \subseteq X \times X$, notated $(X; \leq)$, i.e. a relational algebra, satisfying that:

(i) (*reflexivity*) $\forall x \in X, \ \ x \leq x$;

[20] References: (Davey and Priestley, 2002), (Birkhoff, 1940).

(ii) (*transitivity*) $\forall x, y, z \in X, \quad x \leq y, \; y \leq z \implies x \leq z.$

When in addition it satisfies that:

(iii) (*antisymmetry*) $\forall x, y \in X, \; x \leq y, y \leq x \implies x = y.$

we say that the preorder is a *partial order* or that X is a *poset*. When a partial order satisfies that:

(iv) (*totality*) $\forall x, y \in X, \; x \leq y$ or $y \leq x$,

it is called a *total order*.

When $x \leq y$ and $x \neq y$ we write $x < y$. Two elements $x, y \in X$ are called *comparable* iff $x \leq y$ or $y \leq x$.

The *transitive closure* of a relation $\pi \subseteq X \times X$ is the least relation containing π which is transitive. The *reflexive closure* is defined similarly.

A way to show an order $(X; \leq)$ is through a *Hasse diagram* which consists in a graph where vertexes are the set X and edges are pairs (x, y) such that $x < y$ but there is no any other $z \in X$ such that $x < z < y$. So a Hasse diagram only contains the essential relations; the remaining relations can be deduced by transitive closure.

An *order morphism*, also called *order homomorphism*, *isotonic mapping* or *monotonic mapping* is a mapping $f : (X; \leq) \longrightarrow (Y; \leq')$, such that if $x \leq y \implies f(x) \leq' f(y)$, for any $x, y \in X$.

Semi-lattices and Lattices

Let $(X; \leq)$ be a partial order and let $Y \subseteq X$. An element $x \in X$ is an *upper bound* of Y iff $y \leq x$ for any $y \in Y$. A *Lower bound* is defined dually. We define the *suppremum* $\sup(Y)$ as the least upper bound of Y, provided that it exists. Dually we define the *infimum*, $\inf(Y)$.

We say that $(X; \leq)$ is an *upper semilattice* when for each $x, y \in X$, $\sup(\{x, y\})$ exists. This defines an operation in X denoted $\sup(x, y)$. Dually when the operation $\inf(x, y)$ exists, we say that $(X; \inf)$ is a *lower semilattice*. When both operations occur we have a *lattice*, $(X; \sup, \inf)$.

When $\sup(X)$ exists, we say that the semilattice is *upper bounded*. The bound is called *top*. Dually when $\inf(X)$ exists, we say that the semilattice is *lower bounded*. The bound is called *bottom*. Another common notation for $\sup(x, y)$ and $\inf(x, y)$ are *join* $x \vee y$ and *meet* $x \wedge y$. The operations in a lattice $(X; \vee, \wedge)$ satisfy for all $x, y, z \in X$:

(i) $x \leq y \iff x \wedge y = x \iff x \vee y = y$;

(ii) (*idempotency*) $x \vee x = x$ and $x \wedge x = x$;

(iii) (*associativity*) $(x \vee y) \vee z = x \vee (y \vee z)$ and $(x \wedge y) \wedge z = x \wedge (y \wedge z)$;

(iv) (*commutativity*) $x \vee y = y \vee x$ and $x \wedge y = y \wedge x$;

(v) (*absorption*) $x \vee (x \wedge y) = x$ and $x \wedge (x \vee y) = x$.

When in addition we have an upper and a lower bound, usually notated $\top$ and $\bot$, the lattice $(X; \vee, \wedge, \top, \bot)$ is called *complete* and it satisfies that for all $x \in X$:

(vi) (*bounds absorption*) $x \vee \top = \top$, $x \wedge \top = x$, $x \vee \bot = x$, and $x \wedge \bot = \bot$.

We say that the lattice is *distributive* when for all $x, y, z \in X$:

(vii) (*distributivity*) $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$ and $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$.

Finally if we can add a unary operator called *complement* $\cdot^{\complement}$ satisfying for any $x \in X$:

(viii) (*complement*) $x \vee x^{\complement} = 1$ and $x \wedge x^{\complement} = 0$;

then we call $(X; \vee, \wedge, \top, \bot, \cdot^{\complement})$ a *boolean algebra*.

A $\cup\cap$-combination of sets is a finite combination of intersections and unions of the sets. The $\cup\cap$-closure of a set $\mathscr{X}$ (of sets) is the least set closed by $\cup\cap$-combinations containing $\mathscr{X}$, or equivalently, the lattice generated by $\mathscr{X}$.

1.6.4 Monoids and Free Monoids[21]

Monoids

A *semigroup* is a set M equipped with a operation, $(M; +)$, such that it is associative: $\forall x, y, x \in M$, $(x + y) + z = x + (y + z)$. A *monoid* is a semigroup with a distinguished element called *identity* or *neutral element* which is notated $(M; +, 0)$ in additive notation or $(M; \cdot, 1)$ in multiplicative notation, and satisfying that $\forall x \in M$, $x + 0 = x = 0 + x$ in the additive notation (or $x \cdot 1 = x = 1 \cdot x$ in multiplicative notation). A monoid is called *abelian* iff the operation is commutative.

A *submonoid* is a subset $N \subseteq M$ such that the operation of the monoid restricted to N is closed and it contains the identity. Given a subset of a monoid $G \subseteq M$ we notate $\langle G \rangle$ the least submonoid of M containing G. A subset of a monoid $G \subseteq M$ is a *generator* of M iff $\langle G \rangle = M$. We say that a monoid M is *finitely generated* iff it has a finite generator set. All the monoids considered here will be finitely generated.

A *morphism of monoids* M and N (or *homomorphism*) is a mapping $f : M \longrightarrow N$ such that for any $x, y \in M$, $f(xy) = f(x)f(y)$. A morphism of monoids is determined by the image of a generator set of M.

[21] References: (Sankappanavar and Burris, 1981), (Lang, 2004), (Pin, 2016), (Sakarovitch, 2009).

Free Monoids

A monoid M is called *free* on X iff given a mapping $\nu : X \longrightarrow M$ for each monoid N and for each mapping $f : X \longrightarrow N$ there is a unique homomorphism $g : M \longrightarrow N$ such that $g \circ \nu = f$, i.e. we have the commutative diagram:

$$\begin{tikzcd}
X \arrow[r, "f"] \arrow[dr, "\nu"'] & N \\
& M \arrow[u, "g"']
\end{tikzcd}$$

The mapping ν is necessarily injective and we have that $M = \langle \nu(X) \rangle$. This is the bare algebraic characterization. However in computer science sometimes a more useful notion is string. An *alphabet* A is a finite set the elements of which are called *letters*. A *string* of length t is a tuple $(a_1, \ldots, a_t) \in A^t$. The product of two strings is defined as:

$$(a_1, \ldots, a_t) \cdot (b_1, \ldots, b_s) = (a_1, \ldots, a_t, b_1, \ldots, a_s).$$

An identity ε, called the *empty string* is introduced by convention:

$$\varepsilon \cdot (a_1, \ldots, a_t) = (a_1, \ldots, a_t) = (a_1, \ldots, a_t) \cdot \varepsilon, \quad \varepsilon \cdot \varepsilon = \varepsilon.$$

The set of strings over A is notated A^*. It can be proved that A^* with the product of strings is a (finitely generated) free monoid and that for any free monoid M there is an alphabet A such that $M \cong A^*$. The isomorphism is given by a bijection between generators of M and elements of A which are the generators of A^*.

Given a subset of strings $B \subseteq A^*$, $B^* = \langle B \rangle$ with the product of strings. The operator $\cdot^*$ is called *Kleene star*. The *plus* operator is defined as $A^+ = A^* \setminus \{\varepsilon\}$.

A free monoid A^* satisfies the *cancellation* laws:

$$x \cdot y = x \cdot z \implies y = z, \quad x \cdot y = z \cdot y \implies x = z.$$

for any $x, y, z \in A^*$. A main property of A^* is that for any $a_1, \ldots, a_n \in A$ and $b_1, \ldots b_m \in A$

$$a_1 \ldots a_n = b_1 \ldots b_m \iff n = m \text{ and } a_i = b_i \ \forall i = 1, \ldots, n.$$

In other words any element of a free monoid factorizes in a unique form on the generators.

Specific Notation for Monoids

Given the uniqueness of the factorization of free monoids, we also call the generators *prime factors*. The length of $x \in A^*$ is notated $|x|$. The number of occurrences of a letter $a \in \Sigma$ is notated $|x|_a$ or sometimes $\#_a(x)$.

We are going to use throughout two free monoids ζ^* and Σ^*. Elements in ζ are named *syntactic functions* and we will use preferably Greek letters $\alpha, \beta, \gamma, \ldots$ for abstract cases and the abbreviations *Sb*, *Ob*, *In*, ... for grammatical roles (syntactic functions); in both cases we will use the multiplicative notation.

Regarding Σ, when we deal with classical formal languages, its elements will be called *letters*, we will use the multiplicative notation and we will use preferably roman letters $\Sigma = \{a, b, c, \ldots\}$. However when we deal with natural languages, elements in Σ will be called *words* or *lexical items*, $\Sigma = \{$*John, hunts, frogs, ...*$\}$ and we will use additive notation.[22]

1 is the identity for the multiplicative notation and 0, for additive notation, so in general we are not going to use other common symbols like ε, λ or Λ for the empty word. With this, expressions looks as:

$$\begin{aligned} \alpha^2 \cdot \beta \cdot \gamma &\in \zeta^*, \\ Sb \cdot Ob^2 &\in \zeta^*, \\ a^2 b^2 c^2 &\in \{a^n b^n c^n \mid n \in \mathbb{N}_+\} \subseteq \Sigma^*, \\ John + hunts + frogs &\in \Sigma^*. \end{aligned}$$

The direct sum of two monoids M and N is the set $M \times N$ with the new operation taken component-wise: $(x, y) \cdot (x', y') = (xx', yy')$. We write them $M \oplus N$ in order to notice that the operation is taken component-wise. We also use this notation for singletons: if $x, y \in M$, $x \oplus y = (x, y)$, being both notations possible. For repeated sums we notate $M^n = \bigoplus_{i=1}^n M$. We avoid parenthesis in $(\zeta^*)^n = \zeta^{*n}$.

We have defined that $\langle G \rangle$ is the submonoid generated by G. Notwithstanding this notation, when the monoid is ζ^{*n} and $\xi \subseteq \zeta^{n*}$ we will use the notation ξ^* instead of $\langle \xi \rangle$. This operator is then a generalization of the Kleene star to monoids of several components. We reserve this notation exclusively for these submonoids in ζ^{*n}. For instance, considering the alphabet $\zeta = \{\alpha, \beta\}$ we have the submonoids of ζ^{*2}:

$$\begin{aligned} \{(1, \alpha), (1, \beta), (\alpha, 1), (\beta, 1)\}^* &= \langle (1, \alpha), (1, \beta), (\alpha, 1), (\beta, 1) \rangle = (\zeta^*)^2 = \zeta^{*2}, \\ (\alpha, \beta)^* &= \langle (\alpha, \beta) \rangle = \{(\alpha^n, \beta^n) \mid n \in \mathbb{N}_+\}, \\ \{(\alpha, \beta), (1, \alpha)\}^* &= \langle (\alpha, \beta), (1, \alpha) \rangle = \{(\alpha^n, x) \mid n \in \mathbb{N}_+, x \in \Sigma^*, |x|_\beta = n\}. \end{aligned}$$

Notice that with this notation $\zeta^{*n} = (\zeta^*)^n = (\zeta^n)^*$.

[22] The term "word" is also used for the strings in formal language theory. In the context of formal languages we will use the term "letters" for the elements in Σ and "strings" for the elements in Σ^*. In the context of natural languages we will use the term "words" for the elements in Σ and term "sentences" for the elements in Σ^*. Finally the elements in ζ will be called "syntactic functions" and the elements in ζ^*, "loci".

1.6.5 Formal Languages, Automata Theory and Theory of Computation[23]

Formal Languages

A *language* L is a subset of Σ^*. See page 301 for a list of languages appearing in the thesis. Given an $x \in L$, x^R stands for the reverse string. Operations on languages are the usual $L \cap L', L \cup L', L \cdot L', L^R, L^*, L^+, \ldots$.

A language $L \subseteq \Sigma^*$ and a language of $L' \subseteq \Sigma'^*$ are said to be *similar* (sometimes it is said that L is a *copy* of L') iff there is an isomorphism of free monoids $f : \Sigma^* \longrightarrow \Sigma'^*$ such that $f(L) = L'$. That is L and L' are the same language up to renaming of letters.

A *class* of languages $\mathcal{L}$ is a set of languages which is closed under similarity. That is, if $L \in \mathcal{L}$ and f is an isomorphism then $f(L) \in \mathcal{L}$. Sometimes a class is also called a *family of languages*.

Classes of languages are notated with sans serif typography, $\mathsf{RG}, \mathsf{CF}, \mathsf{CS}, \ldots$, while classes of grammars are notated with a roman one, RGG, CFG, CSG, ... See page 303 for a list of classes and acronyms appearing in the thesis.

A *context-free grammar* is a tuple $G = (N, \Sigma, P, S)$ such that: Σ and N are disjoint alphabets; Σ are the *terminal symbols* or *letters* or *vocabulary*; N are the *non-terminal symbols* or *variables*. $P \subset N \times (N \sqcup \Sigma)^*$ is a finite set of *productions* or *rewriting rules*; and $S \in N$ is the *start symbol*. Productions are usually notated by $X \to Y$, but sometimes in order to reserve the arrow for other meanings, we use the *Backus–Naur form*, $X ::= Y$. Equally instead of the start symbol S we will use the symbol *Start*. $x \Rightarrow y$ means that the string y is derivable in one step from y; $\overset{*}{\Rightarrow}$ denote the reflexive and transitive closure. Notation for other grammars will be presented in the course of the text following similar notations.

Automata Theory and Theory of Computation

For general concepts we have followed (Barton et al., 1987), (Hopcroft and Ullman, 1979) and (Hopcroft et al., 2001), and for a more algebraic style (Berstel, 1979), (Pin, 2016) and (Sakarovitch, 2009). We will suppose well-known fundamental notions on computational theory, for which we take as reference (Arora and Barak, 2009).

1.6.6 Categories and Functors[24]

A *category* $\mathbf{C}$ consists in:

(i) a set of *objects* notated $\mathrm{Ob}(\mathbf{C})$;

[23] References: (Barton et al., 1987), (Hopcroft et al., 2001), (Hopcroft and Ullman, 1979), (Berstel, 1979), (Pin, 2016), (Sakarovitch, 2009), (Kelley, 1995), (Alfonseca Cubero et al., 2007), (Arora and Barak, 2009).

[24] References: (Adámek et al., 2004), (Lang, 2004).

(ii) for any pair of objects $X, Y \in \mathrm{Ob}(\mathbf{C})$, a set $\mathrm{Hom}_{\mathbf{C}}(X, Y)$, called *homomorphisms* from X to Y;

(iii) for any three objects $X, Y, Z \in \mathrm{Ob}(\mathbf{C})$, a mapping

$$\mathrm{Hom}_{\mathbf{C}}(X, Y) \times \mathrm{Hom}_{\mathbf{C}}(Y, Z) \longrightarrow \mathrm{Hom}_{\mathbf{C}}(X, Z),$$

which is named *composition*. The image of this mapping of the pair (f, g) is notated $g \circ f$, o simply gf.

In addition the following axioms must be satisfied:

(*associativity*) the composition is associative: if $f \in \mathrm{Hom}_{\mathbf{C}}(X, Y)$, $g \in \mathrm{Hom}_{\mathbf{C}}(Y, Z)$ and $h \in \mathrm{Hom}_{\mathbf{C}}(Z, W)$, then $h \circ (g \circ f) = (h \circ g) \circ f$;

(*identities*) for any object $X \in \mathrm{Ob}(\mathbf{C})$, there is an element id_X such that $id_X \circ f = f$ and $g \circ id_X = g$, for any $f \in \mathrm{Hom}_{\mathbf{C}}(Y, X)$ and for any $g \in \mathrm{Hom}_{\mathbf{C}}(X, Y)$.

Examples of categories are the *category of sets with usual mappings*, the *monoids and their morphisms*. Similarly other elementary structures such as groups or rings form a category. Homomorphisms are not necessarily mappings in the classical set theory sense.

Let $\mathbf{C}$ and $\mathbf{C}'$ be two categories. A *functor*, $F : \mathbf{C} \longrightarrow \mathbf{C}'$ is a mapping which for each object $X \in \mathrm{Ob}(\mathbf{C})$ assigns an object $F(X) \in \mathrm{Ob}(\mathbf{C}')$, and which for each morphism $f : X \longrightarrow Y \in \mathrm{Hom}_{\mathbf{C}}(X, Y)$ assigns a morphism $F(f) : F(X) \longrightarrow F(Y) \in \mathrm{Hom}_{\mathbf{C}'}\big(F(X), F(Y)\big)$. In addition it must be satisfied that:

(i) $F(id_X) = id_{F(X)}$;

(ii) $F(f \circ g) = F(f) \circ F(g)$, for any homomorphisms f, g.

A functor is called *faithfull* provided that all the hom-set restrictions

$$F : \mathrm{Hom}_{\mathbf{C}}(X, Y) \longrightarrow \mathrm{Hom}_{\mathbf{C}'}\big(F(X), F(Y)\big)$$

are injective. When they are surjective the functor is called *full*. A functor is *isomorphic* when it is faithful, full and bijective on the objects.

2

Syntagmata and their Adequacy

The present and the following chapter are devoted to introduce the most basic notions, namely *syntagmata* and *monoidal patterns*. They can be thought of respectively as the bricks and the mortar with which we will construct the elements of the whole theory.

In this chapter we introduce syntagmata which is nothing more than a simple mapping from the free monoid generated by the set of syntactic functions to the vocabulary. So a syntagma assigns words to strings of syntactic functions, which amount to syntactic addresses.

The graphical representation of a syntagma is a tree with the edges labeled with syntactic functions with the peculiarity of being *functional*. This property means that given a node and a syntactic functions there is at most one node governed by the node through the function. We give linguistic arguments in order to motivate tree-shape and functionality.

2.1 Definitions

Definition 2.1. Given a (possibly empty) finite set ζ called *syntactic functions*, we call the elements of the free monoid ζ^* *loci*. Given a non-empty finite set Σ which we call *vocabulary*, we notate $\Sigma_+ = \Sigma \cup \{0\}$ where 0 is an extra element $0 \notin \Sigma$ called the *null word*.

We keep as far as possible the letters ζ for the syntactic functions and Σ for the vocabulary. We are going to use free monoids ζ^* and Σ^*. 1 is the identity of ζ^* and 0, the identity of Σ^*. We will use the multiplicative notation for ζ^*, while for Σ^* we will use the additive notation. Since Σ represents the vocabulary, the languages are taken from Σ^* with the additive notation, however we make an exception for classical formal languages, like $\{a^n b^n c^n \mid n \in \mathbb{N} \cup \{0\}\}$ for which we prefer preserve the classical multiplicative notation.

Definition 2.2. Given a mapping $S : \zeta^* \longrightarrow \Sigma_+$ we call its *support* the set

$$\mathrm{Spt}(S) = \{x \in \zeta^* \mid S(x) \neq 0\}.$$

A syntagma with syntactic functions ζ and vocabulary Σ is a mapping $S : \zeta^* \longrightarrow \Sigma_+$ such that its support is finite, $|\mathrm{Spt}(S)| < \infty$. We call the *size* or the *order* of a syntagma S the cardinality of the support, $|\mathrm{Spt}(S)|$ and we write $|S| = |\mathrm{Spt}(S)|$.

We notate $\mathbf{Synt}_{\Sigma,\zeta}$ the set of syntagmata $S : \zeta^* \longrightarrow \Sigma_+$. When these sets are given by the context we simply write **Synt**.

The locus $Dt \cdot Sb$, say, must be read as *the determiner of the subject* or the locus $Md \cdot Ad \cdot Ob$ must be read as the *the modifier of the adjective of the object*. In the framework of formal languages syntactic functions will be notated generally by Greek letters, the more usual being: $\alpha, \beta, \gamma, \lambda, \varphi, \psi$.

Example 1. A SYNTAGMA. Let the syntactic functions be $\zeta = \{Sb, Ob, Dt, Ad\}$ and let the vocabulary be $\Sigma = \{\text{cup, dirty, soldier, the, washed, young}\}$. Let S be the mapping

$$S(x) = \begin{cases} \text{washed} & \text{if } x = 1; \\ \text{soldier} & \text{if } x = Sb; \\ \text{cup} & \text{if } x = Ob; \\ \text{the} & \text{if } x = Dt{\cdot}Sb; \\ \text{young} & \text{if } x = Ad{\cdot}Sb; \\ \text{the} & \text{if } x = Dt{\cdot}Ob; \\ \text{dirty} & \text{if } x = Ad{\cdot}Ob; \\ 0 & \text{otherwise.} \end{cases}$$

S is a syntagma with order $|S| = 7$ and $\mathrm{Spt}(S) = \{1, Sb, Ob, Dt{\cdot}Sb, Ad{\cdot}Sb, Dt{\cdot}Ob, Ad{\cdot}Ob\}$.

Definition 2.3. We call *atomic syntagma*, syntagmata with a unique letter $a \neq 0$ at the root. We notate them by $a^\bullet$, that is:

$$a^\bullet(x) = \begin{cases} a & \text{if } x = 1; \\ 0 & \text{otherwise.} \end{cases}$$

We call *null syntagma* the syntagma everywhere null, $S(x) = 0, \forall x \in \zeta^*$. We also notate them by $0^\bullet$, although we do not consider null syntagma as atomic.

Definition 2.4. We introduce the following kit of basic concepts given a syntagma S:

(i) We say that a locus $\varphi \in \zeta^*$ is an *ellipsis* iff $S(\varphi) = 0$, but $S(x\varphi) \neq 0$ for some $x \in \zeta^*$ with $x \neq 1$. We notate $\mathrm{Ell}(S)$ the set of ellipses of S. We say that S is *non-elliptic* iff there are no ellipses in S, that is $\mathrm{Ell}(S) = \emptyset$. We notate **Nell** the set of non-elliptic syntagmata.

(ii) We say that a locus is *definitively null* when it is null but not an ellipsis.

(iii) The *envelope*, notated $\mathrm{Env}(S)$, is the set of loci which are not definitively null.

(iv) A *child of* φ is a locus $\alpha\varphi$ with $\alpha \in \zeta$. We also say that φ *governs* $\alpha\varphi$.

(v) A *leave* is a non-null locus such that all its children are definitively null. The set of leaves is notated $\mathrm{Lvs}(S)$.

(vi) The *depth* of a syntagma $\mathrm{depth}(S)$ is the maximum length of its leaves.

(vii) Given two locus $\varphi, \psi \in \zeta^*$, we say that φ *dominates* ψ, and we write $\varphi \geq \psi$ (or $\psi \leq \varphi$) when φ is a suffix of ψ, i.e. exists a $\phi \in \zeta^*$ such that $\psi = \phi\varphi$.

The term "ellipsis" is reminiscent of linguistic ellipsis (or gapping). In order to indicate the presence of an ellipsis in a sentence we will use the usual symbol $\varnothing$, however in the syntagma we will use the null word.

Definition 2.5. Given a syntagma S and a subset $\Gamma \subseteq \zeta^*$ we define the *syntagma restricted to* Γ as

$$S\Gamma(x) = \begin{cases} S(x) & \text{if } x \in \Gamma; \\ 0 & \text{otherwise.} \end{cases}$$

Notice that $\mathrm{Spt}(S\Gamma) = \{x \in \zeta^* \mid S\Gamma(x) \neq 0\} = \{x \in \zeta^* \mid x \in \Gamma \text{ and } S(x) \neq 0\} = \mathrm{Spt}(S) \cap \Gamma$.

2.2 Graphical Representation of a Syntagma

To represent graphically a syntagma we can use as base the Cayley digraph of a binary operation. The Cayley digraph of a free monoid ζ^* is the directed graph (V, E) with vertexes $V = \zeta^*$ and edges $E = \{(x, \lambda x) \mid x \in \zeta^*, \lambda \in \zeta\}$. Edges $(x, \lambda x)$ are labeled with the function λ. We call this the *right-handed representation*. Since the monoid ζ^* is free, the Cayley digraph is always tree-shaped (this can be proved using the cancellation laws of free monoids).

Now we label the vertices according to the map S with the convention of not picturing definitively null loci, that is, only the envelope $\mathrm{Env}(S)$. This will be the standard representation of a syntagma in the sequel. Fig. 2.1(a) shows a syntagma with all the loci depicted; Fig. 2.1(b) shows the standard representation, i.e. without definitively null loci; Fig. 2.1(c) shows the graphical representation of the syntagma from the last example.

The set of loci ζ^* can be interpreted as a set of addresses. The reading of a locus $Dt \cdot Sb$ (*the determiner of the subject*) gives a path from the locus to the root of the tree.

The notions just introduced depend on the handedness of the products. One can define similarly *co-elliptic* locus φ satisfying $S(\varphi) = 0$, but $S(\varphi x) \neq 0$ for some $x \in \zeta^*$

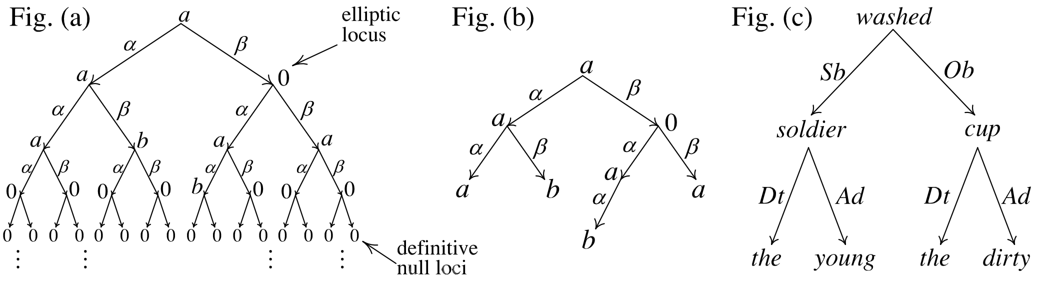


Figure 2.1: (a) a syntagma where all loci are represented, including null loci; (b) standard representation for a syntagma (definitively null loci are omitted, only the envelope); (c) Representation of the syntagma from the Example 1.

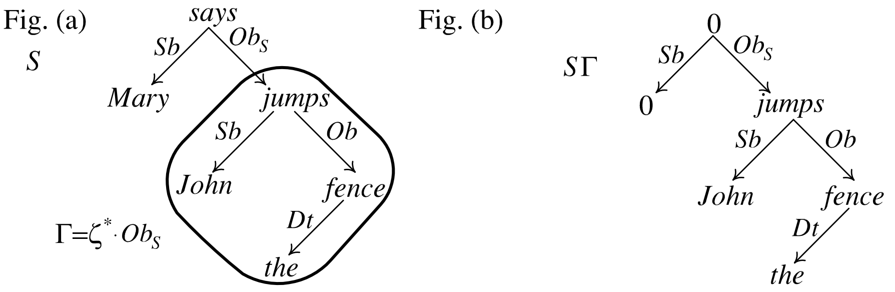


Figure 2.2: (a) an example of a syntagma S with the set of loci $\Gamma = \zeta^* \cdot Ob_S$ encircled; (b) the restricted syntagma $S\Gamma$.

with $x \neq 1$. φ is a *co-definitively null* locus when it is null but not a co-ellipsis. Similarly we can establish *co-leaf* and *co-depth*. As we are going to see these co-notions are not geometrically so immediate as the corresponding notions which we will use in the majority of situations. However in some situations the co-notions will be helpful.

Regarding the graphical representation, note that we could have chosen the other side to define edges: $\{(x, x\lambda) \,|\, x \in \zeta^*, \lambda \in \zeta\}$ (*left-handed representation*). For graphical representations we always use the definition before, but this handedness will play a prominent role in the Thesis.

2.3 Some Immediate Properties of Syntagmata

Lemma 2.6. *For every syntagma S with size s, depth d, and e ellipses over the set of functions ζ, such that $|\zeta| = m > 1$, we have the inequalities:*

$$d + 1 \leq s + e \leq \frac{m^{d+1} - 1}{m - 1}.$$

When $m = 0$ then we have that $e = d = 0$ and $s \leq 1$. When $m = 1$ then $s + e = d + 1$.

Proof. When $m = 0$, S is atomic and then $e = d = 0$ and $s \leq 1$. Suppose $m \geq 1$. First we consider the case $e = 0$. When $m = 1$ it is trivial that $s = d + 1$. When $m > 1$ the inequalities come from counting vertices of the thinnest tree and the widest tree of its graphical representation, both with a fixed depth d. That is, $d + 1$ and $m^0 + m^1 + \cdots + m^d = \frac{m^{d+1}-1}{m-1}$, respectively. For the elliptic case $e > 0$ we can imagine that we relabel the ellipses by a new extra letter. This new syntagma has size $s + e$. □

Lemma 2.7. *The dominance relation $\leq$ is a partial order.*

Proof. Trivial by the neutral, associativity and cancellation laws of free monoids. □

Lemma 2.8. *Given a syntagma S, we have that:*

(i) $\mathrm{Spt}(S) \subseteq \mathrm{Env}(S)$*;*

(ii) $\mathrm{Env}(S) = \{x \in \zeta^* \mid y \leq x,\ y \in \mathrm{Lvs}(S)\} = \{$*suffixes of* $y \mid y \in \mathrm{Lvs}(S)\}$*;*

(iii) $\mathrm{Env}(S) = \mathrm{Spt}(S) \sqcup \mathrm{Ell}(S)$*. Then, S is non-elliptic iff* $\mathrm{Env}(S) = \mathrm{Spt}(S)$.

Proof. (i) Trivial: if a locus is non-null then it is not definitively null. (ii) The second equality is trivial. Consider the first equality ($\subseteq$). If $x \in \mathrm{Env}(S)$ then x is not definitively null, whereby either x is a leaf or there is a leaf under it, say $y \leq x$. So $x \in \{x \in \zeta^* \mid y \leq x,\ y \in \mathrm{Lvs}(S)\}$. Consider the other direction ($\supseteq$). y is a leaf iff it is non-null and all its children are definitively null. Then any locus x such that $y \leq x$, cannot be definitively null (otherwise y would be null). (iii) We can decompose $\mathrm{Env}(S)$ as:

$$\begin{aligned}\mathrm{Env}(S) &= \{x \in \zeta^* \mid x \text{ is not definitively null and } S(x) \neq 0\}\\ &\quad \sqcup \{x \in \zeta^* \mid x \text{ is not definitively null and } S(x) = 0\}\\ &= \mathrm{Spt}(S) \sqcup \mathrm{Ell}(S).\end{aligned}$$

S is non-elliptic iff $\mathrm{Ell}(S) = \emptyset$, hence the equivalence. □

Remark 2.9. Notice that given the support $\mathrm{Spt}(S)$, the set $\mathrm{Env}(S)$ is obtained by suffix closure. For non-elliptic syntagmata, they correspond to the notion of *tree domain* (although sometimes tree domains are taken with an additional condition which orders lexicographically the set ζ^*). Thus syntagmata are a generalization of tree domains with some possible gaps.

For a computational implementation we need to take a finite version of a definition of a syntagma. Given a syntagma we only need to deal with the mapping restricted to the support. Hence a syntagma in computational terms is indeed a pair (A, S) where A is a finite subset $A \subseteq \zeta^*$ and S is a mapping $S : A \longrightarrow \Sigma$.

2.4 Why Are Syntagmata Suitable? Linguistic Considerations

Syntagmata are very simple algebraic structures which in the end represent trees with the edges labeled by functions and nodes labeled by a vocabulary; notice in addition that two edges originating from the same node cannot be labeled by the same function. In other words, given a locus and a syntactic function there is at most one node governed. This property is called *functionality*.[1]

These two properties, tree-shape and functionality, determinate totally the character of a syntagma, as we are going to prove later in the mathematical considerations. However, before this, let us make some linguistic comments.

Tree-shape

In a semantic representation what is sought is representing only the argumental relations involved in a sentence without worrying about morphology or word-order and other syntactic constraints. Those nodes which represent the same real referent create the apparition of co-governed nodes (*Mary* in the example Fig. 2.3(a)). In terms of graphs we are dealing with *directed acyclic graphs*, or more general graphs.

However we are not going to need such representations. The following chapters should prove that the unique kinds of structures really necessary to account for grammaticality are dependency trees for a syntactic representation.

Let us show a simple example of this. We suppose a simplified model with three representations: semantic, syntactic and word-order.[2] Consider the pair of sentences in French:

(1) Marie veut chanter. / Marie veut que Jean chante.
Mary wants to-sing / Mary wants that John sing
'Mary wants to sing. / Mary wants John to sing.'

In the semantic representation of the first sentence the subject of the main clause and the subordinate clause points to the same element, Fig. 2.3(a). In a syntactic representation the second subject is gapped, Fig. 2.3(b). Finally Fig. 2.3(c) shows a linear dependency tree for the word-order.

[1] The term "functionality" was suggested by an anonymous referee of (Cardó, 2016).

[2] Of course, usual formalisms enrich this architecture with more modules or layers. For example in *Meaning Text Theory* (MTT) (Melčuk, 1988) seven layers are considered: one for a semantic representation, two for syntactic representation and the rest for the morphological representation. In Topological Dependency Grammar (TDG) (Debusmann and Duchier, 2002*b*) four modules are postulated, namely: *syntax, topology, semantic arguments, and semantics*. Later, these modules are called *dimensions*. See eXtensible Dependency Grammar (XDG) (Debusmann, Duchier and Kruijff, 2004; Debusmann et al., 2005).

However the above is a simplified example in order to illustrate that only dependency trees and a word-order are necessary to consider the grammaticality.

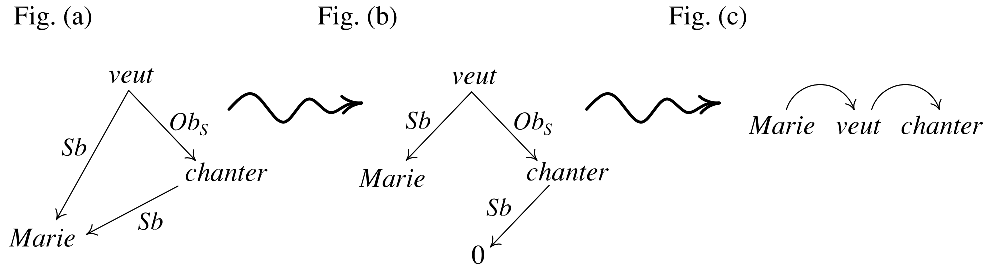


Figure 2.3: (a) semantic representation, (b) syntactic representation, and (c) word-order representation of the sentence *Marie veut chanter*.

In Romance languages the form of the subordinate verb varies according the presence or absence of the subordinate subject. In the first sentence the subordinate verb must be in infinitive form, *chanter*. In the second sentence where the subject of the subordinate clause is different from the main clause the verb takes a finite form, *chante*. We do not need the semantic representation in order to capture this. We only need to stipulate that subordinate verbs without subject must be in infinitive form, otherwise they must be in finite form (and, in particular, the verb must be in subjunctive and it must agree in number with the subordinate subject). This stipulation can be made in a tree-shaped dependency graph.

This is a declarative manner of understanding grammaticality. However there is an alternative, procedural, explanation. One begins from a semantic representation and then the syntax-semantic interface operates adequate changes: if both subjects coincide then we delete the second subject and we put the verb in infinitive. Ultimately, the two interpretations need not be incompatible, indeed they should coincide.[3]

The main propose of this thesis is giving a dependency based formalism to account for grammaticality, i.e. providing a grammar which defines a set of correct sentences of natural language. We do not try to study anaphora mechanisms or transformations of a semantic representation preserving the meaning (Melčuk, 1988). Neither are we concerned in the logical readings of a sentence or any other properly semantic question.[4] Since we are interested just in the problem of grammaticality trees should suffice. All in all, this assumption can be paraphrased as:

> *Claim* 1. The problem of grammaticality is a syntactic problem.

That is, we do not need to know to whom a pronoun refers to know that a sentence

[3] Let us show that this is not surprising through a very naive mathematical metaphor. Consider the set of even numbers, say E. One can describe this set "declaratively" as $E = \{x \in \mathbb{N} \,|\, 2 \text{ divides } x\}$, or through a mapping (that is through an "interface") $f : \mathbb{N} \longrightarrow \mathbb{N}$, $f(x) = 2x$. E is the image of f, $E = \operatorname{im} f$. Both coincide.

[4] These questions remain for future study, but here we deal only with the problem of grammaticality.

is grammatical and we do not need understand all the possible logical readings of a sentence to know that the sentence is well-formed.

This is not exempt of philosophical consequences. For instance, assuming this means that a native English speaker can become a Chinese corrector without understanding anything of Chinese at all. Such a person can correct a Chinese text but he cannot establish a Chinese conversation.[5] However our interests are focused rather in effective procedures and mathematical structures than in philosophical discussions.

Functionality

A first objection to functionality could be made through sentences such as:

(2) a. Mary left yesterday at five.
  b. This girl, Mary, left yesterday.

A possible interpretation is that, for example in the case (2a), *yesterday* and *at five* are both adjuncts of time and that they must be placed in a same level in the dependency tree, as in Fig. 2.4(a.i). Similarly for the sentence (2b) two subjects could be observed: *This girl* and *Mary*, Fig. 2.4(b.i). Nevertheless, we opine that these possible duplications of functions must be interpretated hierarchically. In the first example, the constituent *at five* specifies a part of the day, so we can interpret it as a noun complement which complements *yesterday*, as in Fig. 2.4(a.ii).[6]

Similarly the second sentence is interpreted better as Fig. 2.4(b.ii), where the noun complement is specifying the name of the girl. One can paraphrase the sentence as *This girl, whose name is Mary, left yesterday*.

Functionality is closely related to the issue of coordination which is a controversial point in the dependency based approaches, whereby we devote the Appendix A to the subject. Although the discussions and argumentations are placed in that appendix, let us to comment rapidly here the main issues.

The problem of coordination arises already from the modern foundation of dependency grammar, *Eléments de syntaxe structurale* (Tesnière, 1959, pg. 323-356). One is tempted to use the possibility of non-functionality to form groups of conjuncts. However when we have a combination of different coordinators this strategy is insufficient.

[5] This could be considered the linguistic analogue of Searle's *Gedankenexperiment* (Searle, 1980). In our Chinese room we judge the claim *syntax-suffices-for-grammaticality* instead of the original Searle thought experiment which judges the claim *Turing-machines-suffice-for-human-intelligence*.

[6] An argument for this comes from considering the same sentence in a Romance language. Such languages place generally the complements at the right. Although both sentences are grammatical, the first option prevails, which suggests that *a las cinco* is a complement of *ayer*.

(3) María se marchó ayer a las cinco. / $^{?}$María se marchó a las cinco ayer.
Mary her-self left yesterday at the five / Mary her-self left at the five yesterday
'Mary left yesterday at five. / Mary left at five yesterday.'

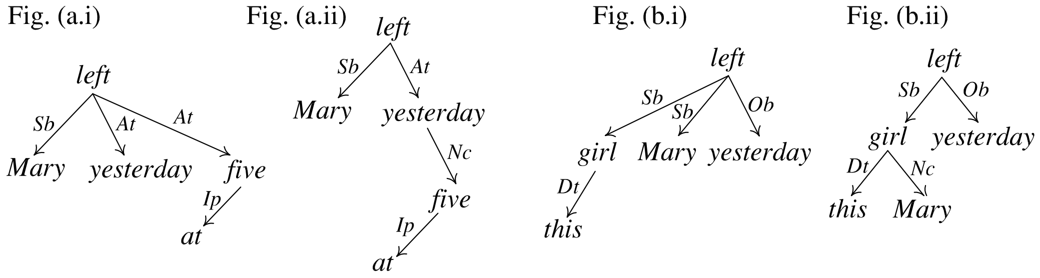


Figure 2.4: (a.i) non-functional and (a.ii) functional analysis of the sentence *Mary left yesterday at 17:00*; (b.i) non-functional and (b.ii) functional analysis of the sentence *This girl, Mary, left yesterday*.

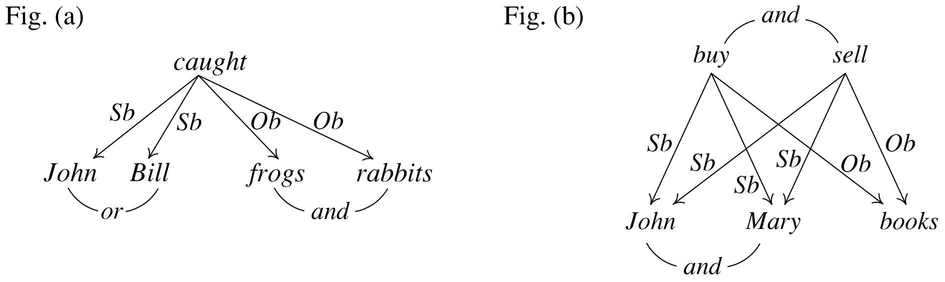


Figure 2.5: (a) Tesnière's solution for coordination; (b) Tesnière's solution with a non-tree-shaped underlying dependency tree.

Tesnière's solution consisted in linking conjuncts; see for example Fig. 2.5(a). Nevertheless this is not yet a pure dependency structure since it contain an extra relation. In fact some authors claim that a bare dependency structure does not suffice in order to capture coordination whereby the structure must be enriched; see in this respect (Kahane, 1997; Lison, 2006).

However, enriching structures has some inconveniences. We just comment two of them. Firstly, some styles of coordination lead to a non-tree shaped analysis, see Fig. 2.5(b), even for a syntactic representation, which makes the linearization less direct.[7] Secondly, adding a supplementary structure makes more complex the mathematical and linguistic treatment.

For these and other reasons (see Appendix A) we advocate for a pure dependency-tree solution affine to the (Melčuk, 1988) style. Such a solution consist in choosing one of the conjuncts and then concatenating the rest of the conjuncts by a syntactic function, say *Coor*. See Fig. 2.6(a) and a variant of this solution in Fig. 2.6(b) and (c) which uses syntactic functions in order to differentiate kinds of coordination and which is the style

[7] Although not impossible. Kuhlmann and Jonsson (2015) consider linearizations also for non-tree-shaped dependency graphs.

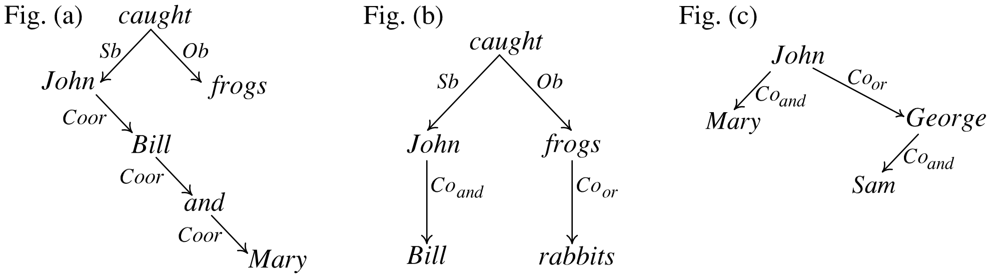


Figure 2.6: (a) Mel'čuk's solution for coordination; (b) a variant of Mel'čuk's solution which differentiates kind of coordination; (c) combination of different coordinations in Mel'čuk's solution.

used in the thesis.[8] The result is a tree-shaped and functional dependency structure.

2.5 ♣ Why Are Syntagmata Suitable? Mathematical Considerations

As we have said the two properties, tree-shape and functionality, characterize syntagmata. First of all we need to formalize both notions. A usual mathematical framework for dependencies is the language of graphs.

Definition 2.10. We define a *dependency graph* as a graph $G = (V, A)$ with vertexes V and edges $A \subseteq V \times V$ together with a pair of mappings $f : V \longrightarrow \Sigma$ and $g : A \longrightarrow \zeta$ which label vertices with letters in Σ and edges with syntactic functions in ζ. We notate these $\mathcal{G} = (G, f, g)$.

Then two dependency graphs $\mathcal{G} = (G, f, g)$ and $\mathcal{G}' = (G', f', g')$ are isomorphic, notated $\mathcal{G} \cong \mathcal{G}'$, iff there is a bijective mapping $\Psi : V \longrightarrow V'$ such that it is a isomorphism of graphs and it commutes with the mappings f, f' and g, g'.

A dependency graph is *tree-shaped* iff the underlying graph is. We write: $x \overset{\alpha}{\rightarrow} y$ iff there are two vertices $x, y \in V$ such that $g(x, y) = \alpha$. Then a dependency graph is *functional* iff for any vertices $x, y, y' \in V$ and for any syntactic function $\alpha \in \zeta$:

$$x \overset{\alpha}{\rightarrow} y,\ \ x \overset{\alpha}{\rightarrow} y' \implies y = y'.$$

We call a *path* a configuration of the form: $x_1 \xrightarrow{\alpha_1} x_2 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_{t-2}} x_{t-1} \xrightarrow{\alpha_{t-1}} x_t$. Similarly to the practice in automata theory we notate this $x_1 \xrightarrow{\alpha_t \cdots \alpha_1} x_t$ with $\alpha_t \cdots \alpha_1 \in \zeta^*$. The reversed order of the string of syntactic functions is just a convention.

Then it is easy to prove that tree-shaped and functional dependency graphs are syntagmata, up to notations. More formally, for each functional dependency tree $\mathcal{G}$

[8] The position of the coordinators is not shown in order to exhibit the bare structure of the functions chaining the conjuncts.

with root u we define the syntagma $S_{\mathcal{G}} : \zeta^* \longrightarrow \Sigma_+$:

$$S_{\mathcal{G}}(\varphi) = \begin{cases} f(x) & \text{if } u \xrightarrow{\varphi} x; \\ 0 & \text{otherwise.} \end{cases}$$

This syntagma is well defined and furthermore:

$$\mathcal{G} \cong \mathcal{G}' \iff S_{\mathcal{G}} = S_{\mathcal{G}'}.$$

This means that representation of functional dependency trees as syntagmata is faithful: no functional tree is lost. We know that for trees, for every node there is a unique path from the root to the node. In addition if the dependency tree is functional this path can be described by a string, or locus, in ζ^*, since a syntactic function cannot appear twice as an immediate descendant of a node. Since this construct does not take account of the vertices nor the edges in themselves, only the paths, $S_{\mathcal{G}}$ is the same for isomorphic functional dependency trees. Conversely, given a syntagma S we can reconstruct a functional dependency tree $\mathcal{G}$ such that $S_{\mathcal{G}} = S$ and it is unique up to isomorphism. Notice, however that there are more syntagmata than classes of isomorphic functional dependency trees. The reason for this is that syntagmata allow ellipses or holes in the structure.

In the sequel "syntagma" will mean a functional tree-shaped dependency structure with the possibility of gaps or ellipses. The first advantage of using syntagmata is the simplicity in their notation. A second advantage is provided by the above equivalence. Two dependency graphs with the same shape and labels are isomorphic although their set of vertices can be different. An isomorphism of dependency graphs means a linguistic "equality". However regarding syntagmata, a pure equality of syntagmata means a linguistic "equality", without the mediation of external isomorphisms. This will simplify the notation for the concepts that we will present.

3

Patterns and Classificatory Monoids

This chapter continues introducing basic concepts. A *monoidal pattern* is a subset of the monoid ζ^{a*} defined as a product of *constants* (singletons) and *submonoids* of ζ^{a*}. The number a is called the arity of the pattern. So patterns are certain types of rational sets, and when the arity is one, they are a subclass of regular languages: that closed by Kleene star and concatenation.

If we notate $\mathbf{k}$ the class of constants and $\mathbb{M}$ the class of submonoids, every pattern can be described in terms of these two basic classes as an element of the free monoid $\{\mathbf{k}, \mathbb{M}\}^*$ which we call the *classificatory monoid* the elements of which are called *types*.

This is the most basic form of describing the structure of a pattern, but in order to differentiate kinds of sets of syntagmata, linearizations, formal languages, and natural constructions we will need more detailed classificatory monoids. We introduce them and study the elementary properties.

3.1 Definitions

Monoidal Patterns are subsets of the non-free monoid $\zeta^{*a} = \bigoplus_{i=1}^{a} \zeta^*$ formed by products of *submonoids* and *constants*.[1] Here "constant" stands for singleton sets. In order to introduce them we need a norm for the elements in ζ^{*a} given by:

$$|(x_1, \ldots, x_a)| = \max_{i=1,\ldots,a} \{|x_i|\},$$

which extends the length for arities ≥ 1.

Definition 3.1. Given a fixed finite set ζ of syntactical functions consider the monoid ζ^{*a}. We call a *1-norm constant of arity a* a set consisting of one element $\{\varphi\} \subseteq \zeta^{*a}$ such that $|\varphi| \leq 1$. We call a *1-norm submonoid of arity a*, a finitely generated submonoid

[1] In the literature on formal languages, *pattern languages* are a class of languages due to Angluin (1980*a*,*b*). However there is no relation between these two concepts. Since we happen to use Angluin pattern languages later, in order to avoid any confusion we will rename them as *Angluin languages*. For the rest of the thesis the term "pattern" will stand for "monoidal pattern".

$\langle\varphi_1,\dots,\varphi_m\rangle \subseteq \zeta^{*a}$ such that $|\varphi_i| \leq 1$ for $i = 1,\dots,m$. We will use the star operator instead of the brackets: $\{\varphi_1,\dots,\varphi_m\}^* = \langle\varphi_1,\dots,\varphi_n\rangle$.

The following lemma justifies the notation $(\cdot)^*$ for the 1-norm submonoids.

Lemma 3.2. *Consider a 1-norm submonoid $\langle\varphi_1,\dots,\varphi_n\rangle \subseteq \zeta^*$ of arity 1. If $x \in \langle\varphi_1,\dots,\varphi_n\rangle$, then all the prime factors of x are in $\{\varphi_1,\dots,\varphi_n\}$. Equivalently, $\langle\xi\rangle \subseteq \zeta^*$ is a 1-norm submonoid if and only if $\xi \subseteq \zeta$.*

Proof. Since $x \in \langle\varphi_1,\dots,\varphi_n\rangle$, x can be expressed through these generators. By assumption it is a 1-norm submonoid and then $|\varphi_1| = 1,\dots,|\varphi_n| = 1$. Since the arity is 1, every φ_i is in ζ. This means that the factorization of x in generators $\varphi_1,\dots,\varphi_n$ is a factorization in prime factors in ζ. The equivalence is trivial. □

Definition 3.3. A *monoidal pattern*, or simply a *pattern* of arity a is a finite product of 1-norm constants and 1-norm submonoids. In other words a pattern Γ is an expression of the form:

$$\prod_{i=1}^{k} \Gamma_i,$$

where each Γ_i is a 1-norm constant or 1-norm submonoid, which we call *basic patterns*. Given a pattern Γ, we call the *length* of Γ the least number k such that $\prod_{i=1}^{k} \Gamma_i$ for some basic patterns Γ_i, $i = 1,\dots,k$. When we fix a specific decomposition we call k the *length of that decomposition*.

When the context permits, we will say simply constants and submonoids, or even simply monoids. Frequently we will write φ instead $\{\varphi\}$. For example, we may write $\zeta^*\varphi$ instead $\zeta^* \cdot \{\varphi\}$.

The most common patterns in the examples are the *trivial* patterns $\{(1)_a\}$, the *full* pattern ζ^{*a} and the *ideal* patterns $\zeta^{*a}\varphi$, $\varphi\zeta^{*a}$, $\varphi\zeta^{*a}\psi$. But we also will use more complex ones such as that in the next example.

Example 2. Some patterns. Constants are patterns of length 1. Submonoids are patterns of length 1. The trivial monoid $\{1\} \subseteq \zeta^*$ is a pattern of arity 1 because it is simultaneously a constant and a submonoid and it has length 1. The trivial monoid of arity 2 $\{(1)_2\} = \{(1,1)\} \subseteq \zeta^{*2}$ or any other arity is also a pattern of length 1.

The full monoid ζ^* is a pattern of arity 1 and length 1. More generally ζ^{*a} is a pattern of arity a of length 1, since $\{\alpha,\beta\}^{*2} = \{(\alpha,1),(\beta,1),(1,\alpha),(1,\beta)\}^*$.

The specific decomposition $\zeta^* = \zeta^*{\cdot}\zeta^*$ has length 2, but really the pattern ζ^* has length 1. So the "real" length is always less or equal than the length of a decomposition. The bilateral ideal pattern $\varphi\zeta^*\psi$ has length 3 provided that $\varphi,\psi \neq 1$.

$\{\alpha^2\}^*$ is not a pattern, because it is a submonoid generated by a element with $|\alpha^2| = 2$. $\alpha\{\alpha, \beta\}^*\gamma^2$ is a pattern of arity 1, or $(1, \gamma^3)\zeta^{*2}$ is a pattern of arity 2.

The set $\{(\alpha\beta^i\gamma\alpha^j, \gamma\alpha^i) \mid i, j \in \mathbb{N}_+\}$ is a monoidal pattern of arity 2 and length 4 because it can be written as $(\alpha, \gamma)\cdot(\beta, \alpha)^*\cdot(\gamma, 1)\cdot(\alpha, 1)^*$. However the set $\{(\alpha^i\beta^j, \beta^j\alpha^i) \mid i, j \in \mathbb{N}_+\}$ is not a monoidal pattern because the set in itself is not a monoid nor a constant and it cannot be decomposed into a succession of products of submonoids and constants. On the other hand, the set $\{(\alpha^i\beta^j, \alpha^i\beta^j) \mid i, j \in \mathbb{N}_+\}$ can be written as: $(\alpha, \alpha)^*\cdot(\beta, \beta)^*$, and this shows that it is a pattern of length 2.

Let us see a last example. $\{(\alpha^n, \alpha^m) \mid n \le m\}$ is a pattern of arity 2 and lenght 2; just consider the product $(\alpha, \alpha)^*(1, \alpha)^* = \{(\alpha^i, \alpha^{i+j}) \mid i, j \in \mathbb{N}_+\}$ and that $i \le i + j$ $\forall i, j \in \mathbb{N}_+$.

In particular when the arity is 1, we have a formal language over the strings of syntactic functions, a kind of regular language. So monoidal patterns can be viewed as a generalization in several components of this type of regular languages. Indeed:

Remark 3.4. *Rational sets* are a generalization of regular languages over free monoids to general monoids. More exactly (we follow definitions from (Berstel, 1979; Sakarovitch, 2009)): let M be a monoid; the class Rat(M) of rational subsets of M is the least class $\mathcal{R}$ of subsets of M satisfying the following conditions:

(i) $\emptyset \in \mathcal{R}$; $\{m\} \in \mathcal{R}$ for all $m \in M$;

(ii) if $A, B \in \mathcal{R}$, then $A \cup B$, $A\cdot B \in \mathcal{R}$;

(iii) if $A \in M$ then $A^* \in \mathcal{R}$.

Thus any subset A of M obtained from singletons by a finite number of unions, products and star operations is in Rat(M). In particular we have that for free monoids Rat(ζ^*) is the set of regular languages over the alphabet ζ. And more generally, Rat(ζ^{a*}) contains the set of patterns of arity a over the functions ζ. Notice that we do not need the union closure in the case of patterns.

We can make more precise the inclusion of patterns in rational sets. The well-known notion of star-height of a rational set is a measurement of its complexity. Let M be a monoid, and define inductively sets $\mathrm{Rat}_0(M) \subset \mathrm{Rat}_1(M) \subset \cdots$ by:

$$\begin{aligned} A \in \mathrm{Rat}_0(M) &\iff A \text{ is a finite subset of } M \\ A \in \mathrm{Rat}_{k+1}(M) &\iff A \text{ is a finite union of sets of the form } B_1B_2\cdots B_n \end{aligned}$$

where B_i is a singleton or a set of the form $B_i = C_i^*$ for some $C_i \in \mathrm{Rat}_h(M)$. This decomposes rational sets as $\mathrm{Rat}(M) = \bigcup_{h\ge 0} \mathrm{Rat}(M)$. The sets $\mathrm{Rat}_h(M) \setminus \mathrm{Rat}_{h-1}(M)$ are said to have *star-height h*. Trivially every pattern has star-height 0 or 1, which means that they occupy the simplest strata in this hierarchy.

> Although rational sets generalize regular languages, some computational properties are lost when we pass from arity 1 to greater arities, see Appendix C.

Given two patterns Γ, Γ', the cartesian product $\Gamma \times \Gamma'$ is also a pattern. We will notate this pattern $\Gamma \oplus \Gamma'$ to emphasize that the product of its elements is taken componentwise. If Γ and Γ' have respectively arities a and b then $\Gamma \oplus \Gamma'$ is a pattern with arity $a + b$.

The j-projection is the mapping $\pi_j : \bigoplus_{i=1}^{a} \Gamma_i \longrightarrow \Gamma_j$, defined $\pi_j(x_1, \ldots, x_k) = x_j$.[2]

3.2 Unambiguous and Proper Patterns

By definition every pattern, say Γ, has at least one *decomposition* or *factorization in basic patterns*, that is a succession $\Gamma_1, \ldots, \Gamma_k$, where Γ_i is a constant or a monoid for all $i = 1, \ldots, k$, such that $\Gamma = \prod_{i=1}^{k} \Gamma_i$. Notice that a factorization suggests automatically a mapping $\pi : \bigoplus_{i=1}^{k} \Gamma_i \longrightarrow \Gamma$, given by the product of the components: $\pi(x_1, \ldots, x_k) = x_1 \cdots x_k$. We call this mapping the *product mapping of the factorization*.

However, a pattern has many factorizations. For example we can always add a trivial factor: $\Gamma = 1 \cdot \Gamma \cdot 1 \cdot 1 \cdot 1$. Or for example, $\zeta^* = \zeta^* \cdot \zeta^*$. More generally, if $\xi \subseteq \zeta$ then $\xi^* \zeta^* = \zeta^*$. These decompositions have a redundancy of factors. Finally we have in certain cases the same factorization with distinct order. For example $\alpha^3\alpha^*\alpha = \alpha^2\alpha^*\alpha^2 = \alpha\alpha^*\alpha^3 = \alpha^*\alpha^4 \ldots$.

Definition 3.5. Given a pattern with the decomposition $\Gamma = \Gamma_1 \cdots \Gamma_k$ we say that the decomposition is *unambiguous* iff we have that:

$$x_1 \cdots x_k = y_1 \cdots y_k, \text{ with } x_i, y_i \in \Gamma_i \implies x_i = y_i, \forall x = 1, \ldots, k.$$

There are other ways to say the same thing. For example the decomposition is unambiguous iff every $x \in \Gamma$ has a unique factorization in basic patterns, $x = x_1 \cdots x_k, x_i \in \Gamma_i$. Another way to say the same is that the product mapping is bijective. So for example the decomposition $\zeta^* = \zeta^* \cdot \zeta^*$ is not unambiguous. However this still allows trivial decompositions. For example $\zeta^* = \zeta^* \cdot \{1\}$ is unambiguous: each $x \in \zeta^*$ has the unique factorization $x \cdot 1$, with $x \in \zeta^*, 1 \in \{1\}$.

Definition 3.6. We say that a decomposition of a pattern $\Gamma = \Gamma_1 \cdots \Gamma_k$ with arity a is *proper* iff either $k = 1$ and $\Gamma = (1)_a$ or the decomposition is unambiguous and the basic patterns are not trivial. A pattern is proper iff it has at least one proper factorization.

Thus ζ^* has only one proper factorization, ζ^* itself. On some occasions a pattern has no proper factorization because it has no unambiguous factorization, for example $\zeta^*\alpha\zeta^*$, with $\zeta = \{\alpha, \beta\}$. Such cases are called in the theory of rational sets *inherently ambiguous*, see (Berstel, 1979; Sakarovitch, 2009).

[2] Of course, j-projection is different from projection in the habitual sense of dependency grammar.

3.3 Classificatory Monoids

In the following chapters we will see that certain classes of sets of trees, linearizations, formal languages, and natural constructions can be differentiated by the forms of the patterns involved.

The very definition of patterns suggests a form of describing them. For example the pattern $(1, \alpha)(\alpha, \beta)^*(\gamma, \gamma)$ is formed as a constant by a monoid by a constant. The following definitions formalize this simple idea and generalize it to obtain several systems of classification.

Given a vector $x = (x_1, \ldots, x_a) \in \zeta^{a*}$, we notate $\varphi^R = (x_1^R, \ldots, x_a^R)$ the *reversed vector*. Given a pattern Γ we notate $\Gamma^R = \{x^R \mid x \in \Gamma\}$ the *reversed pattern*.

Definition 3.7. Let ζ be a fixed set of syntactic functions. We call a *basic type*, $\mathbf{T}$ a set of patterns over ζ satisfying:

(i) $\{(1)_a \mid a \geq 1\} \subseteq \mathbf{T}$;

(ii) If $\Gamma \in \mathbf{T}$, then $\Gamma^R \in \mathbf{T}$.

Given a finite set of basic types $\mathbf{T}_1, \ldots, \mathbf{T}_n$ we call the free monoid $\{\mathbf{T}_1, \ldots, \mathbf{T}_n\}^*$ a *classificatory monoid*. We call its elements *types*. We notate its identity as $\mathbf{1}$.

Let X be a type $X \in \{\mathbf{T}_1, \ldots, \mathbf{T}_n\}^*$, $X \neq \mathbf{1}$, with a decomposition in prime factors $X = X_1 \cdots X_k$. We say that a pattern $\Gamma \subseteq \zeta^{a*}$ *has type* X iff Γ can be decomposed as $\Gamma = \Gamma_1 \cdots \Gamma_k$ and $\Gamma_i \in X_i$, for each $i = 1, \ldots, k$. By convention patterns like $(1)_a$, $a \geq 1$ have type $\mathbf{1}$; we even write $\mathbf{1} = \{(1)_a \mid a \geq 1\}$.

Example 3. For a fixed ζ we will consider the following basic types:

- $\mathbf{k} = \{\Gamma \subseteq \zeta^{a*} \mid \Gamma \text{ is a 1-norm constant}, a \geq 1\}$, called the *constant type*;
- $\mathbb{M} = \{\Gamma \subseteq \zeta^{a*} \mid \Gamma \text{ is a 1-norm submonoid}, a \geq 1\}$, called the *monoid type*;
- $\mathbb{H} = \{(\zeta)_a^* \mid a \geq 1\}$, called the *homogeneous type*;
- $\mathbb{G} = \{(\xi)_a^* \mid \xi \subseteq \zeta, a \geq 1\}$, called the *generalized homogeneous type*;
- $\mathbb{P} = \mathbf{1} \cup \{1 \oplus \xi^* \mid \xi \subseteq \zeta\}$, called the *pivoting type*.

One can check that these are really basic types. For instance, we notice that $(1)_a$ is a constant for any $a \geq 1$. In addition if $\{\varphi\} \in \mathbf{k}$, then $|\varphi| \leq 1$, whereby $\varphi^R = \varphi$. Similarly $(1)_a$ is also monoid for any $a \geq 1$ and since generators of monoids have norm 1, $\Gamma^R = (\{\varphi_1, \ldots, \varphi_n\}^*)^R = \Gamma$. Notice that the condition of 1-norm is necessary, otherwise this last is no longer true.

For formal languages we will use mostly the *general classificatory monoid* $\{\mathbf{k}, \mathbb{M}\}^*$ or the homogeneous classificatory monoid $\{\mathbf{k}, \mathbb{H}\}^*$. For example:

$$(\gamma, \gamma)(\alpha, \beta)^*(\alpha, \alpha)^*(\gamma, \gamma) \text{ has type } \mathbf{k}\mathbb{M}^2\mathbf{k},$$

$$(\alpha, \beta)(\alpha, \alpha)^* \text{ has type } \mathbf{k}\mathbb{H}.$$

The basic types $\mathbf{k}, \mathbb{P}, \mathbb{G}$ will be used in the final chapters for natural languages. Observe the inclusions, $\mathbb{H} \subseteq \mathbb{G} \subseteq \mathbb{M}$ and $\mathbb{P} \subseteq \mathbb{M}$. So these types give more detailed classifications. For instance:

$$(\beta, \beta)^*(\gamma, \gamma)(\alpha, \alpha)^* \text{ has type } \mathbb{G}\mathbf{k}\mathbb{G},$$

$$(\alpha, \beta)(1, \alpha)^* \text{ has type } \mathbf{k}\mathbb{P}.$$

Indeed one can use more combinations such as the classificatory monoid $\{\mathbf{k}, \mathbb{P}, \mathbb{H}, \mathbb{G}, \mathbb{M}\}^*$. For example $(1, \alpha)^*(\beta, \gamma)(1, \alpha)(\alpha, \alpha)^*(\beta, \beta)^*(\zeta)_2^*(\alpha, 1)(\alpha, \beta)^*$ has type $\mathbb{P}\mathbf{k}^2\mathbb{G}^2\mathbb{H}\mathbf{k}\mathbb{M}$.

Notice finally that when patterns have arity 1, the types $\mathbb{M}$ and $\mathbb{G}$ coincide. For example if $\zeta = \{\alpha, \beta\}^*$, then $\alpha\beta^*$ has type $\mathbf{k}\mathbb{M}$ or $\mathbf{k}\mathbb{G}$ indistinguishably, but it has no type $\mathbf{k}\mathbb{H}$.

Example 4. Patterns of arity one and two can be represented graphically; they yield geometrical configurations according to the types. Patterns of arity one depict a distribution of loci in the support of a syntagma. Let $\zeta = \{a, b\}$. Fig. (a) represents a pattern $\zeta^*\alpha\beta^2$ with type $\mathbb{M}\mathbf{k}^3$ which captures the subtree with root at $\alpha\beta^2$. Fig.(b) represents the pattern $\alpha\zeta^*$ which has type $\mathbf{k}\mathbb{M}$. Notice that this pattern has the reversed type of the former consisting in all the loci that begin with an α (or that finish with an α in the arboreal representation; recall that we use the inverse order to represent paths because doing so, this has linguistic meaning). In addition since these patterns have arity one, they can also be classified as $\mathbb{H}\mathbf{k}^3$ and $\mathbf{k}\mathbb{H}$, respectively.

Fig. (a)

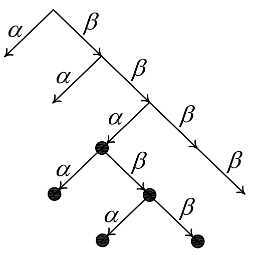
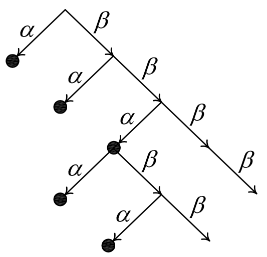
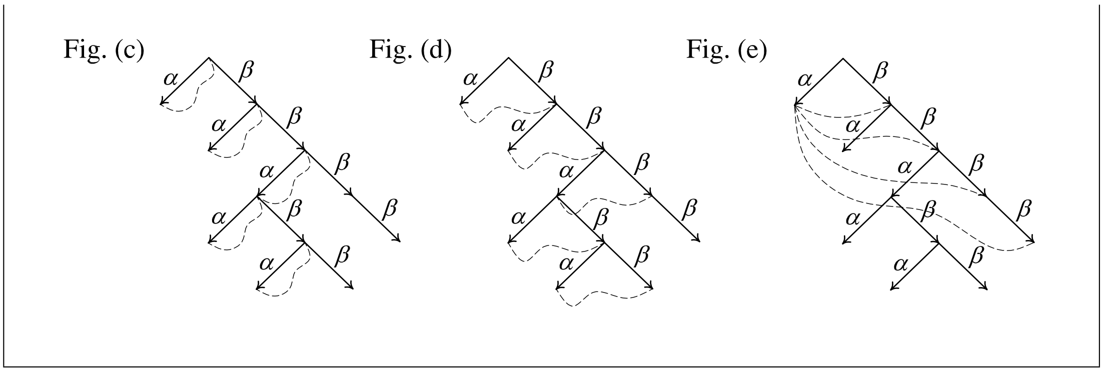


Fig. (b)



When the pattern has arity two, we can represent the pairs by a link (dashed lines in the figures). Fig.(c) represents the pattern $(1, \alpha)(\zeta)_2^*$ with type $\mathbf{k}\mathbb{H}$. Fig.(d) represents the pattern $(\alpha, \beta)(\zeta)_2^*$ also with the same type. Fig. (e) represents a pivoting pattern $(\alpha, \beta)(1 \oplus \beta^*)$ with type $\mathbf{k}\mathbb{P}$.



Remark 3.8. We fix some conventions on notations. In order to differentiate structures better we will notate $X^{\mathfrak{R}}$ the reverse of type X, instead of X^R, e.g. $(\mathbf{k}\mathbb{H})^{\mathfrak{R}} = \mathbb{H}\mathbf{k}$. The notation X^R will be reserved for elements in ζ^{*a}, e.g. $(\alpha\beta)^R = \beta\alpha$, or by extension patterns or subsets in ζ^{*a}, e.g. $(\alpha\zeta^*)^R = \zeta^*\alpha$. In addition, on some few occasions, which will be announced, we will notate the reversed type as $-X$.

By a mild abuse of notation, we will use as synonymous the expression "the pattern Γ has type X" and the expression "the pattern Γ is in X" or even "$\Gamma \in X$".

Given a finitely generated free monoid and two elements x, y, we say that x is a *subsequence* of y iff all the prime factors of x appear in the sequence of prime factors of y preserving the order. We write $x \sqsubseteq y$. This relation is a partial order. So for example: $1, cada, bcdb \sqsubseteq abracadabra$. The following properties on types will be used throughout the thesis.

Proposition 3.9. *Let X, Y be types in the classificatory monoid $\{\mathbf{T}_1, \dots, \mathbf{T}_n\}^*$.*

(i) $(1)_a \in X$, *for any type X and for any $a \geq 1$;*

(ii) If $\Gamma \in X$ and $\Theta \in Y$, then $\Gamma\Theta \in XY$;

(iii) If $\Gamma \in X$, and $X \sqsubseteq Y$, then $\Gamma \in Y$;

(iv) If $\Gamma \in X$, then $\Gamma^R \in X^{\mathfrak{R}}$.

Proof. (i) Let X be a type with prime factorization $X = X_1 \cdots X_k$, with $k \geq 1$. By definition of basic type, $(1)_a$ is included in any basic type, so $(1)_a$ has type X_i for each $i = 1, \dots, k$. Then we can write $(1)_a = (1)_a \cdots (1)_a$, k times, hence $(1)_a$ has type X. Now we suppose that $k = 0$, then $X = \mathbf{1}$ and by definition $(1)_a$ has type $\mathbf{1}$.

(ii) Consider the prime factorizations of the types $X = X_1 \cdots X_k$ and $Y = Y_1 \cdots Y_t$. We suppose that Γ decomposes as $\Gamma = \Gamma_1 \cdots \Gamma_k$ with $\Gamma_i \in X_i$, $i = 1, \dots, k$ and $\Theta = \Theta_1 \cdots \Theta_n$, and $\Theta_i \in Y_i$, $i = 1, \dots, t$. Clearly $\Gamma\Theta = \Gamma_1 \cdots \Gamma_k \Theta_1 \cdots \Theta_t$ has type $X_1 \cdots X_k Y_1 \cdots Y_t = XY$.

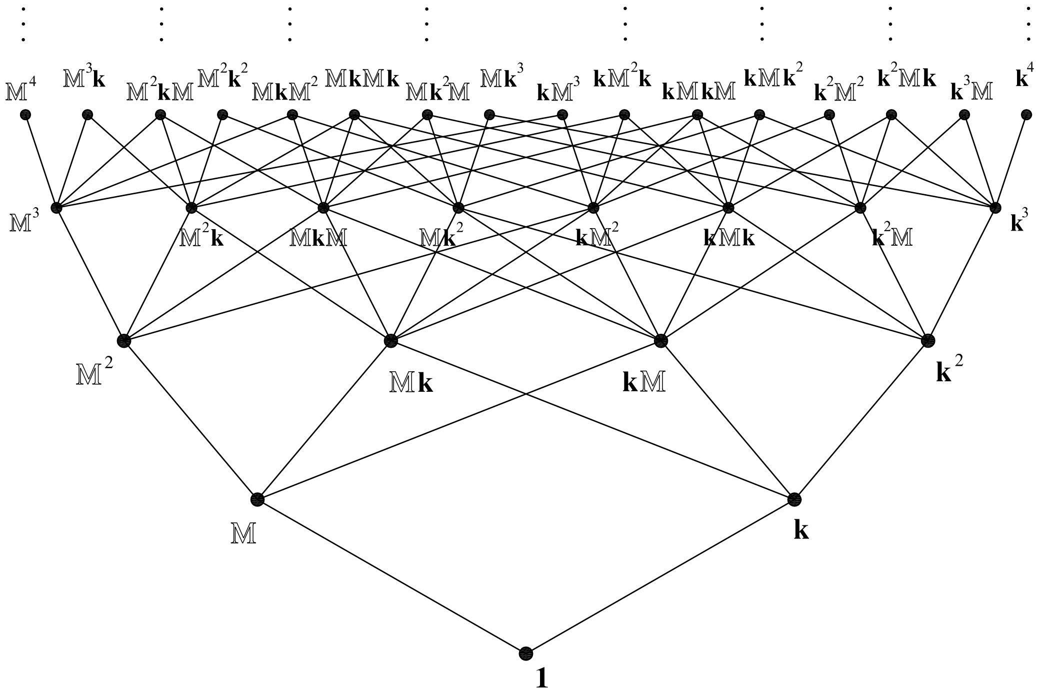

$\mathbb{M}^4$
$\mathbb{M}^3\mathbf{k}$
$\mathbb{M}^2\mathbf{k}\mathbb{M}$
$\mathbb{M}^2\mathbf{k}^2$
$\mathbb{M}\mathbf{k}\mathbb{M}^2$
$\mathbb{M}\mathbf{k}\mathbb{M}\mathbf{k}$
$\mathbb{M}\mathbf{k}^2\mathbb{M}$
$\mathbb{M}\mathbf{k}^3$
$\mathbf{k}\mathbb{M}^3$
$\mathbf{k}\mathbb{M}^2\mathbf{k}$
$\mathbf{k}\mathbb{M}\mathbf{k}\mathbb{M}$
$\mathbf{k}\mathbb{M}\mathbf{k}^2$
$\mathbf{k}^2\mathbb{M}^2$
$\mathbf{k}^2\mathbb{M}\mathbf{k}$
$\mathbf{k}^3\mathbb{M}$
$\mathbf{k}^4$
$\mathbb{M}^3$
$\mathbb{M}^2\mathbf{k}$
$\mathbb{M}\mathbf{k}\mathbb{M}$
$\mathbb{M}\mathbf{k}^2$
$\mathbf{k}\mathbb{M}^2$
$\mathbf{k}\mathbb{M}\mathbf{k}$
$\mathbf{k}^2\mathbb{M}$
$\mathbf{k}^3$
$\mathbb{M}^2$
$\mathbb{M}\mathbf{k}$
$\mathbf{k}\mathbb{M}$
$\mathbf{k}^2$
$\mathbb{M}$
$\mathbf{k}$
$\mathbf{1}$

Figure 3.1: Hasse diagram of subsequences in the classificatory monoid $\{\mathbf{k}, \mathbb{M}\}^*$.

(iii) If $X \sqsubseteq Y$ then there are types $Z_0, \ldots, Z_k$ such that $Y = Z_0 X_1 Z_1 \cdots X_k Z_k$, where $X = X_1 \cdots X_k$ is the prime factorization of X. Consider $\Gamma = \Gamma_1 \cdots \Gamma_k$ such that $\Gamma_i \in X_i$, $i = 1, \ldots, k$. We rewrite the pattern Γ as $\Gamma = (1)_a \Gamma_1 (1)_a \cdots (1)_a \Gamma_k (1)_a$. By (i) $(1)_a$ has every type, and in particular $(1)_a$ has types $Z_0, Z_1, \ldots Z_k$. By (ii), $\Gamma = (1)_a \Gamma_1 (1)_a \cdots (1)_a \Gamma_k (1)_a$ has type $Z_0 X_1 Z_1 \cdots X_k Z_k = Y$.

(iii) We notice that if $\Gamma = \Gamma_1 \cdots \Gamma_n$ is a decomposition in basic monoids, then Γ^R admits the decomposition $\Gamma^R = \Gamma_n^R \cdots \Gamma_1^R$. By definition of basic types, if $\Gamma_i \in X_i$ then $\Gamma_i^R \in X_i$, for $i = 1, \ldots, n$. So if Γ has type X, then Γ^R has type $X^{\mathfrak{R}}$. □

Fig. 3.1 shows the Hasse diagram of the subsequence relation in the general classificatory monoid.

Part II

THEORY

4

Manifolds

Syntactic manifolds constitute the first component for the construction of our formalism. We introduce *valuations* which are boolean conditions over the vocabulary. We apply such a condition at multiple loci in a syntagma which are indicated by a monoidal pattern. This combination (valuation + pattern) imposes a constraint on syntagmata, or in classical terms, yields a description of the dependencies in the tree. A (syntactic) manifold is the set of syntagmata that satisfy a valuation throughout a pattern. We motivate this design of *algebraic governance* by linguistic data. Manifolds can be classified by the type of the pattern used.

We review some classical formal languages and their relevance for linguistics, and we construct corresponding manifolds. In addition we learn some strategies in order to build manifolds and we see dependency trees for a small fragment of English.

We compare manifolds with other tree formalisms: local tree grammars and logic over trees. It is proved that manifolds with type $\mathbf{k}^p\mathbb{H}$, which have served to model the fragment of English, are essentially local.

4.1 Grouping Agreement: Linguistic Basis

In order to understand the central proposal of this chapter we are going to present some linguistic cases and extract an empirical hypothesis which serves us as heuristic for mathematical development.[1]

4.1.1 Verb Inflection in English

We are interested in grouping the several agreement instances of the words in a sentence. The most visible match is when two words are morphologically matched. Some languages are more profuse in this respect, for example Romance languages, however even English manifests some examples. In English when the subject of a sentence is the

[1] The material in this section is published as (Cardó, 2016).

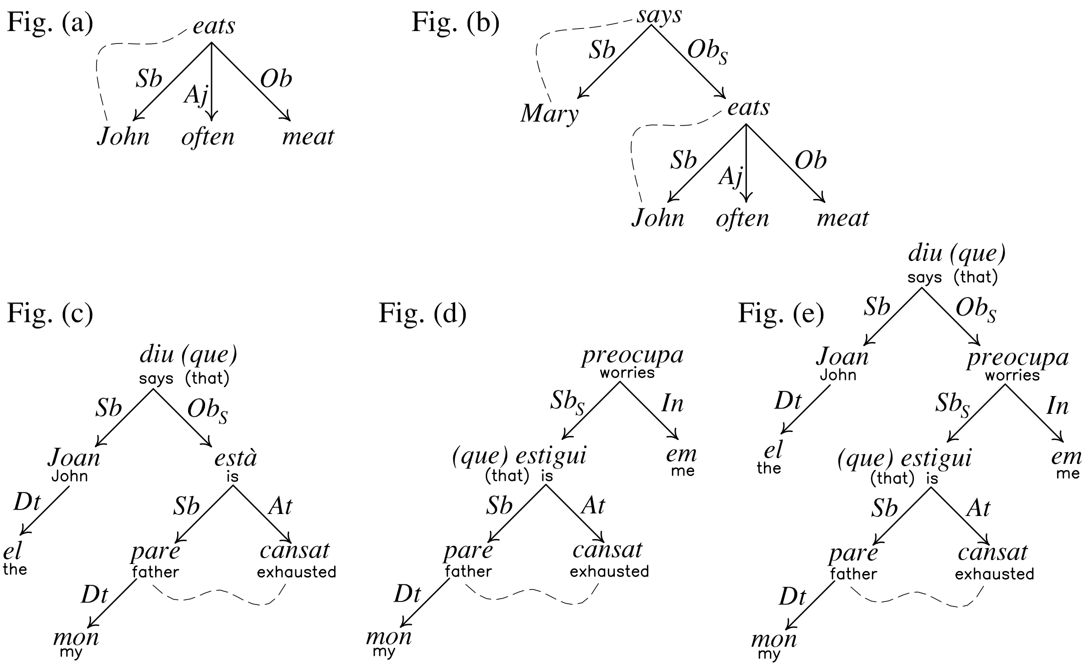


Figure 4.1: (a) and (b) agreement of subject and verb in English; (c) and (d), agreement of subject and attribute in Catalan.

third person singular and the main verb is present tense and not modal, the verb goes with an -s at the end. For example:

(4) $\underline{\text{John}}_{Sb}$ often $\underline{\text{eats}}_1$ meat.

We have underlined the morphologically matched words and subscripted the loci. The analysis is given in Fig. 4.1(a); the agreeing words are linked by a curve dashed line in this and the following examples.[2] Since the main verb is always allocated the locus 1 and the subject is in *Sb* we can represent this match as an ordered pair $(1, Sb)$. At the moment we are interested in the loci involved, not in the rule or condition. Now we can take the new sentence:

(5) $\underline{\text{Mary}}$ $\underline{\text{says}}$ $\underline{\underline{\text{John}}}_{Sb\cdot Ob_S}$ often $\underline{\underline{\text{eats}}}_{Ob_S}$ meat.

There are two matched pairs: a new pair $(1, Sb)$ and the old which is now in a deeper position $(Ob, Sb \cdot Ob)$. The sentence has an analysis as in Fig. 4.1(b). The agreement phenomenon iterates in deeper sentences: *Paul says Mary says John often eats meat*, *Peter says Paul says Mary says John often eats meat*, ... In general we can describe the set of the loci of all these agreements as: $Agr = \{(Ob_S^n, Sb \cdot Ob_S^n) \mid n \in \mathbb{N}_+\}$.

[2] Agreements (dashed lines) are not dependencies (arrows), but loci can be used to describe them.

4.1.2 Matching of Subject and Predicate

We can consider other languages and other types of matching. In some Romance languages the number and gender of the predicate (also called attribute) of a copulative verb (*to be, to look, . . .*) must be equal to that of the subject.

(6) Mon $\underline{\text{pare}}_{Sb}$ està $\underline{\text{cansat}}_{At}$. / Les meves $\underline{\underline{\text{germanes}}}_{Sb}$ estan $\underline{\underline{\text{cansades}}}_{At}$.
My $\underline{\text{father}}$ is $\underline{\text{exhausted}}$ / the my $\underline{\underline{\text{sisters}}}$ are $\underline{\underline{\text{exhausted}}}$

‘My father is exhausted. / My sisters are exhausted.’

These examples are in Catalan, however the same occurs in Spanish, French and Italian. The matched pair is (Sb, At). We also consider other examples in a deeper position:

(7) a. El Joan diu que mon $\underline{\text{pare}}_{Sb\cdot Ob_S}$ està $\underline{\text{cansat}}_{At\cdot Ob_S}$.
The John says that my $\underline{\text{father}}$ is $\underline{\text{exhausted}}$

‘John says my father is exhausted’.

b. Que mon $\underline{\text{pare}}_{Sb\cdot Sb_S}$ estigui $\underline{\text{cansat}}_{At\cdot Sb_S}$ em preocupa.
that my $\underline{\text{father}}$ is $\underline{\text{exhausted}}$ me worries.

‘The fact of my father being exhausted is worrying.’

c. El Joan diu que que mon $\underline{\text{pare}}_{Sb\cdot Sb_S\cdot Ob_S}$ estigui $\underline{\text{cansat}}_{At\cdot Sb_S\cdot Ob_S}$ em
The John says that that my $\underline{\text{father}}$ is $\underline{\text{exhausted}}$ me
preocupa.
worries

‘John says that the fact of my father being exhausted is worrying.’

The analyses are given in Fig. 4.1(c) and Fig. 4.1(d). The first agreement is $(Sb\cdot Ob_S, At\cdot Ob_S)$, but the second is $(Sb\cdot Sb_S, At\cdot Sb_S)$ and the third is $(Sb\cdot Sb_S\cdot Ob_S, At\cdot Sb_S\cdot Ob_S)$. This can be interpreted as that the set of agreements is in general

$$Agr = \{(Sb\cdot x, At\cdot x) \mid x \in \{Sb_S, Ob_S\}^*\},$$

in other words, this match, (Sb, At), would hold everywhere under arbitrary products of Sb_S and Ob_S.

4.1.3 Pied-piping in Some Romance Languages

We consider another phenomenon called *pied-piping* consisting in the embedding of a filler such as a relative pronoun within accompanying material from the extraction site. Consider the sentence in Catalan:

(8) Aquesta $\underline{\text{reina}}_{Sb}$, el pare del pare . . . del pare de
This $\underline{\text{queen}}$ the father of-the father . . . of-the father of
$\underline{\text{la}}_{Dt\cdot Nc\cdot Nc^n\cdot Sb\cdot Ad_S\cdot Sb}$ qual fou destronat, regnà justament.
$\underline{\text{the}}$ who was dethroned, reigned rightly

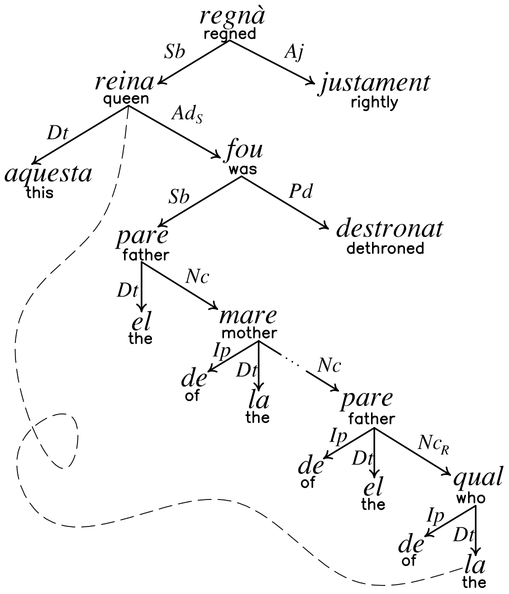


Figure 4.2: agreement in pied-piping in Catalan.

'This queen the father of the father ... of the father of whom was dethroned reigned rightly.'

The underlined words must agree in number and gender; see Fig. 4.2. In English there is not any match and we cannot underline any word because the pronoun is not preceded by any article. The agreement arises because the relative pronoun *qual* (*who*), although it has no gender inflection in Catalan, has as referent *reina* (*queen*) which is a feminine word. Since the relative pronoun must be complemented by an article, which does has gender inflection, this article must agree also with the referent: *la qual.*[3] The proof of this agreement is that in Catalan we cannot say:

(9) *Aquesta reina, el pare del pare ... del pare de el qual fou destronat, regnà justament.

Now the agreement set is

$$Agr = \{(Sb, Dt \cdot Nc \cdot Nc^n \cdot Sb \cdot Ad_S \cdot Sb) \mid n \in \mathbb{N}_+\}.$$

This is structurally revealing because there arises a monoid growing right in the middle of certain loci. But this phenomenon can occur anywhere, for example in the object:

[3] One can suppose that the relative pronoun has an invisible marker of gender. By the way a frequent syntactic error of Spanish children, where the same pied-piping agreement occurs, consists in inflecting forcibly the relative pronoun: *la cual-a* (then the adults humorously reprove them with the non-sense rime:... *la tia Pascuala*). Here we are only interested in the morphological matches, not in the operation of the anaphora.

(10) El poble no acceptà mai aquesta reina$_{Ob}$ el pare del pare ...
The populace not accepted never this queen the father of-the father ...
del pare de la$_{Dt\cdot Nc\cdot Nc^n\cdot Sb\cdot Ad_S\cdot Ob}$ qual fou destronat.
of-the father of the who was dethroned
‘The populace never accepted this queen the father of the father ... of the father of whom was dethroned.’

So the agreements are more general:

$$Agr = \{(x, Dt\cdot Nc\cdot Nc^n\cdot Sb\cdot Ad\cdot x) \mid n\in\mathbb{N}_+, x\in\zeta^*\}.$$

4.1.4 A Working Hypothesis

We can summarize all these agreements using subsets of the monoid $\zeta^{*2} = \zeta^*\oplus\zeta^*$. In the first case we have:

$$Agr = \{(Ob^n, Sb\cdot Ob^n) \mid n\in\mathbb{N}_+\} = \varphi\cdot\Gamma,$$

where $\varphi = (1, Sb)$, $\Gamma = (Ob, Ob)^*$. In the second case we also have:

$$Agr = \{(Sb\cdot x, At\cdot x) \mid x\in\zeta^*\} = \varphi\cdot\Gamma,$$

where $\varphi = (Sb, At)$, $\Gamma = \{(x,x) \mid x\in\{Sb_S, Ob_S\}^*\}$. Finally in the third case:

$$Agr = \{(x, Dt\cdot Nc\cdot Nc^n\cdot Sb\cdot Ad\cdot x) \mid n\in\mathbb{N}_+, x\in\zeta^*\} = \varphi\cdot\Gamma\cdot\psi\cdot\Gamma',$$

where $\varphi = (1, Dt\cdot Nc)$, $\Gamma = (1, Nc)^*$, $\psi = (1, Sb\cdot Ad)$ and $\Gamma' = \{(x,x) \mid x\in\zeta^*\}$. From the examples seen, and many others that could be given, we can extract a hypothesis.

Claim 2. Monoidal Pattern Hypothesis. For any natural language the set of places where agreements occur can be described as a monoidal pattern:

$$Agr = \prod_{i=1}^{k}\Gamma_i \subseteq \zeta^{*n}.$$

This is a weak initial hypothesis. In order to obtain a much better picture of natural languages we will have to strengthen this claim. This is one of several increasingly strong hypotheses which we will make. At the moment it must be taken as a guideline to be continued.

Even though the agreements shown were binary the monoidal hypothesis supports several arities, but the most common are 1, 2 and 3. For example: that *the subject Sb is* *always a noun* can be see as an agreement of arity 1 which must hold generally in all loci of a syntagma, therefore the pattern must be $Sb \cdot \zeta^*$.[4]

At the moment these examples must be taken as the motivation for the following definitions. In Chapter 10 we develop with more detail these constructions and others and we will see that some of the involved patterns are more complex.

4.2 Manifolds

Definition 4.1. Let Σ be a vocabulary and $a \in \mathbb{N}$ a positive integer. A *valuation* of arity a is a boolean function $B : \Sigma_+^a \longrightarrow \{0,1\}$. Here 0 and 1 are the truth values. A syntagma S *satisfies the valuation B throughout the pattern* $\Gamma \subseteq \zeta^{*a}$ iff

$$B\big(S(x_1), \ldots, S(x_a)\big) = 1 \quad \text{for each } (x_1, \ldots, x_a) \in \Gamma.$$

Some valuations that we will frequently use are the following which we write in prefix notation, although we will use also the infix notation:

- The *characteristic function* $(\in \sigma) : \Sigma_+^n \longrightarrow \{0,1\}$ defined as:

$$(\in \sigma)(x) = \begin{cases} 1 & \text{if } x \in \sigma; \\ 0 & \text{otherwise.} \end{cases}$$

- The *equality* $(\approx) : \Sigma_+^2 \longrightarrow \{0,1\}$ defined as:

$$(\approx)(x,y) = \begin{cases} 1 & \text{if } x = y; \\ 0 & \text{otherwise.} \end{cases}$$

- The curried restriction of the equality valuation $(\approx a)(x) = (\approx)(x, a)$:

$$(\approx a)(x) = \begin{cases} 1 & \text{if } x = a; \\ 0 & \text{otherwise.} \end{cases}$$

- The negations of above valuations:

$$(\notin \sigma)(x) = \begin{cases} 1 & \text{if } x \notin \sigma; \\ 0 & \text{otherwise.} \end{cases} \qquad (\not\approx)(x,y) = \begin{cases} 1 & \text{if } x \neq y; \\ 0 & \text{otherwise.} \end{cases}$$

In infix notation: $(x \in \sigma)$, $(x \approx y)$, $(x \approx a)$, $(x \notin \sigma)$, $(x \not\approx y)$ and $(x \not\approx a)$.

[4] We have been talking about morphological agreement but there is no reason not to talk about other kinds. When a verb like *to drink* selects a “drinkable" object it is producing a semantic “agreement” $(1, Ob)$, and so forth.

Example 5. A valuation and a Pattern. Let **Noun** $\subset \Sigma$ be the set of all nouns in English and let $Sb \in \zeta$ be a syntactic function. If we want Sb to always take an element from **Noun** in the syntagma S. The expression:

$$(\in \mathbf{Noun})(S(Sb)) = 1,$$

says exactly that. In infix notation:

$$(S(Sb) \in \mathbf{Noun}) = 1,$$

in other words this last states that $S(Sb) \in$ **Noun**. Let us take the constant pattern $\Gamma = Sb$. Then the syntagma in Fig. (a) satisfies the valuation in the pattern while Fig. (b) does not.

Fig. (a)

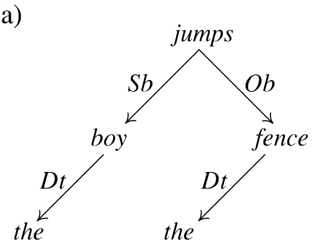
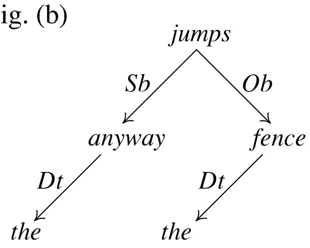


Fig. (b)



However we want this constraint to be hold in any locus of a syntagma. We can take the pattern $\Gamma = Sb{\cdot}\zeta^*$. Now the syntagma in Fig. (c) satisfies the valuation throughout Γ while Fig. (d) and Fig. (e) do not.

Fig. (c)

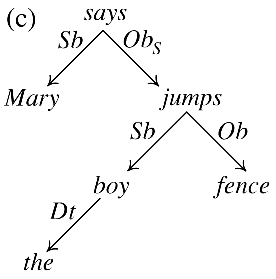


Fig. (d)

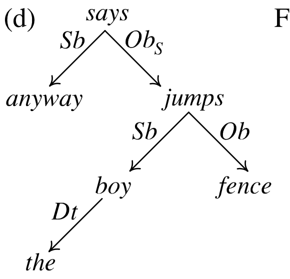
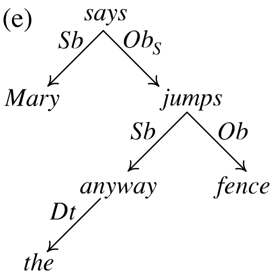


Fig. (e)



In view of the boolean nature of valuations we can define new valuations from others. Let a pair of valuations be $B : \Sigma_+^n \longrightarrow \{0,1\}$, $B' : \Sigma_+^m \longrightarrow \{0,1\}$; then we can define:

$$B \wedge B' : \Sigma_+^{n+m} \longrightarrow \{0,1\}$$

$$(x_1, \ldots, x_n, y_1, \ldots, y_m) \longmapsto B(x_1, \ldots, x_n) \wedge B'(y_1, \ldots, y_n).$$

In the same way we can define: $B \vee B'$, $B \rightarrow B'$, $\neg B$, and so forth. Composing new valuations and rules does not increase our descriptive power but it makes reading easier.

Example 6. A valuation and a pattern for transitive verbs. A transitive verb is a verb which always takes an object (e.g. the following sentence is ungrammatical: **John likes* ∅). Let **Trans** $\subset \Sigma$ be the set of transitive verbs and let Ob be a syntactic function. Consider the valuation in infix notation:

$$B(x, y) = \big(x \in \mathbf{Trans} \rightarrow y \not\approx 0\big).$$

The following expression asserts that the object cannot be null if a verb is transitive:

$$B\big(S(1), S(Ob)\big) = 1;$$

or by a mild abuse of notation: $1 \in \mathbf{Trans} \rightarrow Ob \not\approx 0$. As in the previous example we also want that the constraint to be hold in any place of a syntagma. In this case the arity is 2 and the pattern is $(1, Ob) \cdot \{(x, x) \mid x \in \zeta^*\} = (1, Ob) \cdot (\zeta)^*_2$.

Definition 4.2. Recall that we notated $\mathbf{Synt}_{\Sigma,\zeta}$ the set of all syntagmata with vocabulary Σ and syntactic functions ζ. Given a valuation B and a pattern Γ we define the *simple syntactic manifold* as the set:

$$\mathbf{Synt}_{\Sigma,\zeta}\begin{pmatrix} B \\ \Gamma \end{pmatrix} = \{S \in \mathbf{Synt}_{\Sigma,\zeta} \mid S \text{ satisfies } B \text{ throughout } \Gamma\}.$$

We call a *syntactical manifold* any ∪∩-combination of simple manifolds (i.e. any finite combination of unions and intersections). Given a type X in a classificatory monoid, we notate $\mathbf{Man}(X)$ the class of manifolds the simple manifolds of which have patterns in X.

To simplify we will write **Synt** omitting Σ and ζ. In addition, since we are going to use mostly intersections we abbreviate:

$$\mathbf{Synt}\begin{pmatrix} B_1 \cdots B_n \\ \Gamma_1 \cdots \Gamma_n \end{pmatrix} = \mathbf{Synt}\begin{pmatrix} B_1 \\ \Gamma_1 \end{pmatrix} \cap \cdots \cap \mathbf{Synt}\begin{pmatrix} B_n \\ \Gamma_n \end{pmatrix}.$$

An easy property is the following.

Corollary 4.3. *Simple manifolds reverse the inclusion of patterns. That is, for every patterns Γ, Γ' and any valuation B:*

$$\Gamma \subseteq \Gamma' \implies \mathbf{Synt}\begin{pmatrix} B \\ \Gamma \end{pmatrix} \supseteq \mathbf{Synt}\begin{pmatrix} B \\ \Gamma' \end{pmatrix}.$$

Proof. The valuation B is the same in both manifolds. If S satisfies B in any locus of Γ', since $\Gamma \subseteq \Gamma'$ then in particular S satisfies B for any locus in Γ. Hence the inclusion. □

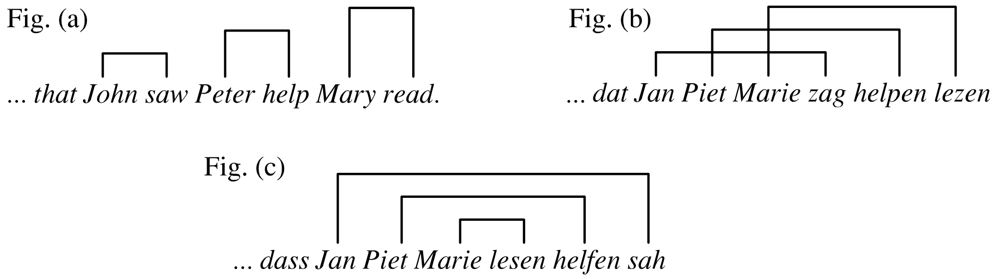


Fig. (a)
... that John saw Peter help Mary read.
Fig. (b)
... dat Jan Piet Marie zag helpen lezen
Fig. (c)
... dass Jan Piet Marie lesen helfen sah

Figure 4.3: (a) non-crossing dependencies in English; (b) serial-cross dependencies in Dutch; (c) nested dependencies in German.

Trivially every class $\mathbf{Man}(X)$ is a bounded lattice. Bounds are given by the false constant valuation 0, $\mathbf{Synt}\binom{0}{\Gamma} = \emptyset$, and the true constant valuation 1, $\mathbf{Synt}\binom{1}{\Gamma} = \mathbf{Synt}$ where Γ is any pattern in X.

Interestingly the descriptive power of a manifold is given not by the valuation but by the pattern, or more specifically the type of the pattern. However we will need to advance some chapters to understand this.

4.3 Some Examples of Manifolds

4.3.1 Five Classical Formal Languages and their Historical Relevance

We fix a vocabulary, for instance $\Sigma^* = \{a, b, c\}$, and consider the languages:

$$\begin{aligned} L_{\text{squa}} &= \{a_1^2 \cdots a_n^2 \mid a_1, \ldots, a_n \in \Sigma, n \in \mathbb{N}_+\}, \\ L_{\text{copy}} &= \{xx \mid x \in \Sigma^*\}, \end{aligned}$$

called respectively the *language of squares* and the *copy language*. The first is a context-free, indeed regular, language. It is a mathematical idealization of a chain of subordinate clauses in English. E.g.:

(11) . . . that John^a saw^a Peter^b help^b Mary^c read^c.

The second is not context-free and it represents a chain of subordinate clauses in Dutch, (Bresnan et al., 1987). E.g.:

(12) . . . dat Jan^a Piet^b Marie^c zag^a helpen^b lezen^c.
. . . that Jan Piet Marie saw help read
'. . . that J. saw P. help M. read.'

This example is said to exhibit *cross-serial dependencies*. Compare Fig. 4.3(a) and Fig. 4.3(b). The same crossing configuration is found in Swiss-German. The links are visible since Swiss-German has case marking:

(13) Jan säit das mer d'chind em Hans es huus lönd hälfe
John said that the children$_{ACC}$ Hans$_{DAT}$ the house$_{ACC}$ let help
aastriiche.
paint.

The abstraction into the formal language follows from the argument structure (i.e. we mark with a same letter the subject and the related verb) which is purely intuitive.[5] However the case of Swiss-German can be formally reduced into the copy language by intersecting the whole Swiss-German language with a regular language and reducing by a homomorphism of monoids according the case mark, see for example Kallmeyer (2010). Since context-free languages are closed under regular intersections and homomorphisms, see e.g. Hopcroft et al. (2001), if we suppose that natural languages are context-free, then the copy language should be. But this is not the case, so natural languages are not context-free.[6]

Closely related to the above languages we consider the *mirror language* defined as

$$L_{\text{mirr}} = \{xx^R \mid x \in \Sigma^*\},$$

where x^R is the reversed string. This language captures the nested dependencies in German (see Fig. 4.3(c)). E.g.:

(14) . . . dass Jan^{a} Piet^{b} Marie^{c} lesen^{c} helfen^{b} sah^{a}.
. . . that Jan Piet Marie read help saw
'. . . that J. saw P. help M. read'.

Interestingly we will see later that this language can be obtained from any of the above manifolds.

Another pair of languages that we are going to consider in the future are the *multiple abc language* and the *respectively abc language*:

$$L_{\text{mult}} = \{(abc)^n \mid n \in \mathbb{N}_+\},$$
$$L_{\text{resp}} = \{a^n b^n c^n \mid n \in \mathbb{N}_+\}.$$

The first is a context-free, indeed regular language, but not the second. The first corresponds to simple coordination, as in sentence (15a), while the second is a mathematical idealization of the *respectively* construction, as in sentence (15b):

(15) a. Jean^{a} seems German^{b} but he is French^{c}, Pietro^{a} seems Russian^{b} but he is Italian^{c} and Peter^{a} seems Belgian^{b}, but he is English^{c}.

[5] This is the case of (Bresnan et al., 1987) which address the strong generative capacity rather than the weak generative capacity.

[6] A similar, earlier, reasoning gave Shieber the key in order to prove the non-context-freeness of Swiss-German, (Shieber, 1985).

b. Jean^a, Pietro^a and Peter^a seem respectively German^b, Russian^b and Belgian^b, but they are French^c, Italian^c and English^c.

Respectively constructions have already been considered as initial counterexamples against context-freeness of English with sentences as:[7]

(16) a. John, Mary and David are a widower, a widow and a widower respectively.

b. This land and these woods can be expected to rend itself and sell themselves respectively.

However the linguistic community considered such constructions too artificial, the Swiss-German data being more clear.[8]

Remark 4.4. We have chosen these formal languages for historical reasons. They have played an important role in formal languages and linguistics, however we are also interested in these languages here because they provide a glimpse of a deeper internal relation. Given a string $x \in \Sigma^*$ we have the usual exponentiation:

$$x^n = \underbrace{x \cdots x}_{n \text{ times}} .$$

If $x = a_1 \cdots a_m$ with $a_1, \ldots, a_m \in \Sigma$, we define the *exponentiation by letters* as

$$x^{\downarrow n} = a_1^n \ldots a_m^n .$$

Then the languages above can be rewritten and contrasted as follows:

$$L_{\text{squa}} = \{x^{\downarrow 2} \mid x \in \Sigma^*\}, \text{ compare to } L_{\text{copy}} = \{x^2 \mid x \in \Sigma^*\};$$
$$L_{\text{mult}} = \{(abc)^n \mid n \in \mathbb{N}_+\}, \text{ compare to } L_{\text{resp}} = \{(abc)^{\downarrow n} \mid n \in \mathbb{N}_+\}.$$

As we will see, this shifting relation which we will call *symmetry* actually operates in the dependency structure rather than the string structure although the effect is, at least for the above particular cases, mildly visible in the latter.

4.3.2 Some Examples of Manifolds

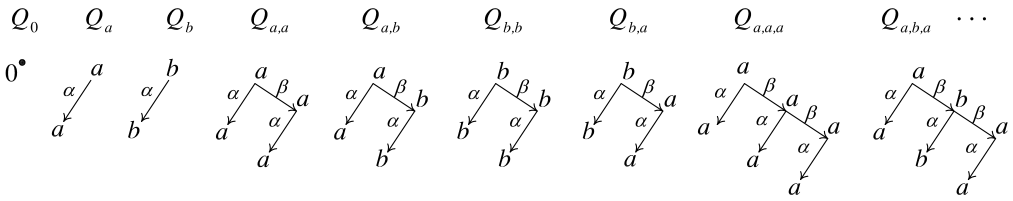
At the moment we cannot show grammars for such languages, however in order to do this in the future we will have to implement adequate manifolds, i.e. a manifold W_{squa} for the squares languages, a manifold W_{copy} for the copy languages, and so forth. The following examples illustrate some general strategies in order to establish a manifold. Let us show how to build the squares manifold.

[7] See (Bar-Hillel and Shamir, 1960) for the first sentence and (Kac et al., 1987) for the second.

[8] See (Pullum and Gazdar, 1982) for a discussion at that time. See later §11.10.

Example 7. The squares manifold W_{squa} and how to build a manifold. We want to achieve a set of syntagmata as in the following Fig. (a) which we denote by $W_{\text{squa}} = \{Q_0, Q_a, Q_b, Q_{a,a}, Q_{a,b}, \ldots\}$:

Fig. (a)



Let us see how to build a manifold capturing them. We assume an alphabet Σ and two syntactic functions, $\zeta = \{\alpha, \beta\}$. Then let there be the following manifold divided in three parts:

$$W_{\text{squa}} = W_{\text{null}} \cap W_{\text{noell}} \cap W_{\text{link}},$$

where:

$$W_{\text{null}} = \textbf{Synt}\begin{pmatrix} \approx 0 & \approx 0 \\ \alpha^2(\zeta)^*_2 & \beta\alpha(\zeta)^*_2 \end{pmatrix} \in \textbf{Man}(\mathbf{k}^2\mathbb{H});$$

$$W_{\text{noell}} = \textbf{Synt}\begin{pmatrix} B & B \\ (1,\alpha)(\zeta)^*_2 & (1,\beta)(\zeta)^*_2 \end{pmatrix} \in \textbf{Man}(\mathbf{k}\mathbb{H});$$

$$W_{\text{link}} = \textbf{Synt}\begin{pmatrix} B' \\ (1,\alpha)(\zeta)^*_2 \end{pmatrix} \in \textbf{Man}(\mathbf{k}^2\mathbb{H}).$$

In general in order to build a manifold it is advisable to first think in terms of the general shape of the trees, and then in terms of the inner correlations. Shaping a tree is similar to a pruning. In order to cut branches we use the valuation, defined by $(\approx 0)(x) = 1 \iff x = 0$, which acts as pruning scissors. The manifold W_{null} says that the loci given by the patterns $\alpha^2(\zeta)^*_2$ and $\beta\alpha(\zeta)^*_2$ must be null. Notice, however, that this does not cut really the branches: underneath there can appear non-null loci and then there would be ellipsis.

To avoid this we can define a manifold which excludes ellipsis. Consider the valuation $B(x, y) = (x \approx 0 \to y \approx 0)$, which is read “if x is null then y must be null". If we apply this valuation to adjacent pairs $(x, \alpha x)$ and $(x, \beta x)$ for any $x \in \zeta$, we obtain the desired effect. This is realized by the manifold W_{noell}, called the *non-ellipticity manifold*. Now if we combine W_{null} and W_{noell} we have real pruning scissors, and the branches cut by the null manifold indeed fall. See below Fig. (b).

Finally, the manifold W_{link} links pairs of loci. The valuation B' is defined by $B'(x, y) = (y \not\approx 0 \to x \approx y)$, in other words if y is not null, x and y must be equal. The pattern is the set $(1,\alpha)(\zeta)^*_2 = \{(x, \alpha x) \mid X \in \zeta^*\}$. So the syntagmata, $S \in W_{\text{link}}$, have to satisfy that $S(x) = S(\alpha x)$ provided $S(\alpha x) \neq 0$. In particular we must have $S(1) = S(\alpha)$, $S(\beta) = S(\alpha\beta)$, $S(\beta^2) = S(\alpha\beta^2)$, …. Graphically this is saying that the vertices connected by the α edge must be equal; see Fig. (c). Notice that given the combination of the above manifolds, for the remaining pairs $(x, \alpha x)$ we have that $S(x) = 0$ and $S(\alpha x) = 0$. Finally we observe that this manifold is in $\mathbf{Man}(\mathbf{k}^2\mathbb{H})$.

Fig. (b)

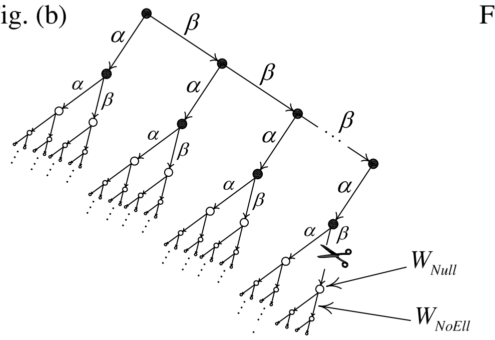
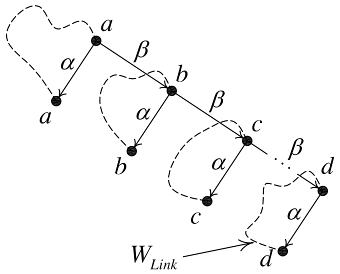


Fig. (c)



The non-ellipticity used in this example, which is captured by the manifold W_{noell}, is a very useful manifold to shape trees, so we are going to fix a notation for it. There are, as we shall see, other ways to prune a tree, however the usage of the non-elliptic manifold will be frequent.[9] In addition we check that the set of atomic syntagmata and the set of non-elliptic syntagmata are manifolds.

Lemma 4.5. *Consider the empty set of syntactic functions $\zeta = \emptyset$; then $\emptyset^* = \{1\}$. The unique pattern that we can consider is the trivial pattern. Consider the valuation $\not\approx 0$, i.e. $(\not\approx 0)(x) = 1$ iff $x \neq 0$. We define:*

$$\mathbf{Atom} = \mathbf{Synt}\begin{pmatrix} \not\approx 0 \\ 1 \end{pmatrix}.$$

Now consider the valuation $B(x, y) = (x \approx 0 \longrightarrow y \approx 0)$. We fix a finite set ζ of syntactic functions and define the following manifolds:

$$\mathbf{Nell} = \bigcap_{\lambda\in\zeta} \mathbf{Synt}\begin{pmatrix} B \\ (1,\lambda)(\zeta)^*_2 \end{pmatrix}; \quad \mathbf{Nell}^{\mathfrak{R}} = \bigcap_{\lambda\in\zeta} \mathbf{Synt}\begin{pmatrix} B \\ (\zeta)^*_2(1,\lambda) \end{pmatrix}.$$

Then **Atom** *comprises all the atomic syntagmata,* **Nell** *comprises all the syntagmata without ellipsis and* $\mathbf{Nell}^{\mathfrak{R}}$ *comprises all the syntagmata without co-ellipsis.*

Proof. Consider the first manifold **Atom**, where $\zeta = \emptyset$ which means that syntagmata S in $\mathbf{Synt}_{\zeta,\Sigma}$ only have one locus, namely 1, thus they are all atomic. If we consider the constant valuation 1 (i.e. $1(x) = 1$, for any $x \in \Sigma_+$) then $S(1)$ can take any letter

[9] In (Cardó, 2016) all the syntagmata considered were non-elliptic, so we did not need include this manifold. However, here we prefer to adopt the general case and make non-ellipticity explicit.

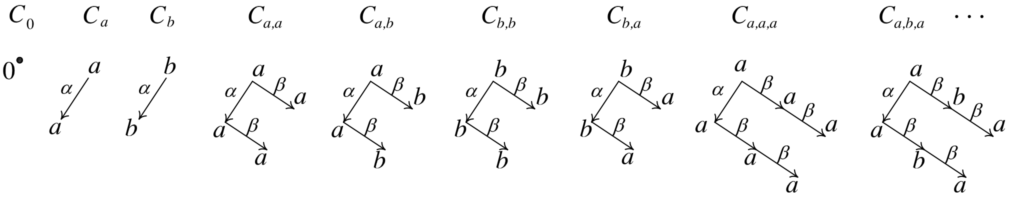
in Σ, so **Atom** contains all the atoms. Regarding **Nell** the pattern can be rewritten as $(1,\lambda)(\zeta)^*_2 = \{(\varphi, \lambda\varphi) \mid \varphi \in \zeta^*)\}$ for any λ. If we apply the valuation we have that

$$S \in \textbf{Nell} \iff (S(\varphi) = 0 \to S(\lambda\varphi) = 0)\ \forall\lambda \in \zeta, \forall\varphi \in \zeta^*.$$

In other words if a locus φ is null then its child $\lambda\varphi$ is null. Since this is the case for any locus and for any λ, when a locus is null all the loci under it are null. So ellipsis is precluded. The case of $\textbf{Nell}^{\mathfrak{R}}$ is similar and, by the same token, precludes co-ellipsis. □

Example 8. Another three manifolds: W_{copy}, W_{mult} and W_{resp}. Consider now the manifold comprising syntagama as in Fig. (a) $W_{\text{copy}} = \{C_0, C_a, C_b, C_{a,a}, C_{a,b}, \ldots\}$:

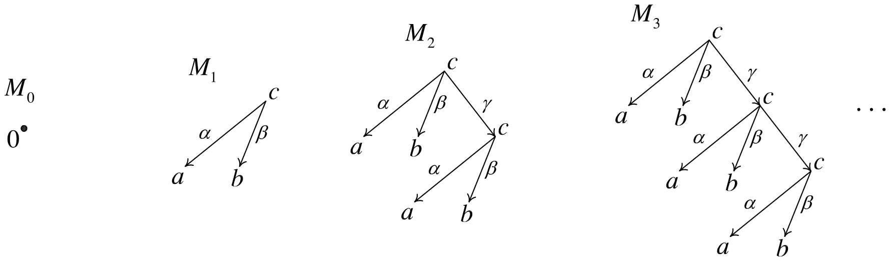
Fig. (a)



$$W_{\text{copy}} = \textbf{Nell}^{\mathfrak{R}} \cap \textbf{Synt}\begin{pmatrix}\approx 0 & \approx 0\\ \zeta^*\alpha^2 & \zeta^*\alpha\beta\end{pmatrix} \cap \textbf{Synt}\begin{pmatrix} y \not\approx 0 \to x \approx y \\ (\zeta)^*_2(1,\alpha)\end{pmatrix},$$

where we are supposing an alphabet Σ and two syntactic functions, $\zeta = \{\alpha, \beta\}$. The understanding of this manifold is as in the previous example. Another example is displayed in Fig. (b) comprising syntagmata $W_{\text{mult}} = \{M_0, M_1, M_2, \ldots\}$:

Fig. (b)

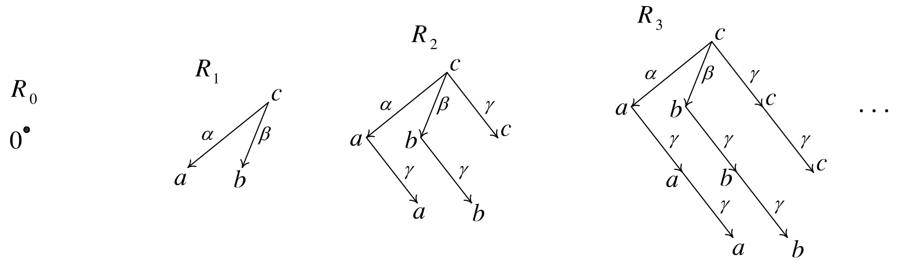


These are captured by the following construction. Let there be the sets $\zeta = \{\alpha, \beta, \gamma\}$, $\Sigma = \{a, b, c\}$, and let there be the manifold in $\textbf{Man}(\textbf{k}\mathbb{H})$: $W_{\text{mult}} = \textbf{Nell} \cap$

$$\cap\ \textbf{Synt}\begin{pmatrix} x \approx c \to (y \approx a \wedge z \approx b \wedge t \in \{c, 0\}) & x \in \{a, b\} \to (y \approx 0 \wedge z \approx 0 \wedge t \approx 0) \\ (1, \alpha, \beta, \gamma)(\zeta)^*_4 & (1, \alpha, \beta, \gamma)(\zeta)^*_4 \end{pmatrix}.$$

Finally, the set of syntagmata $W_{\text{resp}} = \{R_0, R_1, R_2, \ldots\}$ (Fig. (c)):

Fig. (c)



This is obtained by the manifold in **Man**($\mathbb{H}$**k**): W_{mult} = **Nell** ∩

$$\cap\ \mathbf{Synt}\begin{pmatrix} x \approx c \longrightarrow (y \approx a \wedge z \approx b \wedge t \in \{c, 0\}) & x \in \{a, b\} \longrightarrow (y \approx 0 \wedge z \approx 0 \wedge t \approx 0) \\ (\zeta)^*_4(1, \alpha, \beta, \gamma) & (\zeta)^*_4(1, \alpha, \beta, \gamma) \end{pmatrix}.$$

4.3.3 How to Build Manifolds for Natural Languages

Thanks to the boolean nature of valuations we are able to implement a wide variety of rules. If we want to describe a natural language we can build a number of manifolds capturing different phenomena. We can begin with a manifold to describe for each word what kind of words or categories it can govern. For example, a noun can govern a determiner and an adjective, but not an adverb or verb, as in Example 5.

With another manifold we can define for each function what functions can follow, such as the rule for transitive verbs in Example 6. Using the Tesnèrian denomination, this amounts to defining *valence*. We can define still another manifold to describe morphological agreements, and so forth.

Some constraints must be imposed only under certain conditions. For example, a subject and an attribute must agree only when the attribute is not null, that is, when the attribute really appears in the structure.

Next we have to think where these rules must hold, i.e. we must think of the patterns. For all the commented constraints the pattern will have the shape **k**$\mathbb{H}$.

Fortunately in order to add new phenomena, like *pied-piping* or others, we can build other manifolds and intersect them with the existing ones. We will implement some fragments of natural language in chapters 10 and 11. For the moment we see an example which captures a fragment of dependency trees of English.

Example 9. A <span>TOY MANIFOLD FOR ENGLISH</span>. We consider the following set of syntactic functions $\zeta = \{Sb, Ob, Dt, Ad, Md\}$ and the vocabulary $\Sigma = \{$*catch, catches, caught, smile, smiles, smiled, John, Mary, boy, boys, frog, frogs, a, the, this, these, that, those, big, small, very*$\}$. We consider the following subsets of Σ, or *lexical categories*:

Verb = {*catch, catches, caught, smile, smiles, smiled*},
TransitiveVerb = {*catch, catches, caught*},
IntransitiveVerb = {*smile, smiles, smiled*},
Noun = {*John, Mary, boy, boys, frog, frogs*},
Determiner = {*a, the, this, these, that, those*},
Adjective = {*big, small*},
Modifier = {*very*},
Singular = {*a, the, this, that, John, Mary, boy, frog, catches, caught, smiles, smiled*},
Plural = {*the, these, those, boys, frogs, catch, caught, smile, smiled*}.

Given a lexical category $\Sigma' \subseteq \Sigma$, we notate $\Sigma'_+ = \Sigma' \cup \{0\}$. We define parametrically the manifolds:

$$\textsc{govern}(\Sigma_0, \Sigma_1, \zeta_0) = \bigcap_{\lambda \in \zeta_0} \mathbf{Synt}\begin{pmatrix} x \in \Sigma_0 \longrightarrow y \in \Sigma_1 \\ (1, \lambda)(\zeta)^*_2 \end{pmatrix},$$

and

$$\textsc{Rgovern}(\Sigma_0, \Sigma_1, \zeta_0) = \bigcap_{\lambda \in \zeta_0} \mathbf{Synt}\begin{pmatrix} x \in \Sigma_0 \longleftarrow y \in \Sigma_1 \\ (1, \lambda)(\zeta)^*_2 \end{pmatrix}.$$

We also need the manifold **Nell** and the manifold $\textsc{root} = \mathbf{Synt}\binom{x \in \mathbf{Verb}}{1}$. Then we consider the intersection of the following manifolds, which we notate W_{Eng}:

(a.1) **Nell**,
(a.2) $\textsc{root}$,

(b.1) $\textsc{govern}(\mathbf{Verb}, \{0\}, \{Dt, Ad, Md\})$,
(b.2) $\textsc{govern}(\mathbf{Noun}, \{0\}, \{Sb, Ob, Md\})$,
(b.3) $\textsc{govern}(\mathbf{Determiner}, \{0\}, \{Sb, Ob, Dt, Ad, Md\})$,
(b.4) $\textsc{govern}(\mathbf{Adjective}, \{0\}, \{Sb, Ob, Ad\})$,
(b.5) $\textsc{govern}(\mathbf{Modifier}, \{0\}, \{Sb, Ob, Dt, Ad\})$,

(c.1) $\textsc{govern}(\mathbf{Verb}, \mathbf{Noun}_+, \{Sb, Ob\})$,
(c.2) $\textsc{govern}(\mathbf{Noun}, \mathbf{Adjective}_+, \{Ad\})$,
(c.3) $\textsc{govern}(\mathbf{Noun}, \mathbf{Determiner}_+, \{Dt\})$,

(c.4) govern(**Adjective**, **Modifier**$_+$, $\{Md\}$),

(c.5) govern(**Modifier**, **Modifier**$_+$, $\{Md\}$),

(d.1) govern(**Verb**, Σ, $\{Sb\}$),

(d.2) govern(**TransitiveVerb**, Σ, $\{Ob\}$),

(d.3) govern(**IntransitiveVerb**, $\{0\}$, $\{Ob\}$),

(e.1) Rgovern(**Singular**, **Singular**, $\{Sb, Dt\}$),

(e.2) Rgovern(**Plural**, **Plural**, $\{Sb, Dt\}$),

(e.1) govern(**Singular**, Σ, $\{Dt\}$).

Let us comment on the meaning of some of these manifolds. The manifold (a.1) precludes ellipses. (a.2) says that the root of any syntagma is a (finite tense) verb. Manifolds (b.1)-(b.6) forbid arguments according to the lexical category; thus, (b.1) says that verbs cannot govern determiners, adjectives, nor modifiers. Complementarily, manifolds (c.1)-(c.5) establish which lexical categories are governed by each function. For instance, by (c.1), a verb always takes a noun as subject and object (which may be null). Manifolds (d.1)-(d.3) establish obligation or not of the verbal arguments. In particular (d.1) says that the subject is never empty. By (d.2), transitive verbs cannot exhibit an empty object, while by (d.2), intransitive verbs must exhibit an empty object. The manifolds (e.1) and (e.2) attend number agreements. For example, (e.1) says that if the subject is singular, then the verb which governs the subject is also singular, and likewise for the determiner. Notice that in these cases, the implication of the conditional in the valuation is reversed, i.e. the number of the governed implies the number of the governor. For example *caught* is a plural and singular word simultaneously which cannot decide which is the number of the subject; however if the subject is *John* then the verb must be singular. Finally (e.3) says that singular (count) nouns always needs a determiner (*the boy caught the frog*/ **boy caught frog*).

The intersection of all these manifolds yields a manifold which is infinite; this allows syntagmata for sentences such as *the boy caught a very very very (...) very big frog*. For instance Fig.(a) shows a syntagma in W_{Eng}, while the syntagma in Fig.(b) violates the rules in the manifolds (c.1) and (e.1).

Fig. (a)

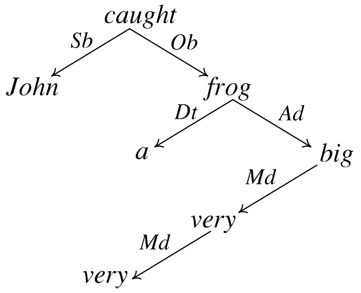
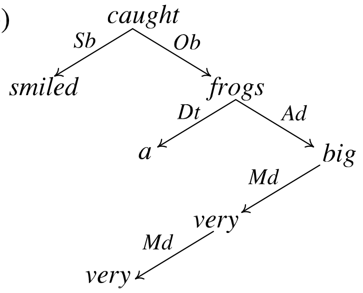


Fig. (b)



4.4 Tree formalisms and the Class $\mathbf{Man}(\mathbf{k}^p\mathbb{H})$

4.4.1 Tree Grammars

There arises the issue of comparing manifolds with other formalisms establishing sets of arboreal structures such as *logic over trees* or *tree grammars*. Let us consider in this subsection the case of tree grammars.

Some frameworks use the notation of *terms* of a free algebra to encode trees.[10] More formally, the set of *trees over* Σ, denoted by T_Σ, is the language over the alphabet $\Sigma \cup \{[,]\}$ defined inductively as follows. If $a \in \Sigma_0$, then $a \in T_\Sigma$. For $k \geq 1$, if $a \in \Sigma_k$ and $t_1, \ldots, t_k \in T_\Sigma$, then $a[t_1 \ldots t_k] \in T_\Sigma$. Thus, $a[t_1 \ldots t_k]$ encodes a tree with root labeled by a and subtrees hanging from the root, $t_1, \ldots, t_k$. A tree language is a subset of T_Σ.

Several notions from languages are transported into tree languages such as automata, regular grammars or local languages. For example, the regular tree grammar with the rules $S \longrightarrow p[aTa]$, $T \longrightarrow q[cp[dT]b]$ and $T \longrightarrow e$ (i.e. rules consisting in rewriting the non-terminal symbols S, T) generates the tree given by the derivation:

$$S \Rightarrow p[aTa] \Rightarrow p[aq[cp[dT]b]a] \Rightarrow p[aq[cp[de]b]a].$$

For every tree we can consider the *yield* consisting in the letters of the leaves taken in the same order as in the tree expression: $\text{Yield}(p[aq[cp[de]b]a]) = acdeba$. Given a tree language L, $\text{Yield}(L)$ is the set of yields of each tree. Then it is proved that:

Theorem 4.6. *The following statements hold.*

(i) Let G be a context-free string grammar; then the set of derivation trees of $L(G)$ is a regular tree language.

(ii) Let L be a regular tree language; then $\text{Yield}(L)$ *is a context-free string language.*

(iii) There exists a regular tree language which is not the set of derivation trees of any context-free language.

Proof. See (Comon et al., 2007). □

Thus, by (i) and (ii), context-free (string) languages can be regarded as the set of (regular) syntactic structures conveniently linearized by the yield function. This connects with our precept of separating the syntactic structure and the word-order. Since context-free constructions are a fundamental part of language, syntactic structures of a formalism should exhibit an expressive power no greater than regular trees, (at least in order to explain the context-free part). Actually, in some formalisms regularity of trees is assumed fully, whereby non-context-free constructions must be achieved in the linearization process. Mildly context-sensitive formalisms exhibit regular structures.

[10] We follow definitions and notations from (Engelfriet, 2015; Comon et al., 2007).

Dependency languages that can be generated by TAG or even more generally, by LCRFS, are regular in the sense of Kuhlmann and Möhl (2007). Likewise the MTS (Model Theoretic Syntax) advocated by Rogers and Pullum, although in the logic framework, uses by definition regular structures.[11]

It is indeed our conviction also that syntactic structures should approximate regularity. We will resume this issue in the final chapter in the light of several linguistic examples. For the moment let us consider the following geometrical interpretation of the little fragment of English structures in Example 9. The simple manifolds used in order to define W_{Eng} only contain patterns in $\mathbf{k}\mathbb{H}$. The effect of these patterns over the constraint B consists in sliding B throughout the tree. When we multiply a constant φ by a homogeneous monoid $(\zeta)^*_a$, the positions in the components of φ are translated simultaneously. Since these components fit in a triangle (a node with its immediate descendants), one can check if a syntagma is in the manifold by sliding a triangular window and checking whether every triangle is in a sample of triangles given in advance; this sample is given by the constraint B. The following subsections prove this formally.

This idea is in fact the notion of *local testability*. In the framework of languages, it is said that a language is strictly locally k-testable iff by sliding an interval of length k over the word we can decide if the word is in the language or not.[12]

The notion of locality can be elevated to the level of trees. A tree language is said to be *k-locally testable* when it can be determined by three sets of subtrees of depth k: top, triangles and leaves.[13] A tree t is in the local grammar iff the top of t is in the top set, every subtree is in the triangles set and leaves of t are in the leaves sample.[14]

It is the case that local tree languages are regular tree languages, thus the manifold W_{Eng} in Example 9 satisfies the requirement, above commented, that the sets of syntactic structures should not be too expressive. We will see that in fact these kinds of manifolds can also be used to generate non-context-free constructions. But this will be addressed when we introduce linearizations.

Despite the simplicity of regular grammars, not every regular tree language is learnable in the sense of identification in the limit, Gold (1967). However local testable tree languages are, (Knuutila, 1993; Rico-Juan et al., 2005). This could be a good starting point in order to study machine-learnability questions of our model.

[11] Thanks to an anonymous reviewer for references.

[12] For example $\{a, b\}^*$ is 1-strictly locally testable, and $a(baa)^+$ is 3-strictly locally testable, while $(aa)^*$ is not strictly locally testable for any k. Although these languages are subregular, and hence nor sufficiently expressive for natural languages, they enjoy applications to linguistic fields beyond syntax, such as morphology and phonology, cf. (Bordihn et al., 2009; Rogers and Hauser, 2010; Rogers et al., 2013; Edlefsen et al., 2008; Heinz et al., 2011).

[13] Leaves are rather a vector of possible leaves, not a subtree.

[14] In the original definition from Knuutila (1993) these subtrees of depth k are called *k-forks*.

4.4.2 Locality and the Class **Man**($\mathbf{k}^p\mathbb{H}$)

We prove the correspondence between manifolds in **Man**($\mathbf{k}^p\mathbb{H}$) and local testability. Notice, however, that trees as defined above are not exactly syntagmata: in syntagmata the arrows are labeled with syntactic functions, nodes are not ordered and, in addition, ellipses are allowed. It is convenient to translate the notions of locally testability to the field of syntagmata.

A peculiarity of syntagmata, in relation to definition of locality, is that it is not necessary to specify separately conditions over leaves. These conditions can be made by the set of inner triangles, because we can use triangles with null loci to scan the bottom parts of the syntagma in hand.

The Class **Man**($\mathbf{k}^p$)

Given a syntagma S, we define the *p-top of S S'* as the result of trimming branches deeper than p:

$$S'(x) = \begin{cases} S(x) & \text{if } |x| \leq p; \\ 0 & \text{otherwise.} \end{cases}$$

Definition 4.7. We say that a set of syntagmata W is a *p-top set* iff there is a finite set of syntagmata with depth $\leq p$, $\mathcal{U}$, such that $S \in W \iff$ the p-top of S is in $\mathcal{U}$. Then we notate $\mathrm{Top}_p[\mathcal{U}] = W$. We notate $\mathbf{TOP}_p$ the class of p-top sets.

Theorem 4.8. **Man**($\mathbf{k}^p$) = $\mathbf{TOP}_p$.

Proof. We assume syntagmata over Σ and ζ, with $|\zeta| = m$. We set $q = \frac{m^{p+1}-1}{m-1}$. ($\supseteq$) Let $\mathrm{Top}_p[\mathcal{U}]$ be a p-top set. We encode a syntagma of depth p by the vector in Σ_+^q where the first component is the root, the following m components encode the children of the root, the following m^2 components encode the grand-children, and so forth. Thus we can imagine the set $\mathcal{U}$ as a subset of Σ_+^q, notating this last subset as $\overline{\mathcal{U}}$. Then we define the manifold:

$$W = \mathbf{Synt}\begin{pmatrix}(x_1, \ldots, x_q) \in \overline{\mathcal{U}} \\ \varphi\end{pmatrix},$$

where $\varphi = (1, \underbrace{\lambda_1, \ldots, \lambda_m}_{\text{children}}, \underbrace{\lambda_1^2, \lambda_2\lambda_1, \lambda_3\lambda_1, \ldots, \lambda_m^2}_{\text{grandchildren}}, \;\ldots\ldots,\; \underbrace{\lambda_1^p, \ldots, \lambda_m^p}_{p\text{-grandchildren}}).$

Thus, $W = \mathrm{Top}_p[\mathcal{U}] \in$ **Man**($\mathbf{k}^p$). Let us see the other direction ($\subseteq$). Consider $W =$ **Synt** $\binom{B}{\varphi}$ a simple manifold with $\varphi \in \mathbf{k}^p$. We observe that the constraint given by the valuation B only affects those loci $x \in \zeta^*$ such that $|x| \leq p$, in other words we can only control the top of the tree, while the rest of the syntagma can take any letter. Since there are a finite number of loci such $|x| \leq p$, the constraint is defining a finite number of combinations of letters on the top. So there is a set $\mathcal{U}$ such that $S \in W \iff$ the p-top of S is in $\mathcal{U}$. Now we have to consider non-simple manifolds. The class $\mathbf{TOP}_p$ is closed by unions and intersections: $\mathrm{Top}_p[\mathcal{U}] \cup \mathrm{Top}_p[\mathcal{U}'] = \mathrm{Top}_p[\mathcal{U} \cup \mathcal{U}']$, $\mathrm{Top}_p[\mathcal{U}] \cap \mathrm{Top}_p[\mathcal{U}'] = \mathrm{Top}_p[\mathcal{U} \cap \mathcal{U}']$. So we are done. □

The Class $\mathbf{Man}(\mathbf{k}^p\mathbb{H})$

We call the *triangle of depth p and root* φ of S, or simply *p-triangle*, the syntagma S':

$$S'(x) = \begin{cases} S(x) & \text{if } x \in \zeta^*\varphi \text{ and } |x| \le p + |\varphi|; \\ 0 & \text{otherwise.} \end{cases}$$

I.e. geometrically S' is the subtree of S with depth at most p and root at φ.

Definition 4.9. We say that a set of syntagmata W is *strictly strongly p-locally testable* iff there is a finite set of syntagmata $\mathcal{V}$ with depth $\le p$, such that $S \in W$ iff all the p-triangles of S are in $\mathcal{V}$. We notate $\mathrm{Tri}_p[\mathcal{V}] = W$.

We say that a set of syntagmata is *lattice p-locally testable* iff it is the $\cup\cap$-closure of p-top and strictly strongly p-locally testable sets. We notate $\mathfrak{L}\mathbf{LOCAL}_p$ the class of these sets.

Theorem 4.10. $\mathbf{Man}(\mathbf{k}^p\mathbb{H}) = \mathfrak{L}\mathbf{LOCAL}_p$, *for any* $p \ge 0$.

Proof. We assume syntagmata over Σ and ζ, with $|\zeta| = m$. We set $q = \frac{m^{p+1}-1}{m-1}$. Let us see the inclusion ($\supseteq$). We distinguish two cases. First, let $\mathrm{Top}_p[\mathcal{U}]$ be a p-top set; then by the previous Theorem 4.8, $\mathrm{Top}_p[\mathcal{U}] \in \mathbf{Man}(\mathbf{k}^p) \subseteq \mathbf{Man}(\mathbf{k}^p\mathbb{H})$. Second, let $\mathrm{Tri}_p[\mathcal{V}]$ be a strictly strongly p-locally testable. As in the proof of the previous theorem we encode a syntagma of depth p by the vector in Σ_+^q, being $\overline{\mathcal{V}} \subseteq \Sigma_+^q$ the encoding of the set $\mathcal{V}$. Now we define the manifold W:

$$W = \mathbf{Synt}\begin{pmatrix} (x_1, \dots, x_q) \in \overline{\mathcal{V}} \\ \varphi(\zeta)_q^* \end{pmatrix},$$

where $\varphi = (1, \underbrace{\lambda_1, \dots, \lambda_m}_{\text{children}}, \underbrace{\lambda_1^2, \lambda_2\lambda_1, \lambda_3\lambda_1, \dots, \lambda_m^2}_{\text{grandchildren}}, \;\dots\dots,\; \underbrace{\lambda_1^p, \dots, \lambda_m^p}_{p\text{-grandchildren}}).$

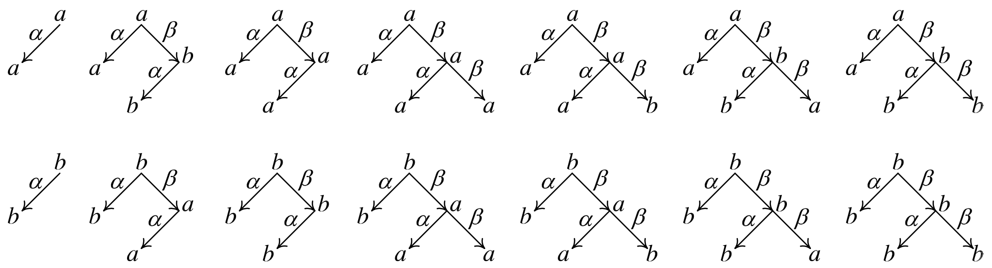
We have that $W = \mathrm{Tri}_p[\mathcal{V}]$ and that $W \in \mathbf{Man}(\mathbf{k}^p\mathbb{H})$. Since a set in $\mathfrak{L}\mathbf{LOCAL}_p$ is a $\cup\cap$-combination of top sets and strictly strongly locally testable sets, and since $\mathbf{Man}(\mathbf{k}^p\mathbb{H})$ is $\cup\cap$-closed, $\mathbf{Man}(\mathbf{k}^p\mathbb{H}) \supseteq \mathfrak{L}\mathbf{LOCAL}_p$.

Let us see the inclusion ($\subseteq$). Consider $W = \mathbf{Synt}\binom{B}{\Gamma}$ a simple manifold which has a pattern $\Gamma \in \mathbf{k}^p\mathbb{H}$. So either $\Gamma = \varphi$ or $\Gamma = \varphi(\zeta)_a^*$, where $\varphi \in \zeta^{a*}$ and $|\varphi| \le p$. In the first case we saw in the previous theorem that this manifold defines a p-top language. So we consider $\Gamma = \varphi(\zeta)^*_a$. We notice that:

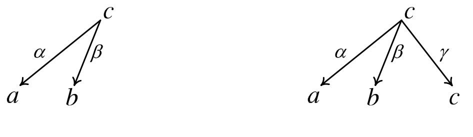
$$\mathbf{Synt}\begin{pmatrix} B \\ \varphi(\zeta)^*_a \end{pmatrix} \subseteq \mathbf{Synt}\begin{pmatrix} B \\ \varphi \end{pmatrix}.$$

The effect of this pattern consists in sliding the constraint defined by B and φ throughout the syntagma. As in the above proof B and φ are defining a finite set of syntagmata of depth $\leq p$, say $\mathcal{V}$, such that $W = \mathrm{Tri}_p[\mathcal{V}]$. So W is a strictly strongly p-locally testable set. Since the class $\mathfrak{L}\mathbf{LOCAL}_p$ is the $\cup\cap$-closure of strictly strongly p-local sets and p-top sets, any non-simple manifold in $\mathbf{Man}(\mathbf{k}^p\mathbb{H})$ is also in $\mathfrak{L}\mathbf{LOCAL}_p$. □

Example 10. Some manifolds as local sets. The squares manifold uses patterns with types in $\mathbf{k}^2\mathbb{H}$, or in other words $W_{\mathrm{squa}} \in \mathbf{Man}(\mathbf{k}^2\mathbb{H})$, and is therefore lattice local by the above theorem. We have that $W_{\mathrm{squa}} = \mathrm{Top}_2[\mathcal{U}] \cap \mathrm{Tri}_2[\mathcal{V}]$, where $\mathcal{U} = \{0^\bullet\} \cup \mathcal{W}$, $\mathcal{V} = \{0^\bullet, a^\bullet, b^\bullet\} \cup \mathcal{W}$ and where $\mathcal{W}$ is the set of syntagmata:



The manifold $W_{\mathrm{mult}} \in \mathbf{Man}(\mathbf{k}\mathbb{H})$ is also lattice local and $W_{\mathrm{mult}} = \mathrm{Top}_1[\mathcal{U}] \cap \mathrm{Tri}_1[\mathcal{V}]$, where $\mathcal{U} = \{0^\bullet\} \cup \mathcal{W}$, $\mathcal{V} = \{0^\bullet, a^\bullet, b^\bullet\} \cup \mathcal{W}$ and where $\mathcal{W}$ is the set of syntagmata:



4.4.3 Logic over Trees

Another family of tree formalisms which can be compared with manifolds are those based on tree logic, (Rogers and Nordlinger, 1998). Several logical languages have been proposed to encode and define trees. Such languages have variable symbols for the vertices of a tree (or, more in general, graphs), some usual logical constants ($\forall, \exists, \neg, \wedge, \vee, \longrightarrow$) and some basic symbols for the relations over vertices such as "successor" or "sister" relations. This conforms to standard first order logic (FOL). In monadic second order logic (MSOL), quantification over sets of vertices is allowed, which proportions an extra expressive power. In this framework a set of finite trees is definable in MSOL iff it is recognizable by a finite tree automaton, (Thomas, 1997). Other more specific logics as wMSO, $L^2_{K,P}$ or SnS formalize *ad hoc* linguistic theoretic concepts; see (Rogers and Nordlinger, 1998; Rogers, 2003, 1996). These approaches are located in the framework of MTS (Model Theoretic Syntax). Linguistic structures are the set objects which satisfy logical formulas, i.e. the set of models in the logical sense, see (Pullum and Scholz, 2001; Pullum, 2007). In this respect consider the definition of simple manifold. The condition for a syntagma S to be in the manifold **Synt** $\binom{B}{\Gamma}$ is that S satisfies:

$$B(S(x_1), \dots, S(x_a)) = 1 \text{ for all } (x_1, \dots, x_n) \in \Gamma.$$

So our approach through manifolds is similar to MTS in the sense that licensed structures are models of a logical formula, but we concentrate the descriptive power in the exact form of the domains of quantification which we constraint to be patterns.

If we restrict ourself to FOL, it is provable that a first-order definable set of graphs of bounded degree is locally threshold testable, (Thomas, 1997).[15] In simpler words "first order logic is local". Here we show that manifolds in **Man**($\mathbf{k}^p\mathbb{H}$) are definable in a fragment of FOL (and viceversa), and recall that these manifolds are local sets of syntagmata in the sense of Theorem 4.10.[16]

The condition for a syntagma S to be in the manifold **Synt** $\binom{B}{\Gamma}$ can represented in prenex form:

$$\forall x_1 \forall x_2 \dots \forall x_a \Big(P_\Gamma(x_1, \dots, x_a) \to Q_B(S(x_1), \dots, S(x_a)) \Big),$$

where P_Γ and Q_B are predicates interpretated as that $P_\Gamma(x_1, \dots, x_a)$ is true iff $(x_1, \dots, x_a) \in \Gamma$ and $Q_B(y_1, \dots, y_a)$ is true iff $(y_1, \dots, y_a) \in B^{-1}(1)$.

Expressing Q_B in FOL is not a problem. Recall that B is a finite mapping $B : \Sigma_+^k \longrightarrow \{0, 1\}$ or, what is the same, it is given by the characteristic mapping of the set B^{-1}. Then we can encode valuations as:

$$Q_B(x_1, \dots, x_k) = \bigvee_{(a_1,\dots,a_k) \in B^{-1}(1)} \bigwedge_{i=1}^{k} \approx_{a_i}(x_i),$$

where $\approx_a(x)$ is a unary predicate meaning $x = a$.

Predicates P_Γ are a bit more difficult to translate into a logical formula because of the rich variety of patterns Γ. However when the pattern is of the form $\mathbf{k}\mathbb{H}$, a certain set of formulas of the FOL characterize the class **Man**($\mathbf{k}\mathbb{H}$).

[15] Local threshold testability is a concept a bit more general than locality. Structures are recognized by means of a sliding window in which in addition one has to take care of number of occurrences of certain segments; Ruiz et al. (1998).

[16] In order to define manifolds with more complex patterns probably it is necessary to move to MSOL, but we are not going to deal with this in this thesis.

Let us see this informally. Consider the fragment of FOL language with symbols:

$$\forall, \wedge, \vee, x, \mathbf{S}, \approx_a, \mathbf{Ch}_\alpha, \mathbf{r},$$

where $\forall, \wedge, \vee$ are the usual logical constants; x is a variable x (for vertices); $\mathbf{S}$ is a unary functional symbol (meaning the letter labels on vertices); $\approx_a$, with $a \in \Sigma$, are unary predicates; $\mathbf{Ch}_\alpha$, with $\alpha \in \zeta$, are unary functional symbols (meaning $\mathbf{Ch}_\alpha(x)$ is the child of x by the syntactic function α); and $\mathbf{r}$ is a constant (meaning the root), and where Σ and ζ are any finite sets.

As semantics for this language we take *syntagma structures* $\mathcal{S} = (S, \Sigma, \{f_\alpha\}_{\alpha\in\zeta}, 1)$ where $S : \zeta^* \longrightarrow \Sigma$ is a syntagma; Σ is the set of letters; $f_\alpha : \zeta^* \longrightarrow \zeta^*$ are the functions defined $f_\alpha(x) = \alpha x$; and 1 is the identity of ζ^*.

A syntagma structure $\mathcal{S}$ *models the formula* q in the above language, $\mathcal{S} \models q$, iff q is true for the symbol $\mathbf{S}$ interpreted as the mapping S; $\approx_a(x)$ interpreted as $x = a$; $\mathbf{Ch}_\alpha$ interpreted f_α; and $\mathbf{r}$ interpreted as the identity 1.

Then, if $\varphi = (1, \alpha_1, \ldots, \alpha_k)$ with $\alpha_i \in \zeta, 1 \leq i \leq k$, formulas such as:

$$\forall x\, Q_B\Big(x, \mathbf{S}(\mathbf{Ch}_{\alpha_1}(x)), \ldots, \mathbf{S}(\mathbf{Ch}_{\alpha_k}(x))\Big),$$

are modeled by the same set of syntagmata as the simple manifold $\mathbf{Synt}\binom{B}{\varphi(\zeta)^*_k}$, while formulas such as:

$$Q_B\Big(\mathbf{S}(\mathbf{r}), \mathbf{S}(\mathbf{Ch}_{\alpha_1}(\mathbf{r})), \ldots, \mathbf{S}(\mathbf{Ch}_{\alpha_k}(\mathbf{r}))\Big),$$

are modeled by the same set of syntagmata as the simple manifold $\mathbf{Synt}\binom{B}{\varphi}$. Therefore, the fragment of FOL consisting in $\vee\wedge$-combinations of the above formulas defines any manifold in $\mathbf{Man}(\mathbf{k}\mathbb{H})$; in other words, every set of syntagmata which satisfies a logical formula as above is a manifold with type $\mathbf{k}\mathbb{H}$. And conversely, any manifold is definable by these formulas. A simple generalization allowing the p-th composition $\mathbf{Ch}_{\alpha_1}(\mathbf{Ch}_{\alpha_2}(\cdots(\mathbf{Ch}_{\alpha_p}(x))\cdots)$ characterizes the classes $\mathbf{Man}(\mathbf{k}^p\mathbb{H})$.

4.5 Ellipticity of Manifolds

Definition 4.11. We call the *ellipticity* of S the number of ellipsis of S, $|\mathrm{Ell}(S)|$. Given a manifold W we say that it *has effectively bounded ellipticity* iff there is a computable function $\varepsilon : \mathbb{N}_+ \longrightarrow \mathbb{N}_+$ such that $|\mathrm{Ell}(S)| \leq \varepsilon(|S|)$ for every $S \in W$.

Example 11. Manifolds with and without effectively bounded ellipticity. The manifold **Nell** trivially has effectively bounded ellipticity, $|\mathrm{Ell}(S)| = 0, \forall S \in W$. Since $W_{\text{squa}}, W_{\text{mult}} \subset$ **Nell** they also have effectively bounded ellipticity. Indirectly one can check that $W_{\text{copy}}, W_{\text{resp}}$ are not elliptic.

No manifold in $W \in \mathbf{Man}(\mathbf{k}^p)$, with $\zeta \neq \emptyset$, has effectively bounded ellipticity, because given an integer n there are infinite syntagmata $S \in W$ such that $|S| = n$, which means that the number of ellipses is not bounded. More specifically, consider a syntagma $S \in W \in \mathbf{Man}(\mathbf{k}^p)$, and let $x \in \zeta^*$ be such that $|x| = \text{depth}(S)$. From depth $p+1$ the valuation cannot control loci, whereby we can add a chain to the locus x with k ellipses (as many as we want) and a leaf at the end. Notating S' this new syntagma we have that $S' \in W$ and that $|\text{Ell}(S)| \geq k$.

Ellipticity in manifolds of the kind $\mathbf{k}^p\mathbb{H}$ depend on the specific cases. One can define a manifold in $\mathbf{Man}(\mathbf{k}^2\mathbb{H})$ in which, although it tolerates ellipses, $|\text{Ell}(S)| \leq |S|$. We only need a rule saying *if a locus and all its children are null, then all the grandchildren are null*. When this happens these loci becomes definitively null, but notice that this permits the existence of chains of ellipses as long as we want provided they govern at least one non-null locus. This seems to be the situation in natural languages, as in the following example.

Example 12. Bounded ellipticity in natural languages. In natural languages generally gaps in a sentence are suggested by some surrounding lexical pieces such as some complements of the gapped element. Consider for example the sentences in Catalan (analysis in Fig (a) and (b) below):

(17) a. La noia del tercer pis ha marxat
The girl of-the third floor has moved
'The girl of the third floor has moved.'

b. La ∅ del tercer ∅ ha marxat
The ∅ of-the third ∅ has moved

Fig. (a)

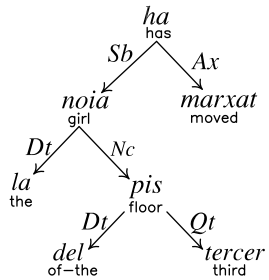
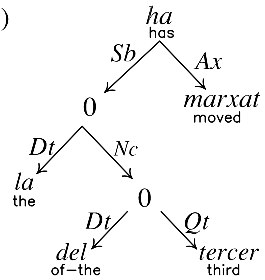


Fig. (b)



The number of ellipses is unbounded in the sense that we can find sentences with as many ellipses as we want: *la ∅(noia) del tercer ∅(pis) del segon ∅(bloc) s'ha mudat* (the ∅(girl) of the third ∅(floor) of the second ∅(building) has moved). However, since in each ellipsis there is a particle which announces it, the number of ellipses of each sentence is bounded by the length of the sentence.

4.6 Decidability of Manifolds

Regarding the decidability of a manifold, we study the question in Appendix C. Amongst other results, we prove that in general manifolds in $\mathbf{Man}(X)$ with a polynomially bounded ellipticity can be decided in polynomial time for any type $X \in \{\mathbf{k}, \mathbb{H}\}^*$. However in particular, given the characterization of manifolds $\mathbf{Man}(\mathbf{k}^p\mathbb{H})$ as lattice local sets (Theorem 4.10), the universal recognition problem for these manifolds becomes linear on the size of the syntagma. Since a manifold is a $\cap\cup$-combination of simple manifolds, the problem is trivially reduced to that of simple manifolds. If we consider a simple manifold $\mathbf{Synt}\binom{B}{\varphi(\zeta)^*_a}$, we just have to build from B and φ the sample of triangles defining the local set. In order to recognize whether a syntagma is in the manifold we check if all the triangles are in the sample.

♣ The above comment should be enough to understand the idea, but the following definition and theorem formalize the proof.

Definition 4.12. The *input format of a simple manifold* $W = \mathbf{Synt}\binom{B}{\Gamma}$ is given by $(B, (\Gamma_1, \ldots, \Gamma_n))$ where $\Gamma_1, \ldots, \Gamma_n$ are the basic types of the decomposition of Γ. We say that a *simple manifold W is decidable* when the problem $S \in^? W$ given a simple manifold W in input format and a syntagma S is decidable.

The *input format for a (not necessarily simple) manifold* is given by a $\cap\cup$-term of simple manifolds. We say that a *manifold W is decidable* when the problem $S \in^? W$ given a manifold W in input format and a syntagma S is decidable.

Theorem 4.13. *For any $p \geq 0$, manifolds in* $\mathbf{Man}(\mathbf{k}^p)$ *are decidable in constant time $O(1)$ in the size of the syntagma, and manifolds in* $\mathbf{Man}(\mathbf{k}^p\mathbb{H})$ *are decidable in linear time $O(|S|)$ in the size of the syntagma S.*

Proof. The time for deciding an intersection/union of manifolds is the sum of the time of deciding the manifolds. So we suppose that the manifold is simple. We suppose $W = \mathbf{Synt}\binom{B}{\varphi} \in \mathbf{Man}(\mathbf{k}^p)$. By Theorem 4.8 W is a p-top set. We only have to transform B and φ into a sample set $\mathcal{U}_{B,\varphi}$, such that $W = \mathrm{Top}_p[\mathcal{U}_{B,\varphi}]$. Notice that $\mathcal{U}_{B,\varphi} = \{S \in \mathbf{Synt} \mid \mathrm{depth}(S) \leq p,\ B(S(\varphi_1), \ldots, S(\varphi_a)) = 1\}$, where $\varphi = (\varphi_1, \ldots, \varphi_a)$ which can be made in constant time $\frac{m^{p+1}-1}{m-1}$, where $m = |\zeta|$. Given $S \in \mathbf{Synt}$ we have that $S \in W$ iff the p-top of S is in $\mathcal{U}_{B,\varphi}$ which can be checked in constant time. Now we consider the case $W = \mathbf{Synt}\binom{B}{\varphi(\zeta)^*_a} \in \mathbf{Man}(\mathbf{k}^p\mathbb{H})$. Again, we construct the set $\mathcal{V}_{B,\varphi}$ such that $W = \mathrm{Tri}_p[\mathcal{V}_{B,\varphi}]$. Notice that $\mathcal{V}_{B,\varphi} = \{S \in \mathbf{Synt} \mid \mathrm{depth}(S) \leq p,\ B(S(\varphi_1), \ldots, S(\varphi_a)) = 1\}$ which is made in constant time. Given $S \in \mathbf{Synt}$ we have that $S \in W$ iff every p-triangle of S is in $\mathcal{V}_{B,\varphi}$. This can be checked in $|S| + 1$ steps because S have $|S| + 1$ triangles ($|S|$ proper triangles and the null syntagma). □

4.7 ♣ A Generalization of Manifolds for Dependency Graphs

Although we are not going to use non-tree-shaped dependency structures in the thesis, we would like to show that the notion of manifold can be extended to more general structures.

Recall that a dependency graph (see §2.5) is a triple $\mathcal{G} = (G, f, g)$ where $G = (V, A)$ is a graph with vertexes V and edges $A \subseteq V \times V$, together with a pair of mappings $f : V \longrightarrow \Sigma$ and $g : A \longrightarrow \zeta$ which label vertices with letters in Σ and edges with syntactic functions in ζ.

A vertex $u \in V$ of a dependency graph is a *generator* iff for all $x \in V$ there is a $\varphi \in \zeta^*$ such that there is the path $u \xrightarrow{\varphi} x$. In particular, when the graph is tree shaped there is a unique generator, the root of the tree.

Definition 4.14. Given a valuation B and a pattern Γ we say that a dependency graph $\mathcal{G} = (G, f, g)$ *satisfies the valuation B throughout the pattern* Γ iff for each generator of G, say u, and for all $(\varphi_1, \ldots, \varphi_a) \in \Gamma$ we have that:

$$u \xrightarrow{\varphi_1} x_1, \ldots, u \xrightarrow{\varphi_1} x_a \implies B(f(x_1), \ldots, B(f(x_a)) = 1,$$

and in the case that some of the vertices x_i do not exist for φ_i we have to substitute the corresponding i-th components by the null word 0, and then it must be satisfied that:

$$B(f(x_1), \ldots, 0, \ldots, 0, \ldots, B(f(x_a)) = 1.$$

Definition 4.15. We notate $\mathbf{GSynt}_{\Sigma,\zeta}$ the set of all dependency graphs. The *simple manifold of dependency graphs* is the set:

$$\mathbf{GSynt}_{\Sigma,\zeta}\binom{B}{\Gamma} = \{\mathcal{G} \in \mathbf{GSynt}_{\Sigma,\zeta} \mid \mathcal{G} \text{ satisfies } B \text{ throughout } \Gamma\}.$$

One can check that these definitions generalize the analogous definitions for syntagmata.

5

Morphisms of Syntagmata

This chapter introduces some simple notions, some well-known, for the following chapters. As is usual in algebra a morphism is a mapping preserving structures. In the case of syntagmata morphisms preserve the assignment of vocabulary. This is a very weak condition which, however, permits, for the moment, introduction of two elementary concepts: *symmetry* and *individual linearization*.

Two syntagmata are symmetric when we can obtain one from the other by reversing the loci. Symmetry extends to manifolds. We see that a symmetric manifold results from reversing the pattern. In particular symmetric manifolds are described by reversed types.

We distinguish *individual* and *global* linearizations. We present strings as a linear syntagmata whereby an individual linearization becomes an isomorphism of syntagmata. *Projectivity* is a well-known notion in dependency grammar. We introduce projectivity as a property of individual linearizations and we study some properties and their graphical representations.

5.1 The Category of Syntagmata

As is usual in algebra a morphism is a mapping preserving structures, cf. (Sankappanavar and Burris, 1981). In the following we will use the vocabulary of category theory, see for example (Adámek et al., 2004).

Definition 5.1. Given a pair of syntagmata S, S' we say that a mapping $f : \mathrm{Spt}(S) \longrightarrow \mathrm{Spt}(S')$ is a *morphism of syntagmata* iff f makes the following diagram commute:

$$\begin{tikzcd}
\mathrm{Spt}(S) \arrow[rr, "f"] \arrow[dr, "S"'] & & \mathrm{Spt}(S') \arrow[dl, "S'"] \\
& \Sigma &
\end{tikzcd}$$

We notate $f : S \longrightarrow S'$ a morphism. We say that f is a monomorphism iff it is injective, an epimorphism iff it is surjective, and an isomorphism when it is bijective. Two syntagma S, S' are isomorphic iff there is an isomorphism f between them.

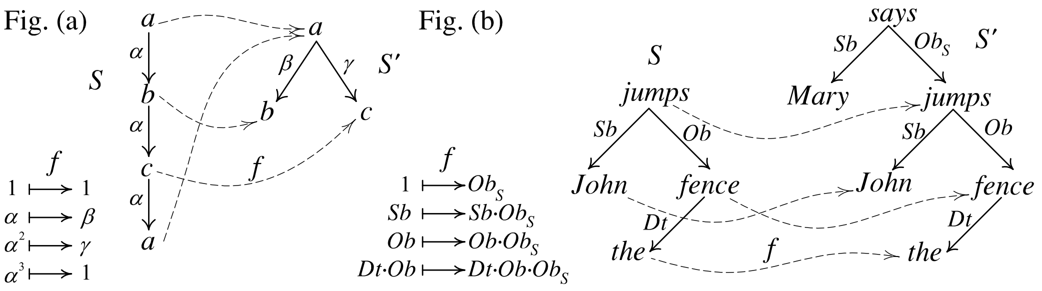


Figure 5.1: (a) an epimorphism of syntagmata $f : S \longrightarrow S'$; (b) a monomorphism of syntagmata $f : S \longrightarrow S'$.

Lemma 5.2. *We have a* category of syntagmata *with syntagmata in* **Synt** *as objects and morphisms of syntagmata as morphisms.*

Proof. We only need to see that the identity is a morphism of syntagmata, which is trivial, and that the composition of morphisms of syntagmata is also a morphism of syntagmata. To check this we just compose two triangular diagrams:

$$\begin{tikzcd}
\mathrm{Spt}(S) \arrow[r, "f"] \arrow[rr, bend left, "g\circ f"] \arrow[dr, "S"'] & \mathrm{Spt}(S') \arrow[r, "g"] \arrow[d, "S'"] & \mathrm{Spt}(S'') \arrow[dl, "S''"] \\
& \Sigma &
\end{tikzcd}$$

□

Fig.5.1 depicts some examples of the these concepts. Fig. 5.1(a) shows an epimorphism of syntagmata. Fig. 5.1(b) shows a monomorphism of syntagmata. Notice that the mapping f can be described as a translation $f(x) = x \cdot Ob_S$. In general subtrees are embedded into a syntagma through right-handed translations. Another example of monomorphism is given by the restricted syntagma (Definition 2.5). We just consider the inclusion mapping $\eta : \mathrm{Spt}(S\Gamma) = \mathrm{Spt}(S) \cap \Gamma \longrightarrow \mathrm{Spt}(S)$.

One is tempted to define that S' is a subsyntagma of S when there exists a monomorphism f of syntagmata $f : S' \longrightarrow S$. However all the notions above are defined by conditions which are very weak in the structural sense. Fig. 5.1(a) shows that the arrows (dependencies) are not necessarily preserved by a morphism. The restricted syntagma $S\Gamma$, although it will be algebraically useful, is not linguistically interesting. Amongst other things it has a lot of inconvenient null-loci. Since this notion of substructure is too general we will give later a more refined notion called *subsyntagma induced by a pattern* or Γ-subsyntagma, which forms a fundamental part of our formalism.

5.2 Symmetries

In (Cardó, 2016) we introduced symmetry as a permutation of a certain factorization of the supports of syntagmata. We call this ρ-symmetry. Even though this concept enjoys linguistics applications, here we are going to use in the main another kind of symmetry, even simpler, which allows us to explore more algebraic properties.

Definition 5.3. The reversal string x^R induces the mapping:

$$\begin{aligned} (\cdot)^{\mathfrak{R}} : \mathbf{Synt} &\longrightarrow \mathbf{Synt} \\ S &\longmapsto S^{\mathfrak{R}}, \end{aligned}$$

defined by $S^{\mathfrak{R}}(x) = S(x^R)$. We call this involutive mapping $\mathfrak{R}$-symmetry. We say that two syntagmata S, S' are $\mathfrak{R}$-symmetric iff $S' = S^{\mathfrak{R}}$. We extend $\mathfrak{R}$-symmetry to manifolds by $W^{\mathfrak{R}} = \{S^{\mathfrak{R}} \mid S \in W\}$. We say that two manifolds W, W' are $\mathfrak{R}$-symmetric iff $W' = W^{\mathfrak{R}}$. Since we are going to use only this kind of symmetry we call it simply *symmetry*.

Remark 5.4. Symmetry introduces a duality on the manifolds, and, as we will see, this duality works also at the level of linearizations. It is immediate to check that elementary notions of syntagmata (ellipsis, leaves, depth, ...) in a syntagma S become the corresponding co-notions in the symmetric syntagma $S^{\mathfrak{R}}$. Notice that $\mathrm{Spt}(S^{\mathfrak{R}}) = \big(\mathrm{Spt}(S)\big)^R$.

Lemma 5.5. *If $f : S \longrightarrow S'$ is a morphism of syntagmata S and S' then $f^{\mathfrak{R}}$ defined by $f^{\mathfrak{R}}(x) = (f(x^R))^R$ is a morphism of syntagmata $S^{\mathfrak{R}}$ and $S'^{\mathfrak{R}}$. If f is a mono/epi/iso-morphism then $f^{\mathfrak{R}}$ is a mono/epi/iso-morphism. Furthermore $(g \circ f)^{\mathfrak{R}} = g^{\mathfrak{R}} \circ f^{\mathfrak{R}}$ and $id^{\mathfrak{R}} = id$. So $(\cdot)^{\mathfrak{R}}$ is a covariant and isomorphic functor from the category of syntagmata* **Synt** *to itself.*

Proof. The first statement arises from the totally commutative diagram:

$$\begin{tikzcd}
\mathrm{Spt}(S) \arrow[rr, "f"] \arrow[dr, "S"] \arrow[dd, "R"'] & & \mathrm{Spt}(S') \arrow[dl, "S'"'] \arrow[dd, "R"] \\
& \Sigma & \\
\mathrm{Spt}(S^{\mathfrak{R}}) \arrow[ur, "S^{\mathfrak{R}}"] \arrow[rr, "f^{\mathfrak{R}}"] & & \mathrm{Spt}(S'^{\mathfrak{R}}) \arrow[ul, "S'^{\mathfrak{R}}"']
\end{tikzcd}$$

Since $(\cdot)^R$ is bijective, the injectivity and surjectivity of f is preserved in $f^{\mathfrak{R}}$. To see the composition of morphisms we just take a diagram for f and a diagram for g and concatenate:

$$\begin{tikzcd}
\mathrm{Spt}(S) \arrow[rr, "f"] \arrow[rrrr, bend left, "g\circ f"] \arrow[dr, "S"] \arrow[dd, "R"'] & & \mathrm{Spt}(S') \arrow[rr, "g"] \arrow[dl, "S'"'] \arrow[dd, "R"] & & \mathrm{Spt}(S'') \arrow[dlll, "S''"'] \arrow[dd, "R"] \\
& \Sigma & & & \\
\mathrm{Spt}(S^{\mathfrak{R}}) \arrow[ur, "S^{\mathfrak{R}}"] \arrow[rr, "f^{\mathfrak{R}}"] \arrow[rrrr, bend right, "g^{\mathfrak{R}}\circ f^{\mathfrak{R}}=(g\circ f)^{\mathfrak{R}}"'] & & \mathrm{Spt}(S'^{\mathfrak{R}}) \arrow[ul, "S'^{\mathfrak{R}}"'] \arrow[rr, "g^{\mathfrak{R}}"] & & \mathrm{Spt}(S''^{\mathfrak{R}}) \arrow[ulll, "S''^{\mathfrak{R}}"]
\end{tikzcd}$$

□

The meaning of this result is that a lot of properties should be preserved by symmetry. Let us consider an example:

Example 13. Symmetric Manifolds. Reviewing the first examples of manifolds, a remarkable fact is that W_{copy} and W_{squar} share the same valuations. Furthermore, the patterns in W_{copy} are the patterns of W_{squar}, but with the reversed order. The fact is that we can check graphically in Fig. (a) that they are symmetric: $Q^{\mathfrak{R}}_{x_1,\ldots,x_n} = C_{x_1,\ldots,x_n}$. So the manifolds are also symmetric: $W^{\mathfrak{R}}_{\text{copy}} = W_{\text{squar}}$. Another example of symmetry, see Fig. (b), is $W^{\mathfrak{R}}_{\text{mult}} = W_{\text{resp}}$, since $M^{\mathfrak{R}}_k = R_k$ for all syntagmata.

Fig. (a)

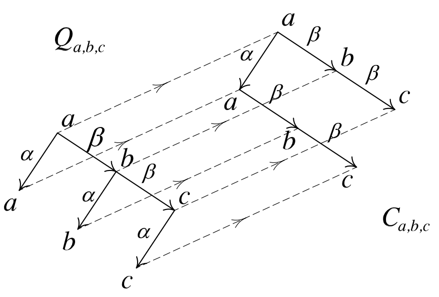
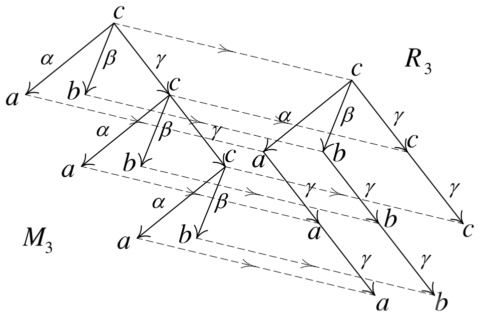


Fig. (b)



This example suggests that the algebraic content of a manifold commutes with the whole manifold under symmetry:

Lemma 5.6. *Let* Γ *be a pattern and B be a valuation. Then:*

$$\mathbf{Synt}\begin{pmatrix} B \\ \Gamma \end{pmatrix}^{\mathfrak{R}} = \mathbf{Synt}\begin{pmatrix} B \\ \Gamma^R \end{pmatrix}.$$

Proof. Since the arity here is not crucial, we abbreviate the quantified expression:

$$B(S(x_1), \ldots, S(x_a)) = 1 \ \forall (x_1, \ldots, x_a) \in \Gamma$$

as $B(S(x_i)) = 1 \ \forall (x_i) \in \Gamma$, and see the equalities:

$$\begin{aligned}
\mathbf{Synt}\begin{pmatrix} B \\ \Gamma \end{pmatrix}^{\mathfrak{R}} &= \{S \in \mathbf{Synt} \mid B(S(x_i)) = 1, \forall (x_i) \in \Gamma\}^{\mathfrak{R}} \\
&= \{S' \in \mathbf{Synt} \mid S' = S^{\mathfrak{R}},\ B(S(x_i)) = 1, \forall (x_i) \in \Gamma\} \\
&= \{S' \in \mathbf{Synt} \mid S'^{\mathfrak{R}} = S,\ B(S(x_i)) = 1, \forall (x_i) \in \Gamma\} \\
&= \{S' \in \mathbf{Synt} \mid B(S'^{\mathfrak{R}}(x_i)) = 1, \forall (x_i) \in \Gamma\} \\
&= \{S' \in \mathbf{Synt} \mid B(S'(x_i^R)) = 1, \forall (x_i) \in \Gamma\} \\
&= \{S' \in \mathbf{Synt} \mid B(S'(y_i)) = 1,\ (y_i) = (x_i)^R, \forall (x_i) \in \Gamma\} \\
&= \{S' \in \mathbf{Synt} \mid B(S'(y_i)) = 1,\ \forall (y_i) \in \Gamma^R\} \\
&= \mathbf{Synt}\begin{pmatrix} B \\ \Gamma^R \end{pmatrix}.
\end{aligned}$$
□

Corollary 5.7. *If* $W \in \mathbf{Man}(X)$ *then* $W^{\mathfrak{R}} \in \mathbf{Man}(X^{\mathfrak{R}})$.

Proof. Notice that $\mathfrak{R}$ is a bijective mapping and thus it preserves set union and set intersection. So if W and W' are simple manifods, $(W \cup W')^{\mathfrak{R}} = W^{\mathfrak{R}} \cup W'^{\mathfrak{R}}$ and $(W \cap W')^{\mathfrak{R}} = W^{\mathfrak{R}} \cap W'^{\mathfrak{R}}$. Then, first recall Lemma 3.9 and then apply Lemma 5.6. □

As an application of this result we have that manifolds in $\mathbf{Man}(\mathbb{H}\mathbf{k}^p)$ are decidable in linear time. We only have to built the symmetric syntagma (which is made in linear time in the size of the syntagma) and check if it is in the symmetric manifold, now in $\mathbf{Man}(\mathbf{k}^p\mathbb{H})$, and we know that this can be done in linear time (Theorem 4.13).

Lemma 5.8. *Given a syntagma* S *over* ζ *and* Σ, *we set* $m = |\zeta|$, $s = |S|$, $s' = |S^{\mathfrak{R}}|$, $d = \mathrm{depth}(S)$, $d' = \mathrm{depth}(S^{\mathfrak{R}})$, $e = |\mathrm{Ell}(S)|$, $e' = |\mathrm{Ell}(S^{\mathfrak{R}})|$. *We have that:*

$$s = s',\ \ d = d',\ \ \textit{and}\ e' \le \frac{m^{s'+e} - 1}{m - 1} - s' \ \textit{provided}\ m > 1;$$

when $m = 0$ *or* $m = 1$ *then* $e = e'$. *In particular if a manifold* W *has effectively bounded ellipticity,* $W^{\mathfrak{R}}$ *does.*

Proof. It is trivial to prove that $s = |S| = |\mathrm{Spt}(S)| = \mathrm{Spt}(S^{\mathfrak{R}})| = |S^{\mathfrak{R}}| = s'$. Let $x \in \mathrm{Spt}(S)$ be such that $|x| = d$. Then $S^{\mathfrak{R}}(x^R) = S((x^R)^R) = S(x) \neq 0$, since $x \in \mathrm{Spt}(S)$. Then the depth of S^R is greater than $|x^R| = |x|$ which is the depth of S. So we have that $\mathrm{depth}(S^{\mathfrak{R}}) \geq \mathrm{depth}(S)$. We can use this twice: $\mathrm{depth}(S) = \mathrm{depth}((S^{\mathfrak{R}})^{\mathfrak{R}}) \geq \mathrm{depth}(S^{\mathfrak{R}}) \geq \mathrm{depth}(S)$ and then $d = \mathrm{depth}(S) = \mathrm{depth}(S^{\mathfrak{R}}) = d'$. When $m = 0$ or 1, then trivially $S = S^{\mathfrak{R}}$ and then $e = e'$. Consider $m > 1$; by Lemma 2.6 $d+1 \leq s+e \leq \frac{m^{d+1}-1}{m-1}$ and $d' + 1 \leq s' + e' \leq \frac{m^{d'+1}-1}{m-1}$; then $e' \leq \frac{m^{d'+1}-1}{m-1} - s' = \frac{m^{d+1}-1}{m-1} - s' \leq \frac{m^{s+e}-1}{m-1} - s' = \frac{m^{s'+e}-1}{m-1} - s'$. Thus if W has effectively bounded ellipticity with $|\mathrm{Ell}(S)| \leq \varepsilon(|S|)$, then $W^{\mathfrak{R}}$ has effectively bounded ellipticity with $|\mathrm{Ell}(S)| \leq \frac{m^{|S|+\varepsilon(|S|)}-1}{m-1} - |S|$. □

5.3 Linearizations and Projectivity

5.3.1 Strings as Syntagmata, Individual and Global Linearizations

Notice that a string in Σ^* can be understood as a syntagmata. Let us consider a syntactic function u understood as "the next". Then a string $x_1 + \cdots + x_n \in \Sigma^*$ can be thought of as a mapping $x : u^* \longrightarrow \Sigma_+$, such that:

$$x(1) = x_1,\ \ x(u) = x_2,\ \ x(u^2) = x_3,\ \ \ldots,\ \ x(u^{n-1}) = x_n,\ \ x(u^n) = 0,\ \ x(u^{n+1}) = 0,\ \ \ldots.$$

We call syntagmata with just one syntactic function *linear*. We can endow linear syntagmata with the obvious operation:

$$(a_1 \xrightarrow{u} \cdots \xrightarrow{u} a_n) + (b_1 \xrightarrow{u} \cdots \xrightarrow{u} b_m) = a_1 \xrightarrow{u} \cdots \xrightarrow{u} a_n \xrightarrow{u} b_1 \xrightarrow{u} \cdots \xrightarrow{u} b_m$$

Clearly this operation is associative and $0^\bullet$ is the identity, which properties provide us with a monoid. Since atomic syntagmata are a special case of linear syntagmata, we can consider the set generated by atoms under the above sum, which we notate $\mathbf{Atom}^\star$. This set is the set of linear syntagma which are not elliptic. So definitively we have an isomorphism:

$$\begin{aligned} \Sigma^* &\longrightarrow \mathbf{Atom}^\star \\ a_1 + \cdots + a_n &\longmapsto a_1^\bullet + \cdots + a_n^\bullet = a_1 \xrightarrow{u} \cdots \xrightarrow{u} a_n. \end{aligned}$$

Since this isomorphism $\Sigma^* \cong \mathbf{Atom}^\star$ is so natural we are be able to represents strings by both forms; we can even say by a mild abuse of notation that $\mathbf{Atom}^\star = \Sigma^*$.

Definition 5.9. Lets $S \in \mathbf{Synt}$ and $x \in \Sigma^*$. An *individual linearization* is an isomorphism of syntagmata $S \longrightarrow x$, i.e.: a bijection ℓ that makes the following diagram commute:

$$\begin{tikzcd}
\mathrm{Spt}(S) \arrow[rr, "\ell"] \arrow[dr, "S"'] & & \mathrm{Spt}(x) \arrow[dl, "x"] \\
& \Sigma &
\end{tikzcd}$$

A *global linearization* is a mapping $\Pi : \mathbf{Synt} \longrightarrow \wp(\Sigma^*)$ such that for all $x \in \Pi(S)$ there is an individual linearization $\ell : S \longrightarrow x$. Notice that then $\Pi(S)$ is a finite set of strings.

Example 14. An individual linearization. Take the syntagma S from the first example and take the string:

$$\text{the + young + soldier + washed + the + dirty + cup},$$

or what amounts to the same thing, the string in **Atom***:

$$x = \text{the}^\bullet + \text{young}^\bullet + \text{soldier}^\bullet + \text{washed}^\bullet + \text{the}^\bullet + \text{dirty}^\bullet + \text{cup}^\bullet.$$

The following mapping defines a linearization $\ell : \mathrm{Spt}(S) \longrightarrow \mathrm{Spt}(x)$. $\ell(Dt \cdot Sb) = 1$, $\ell(Ad \cdot Sb) = u$, $\ell(Sb) = u^2$, $\ell(1) = u^3$, $\ell(Dt \cdot Ob) = u^4$, $\ell(Ad \cdot Ob) = u^5$, $\ell(Ob) = u^6$. Fig. (a) represents graphically this mapping ℓ on the loci; the mapping S is not represented. Fig. (b) shows the same linearization with the mappings S and x; this will be the standard form of representing an individual linearization.

Fig. (a)

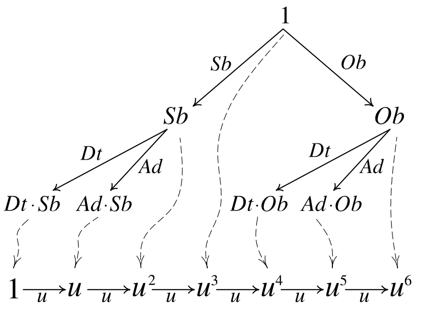
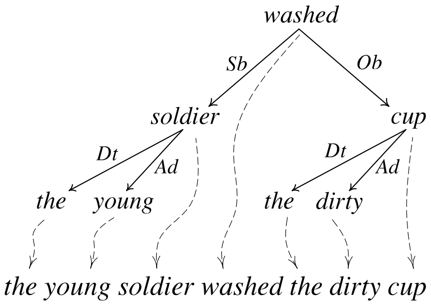


Fig. (b)



So "individual linearization" just means an ordering of the lexical items of a specific given isolated syntagma, while "global linearization" means an a organization many isolated linearizations. When the context permits we avoid the distinction individual-global. However this distinction is fundamental.

Our interpretation of dependency grammar is as a pair $\mathscr{G} = (W, \Pi)$ where W is a manifold and Π is a global linearization. The language of the grammar is then the set $\mathscr{L}(\mathscr{G}) = \bigcup_{S \in W} \Pi(S)$. However some constraint over the concept of global linearization is required. We will need some additional mathematical preliminaries in order to arrive at an adequate formulation, which is made in the following two chapters. Combining a manifold with this adequate global linearization we will arrive at the concept of *Algebraic Dependency Grammar*.

Notwithstanding we can continue here studying linearizations individually. We examine in the following section a property of individual linearizations much celebrated in dependency grammar, namely *projectivity*.

5.3.2 Projectivity

Since linearizations, understood as simple orderings of the nodes of a dependency tree, are too general; theorists have tried to find adequate constraints for natural language. The most fundamental class of linearizations considered is the class of *projective linearizations* or simply *projections*. Intuitively a projection is a relation between the dependency tree and a linear order the graphical representation of which connects the vertices of the tree to the vertices of the linear order by lines without crossing each other; see Fig. 5.2. However some additional restrictions must be made for a graphical representations, see later §5.4. Before giving a geometric characterization it is preferable to have a more formal definition of projectivity. There exist several equivalent characterizations in terms of adjacency, governance and intervals.[1] Here we take as reference a characterization of Kuhlmann (2010):[2] *a linearization is* projective *iff it transforms subtrees of the dependency tree into intervals of the word-order*;[3] see Fig. 5.3. Later we will translate this definition into our vocabulary of syntagmata.

Projectivity appears to capture well the linguistic intuition of "continuity". Nonetheless, projective structures do not cover the totality of the sentences of natural language.[4] To explain the non-projective part there are two options. Either we must assume a less intuitive analysis of the sentences, or we must loosen the sense of projectivity. The first option is not considered a good alternative. Generally, even in a syntactic representation, the dependencies in the dependency tree must conform to the semantic arguments (although it is an intuitive condition rather a formal constraint).[5] For this reason other wider kinds of linearizations have been proposed, such as *pseudo-projective* (Kahane et al., 1998), *planar*[6] (Temperley et al., 1993), *multiplanar* (Yli-Jyrä et al., 2003), *well-nested* (Obrebski and Gralinski, 2004), or *block-degree restricted* (Kuhlmann, 2010) linearizations. See Kuhlmann and Nivre (2006) for general discussion. We also will consider other forms of linearization in Chapters 7 and 11.

In sum, projectivity is a property of individual and isolated linearizations which, although it does not explain natural language in its entirety, any formalism should take into account.

[1] Kuhlmann (2010), following Marcus (1967), lists at least four characterizations: those of K. Harper and D. Hays; Y. Lecerf and P. Ihm; S. Filiatov; and J. Robinson. Some sources attribute to Lecerf (1961) the first characterization.

[2] Which in its turns is based in a characterization attributed to S. Filiatov.

[3] Kuhlmann (2010) says "convex sets" instead of intervals.

[4] See (Kuhlmann, 2010; Debusmann and Kuhlmann, 2010; Straka et al., 2016) for some statistics.

[5] In (Cardó, 2016) we explore this possibility.

[6] Kuhlmann (2010) calls these linearizations "weakly non-projective" in order to avoid confusion with the more widespread concept of planarity for general graphs.

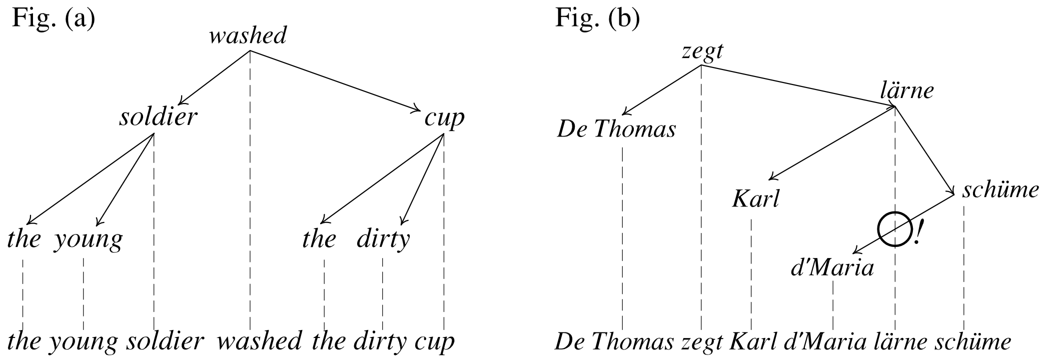


Figure 5.2: (a) projective linearization; (b) non-projective linearization.

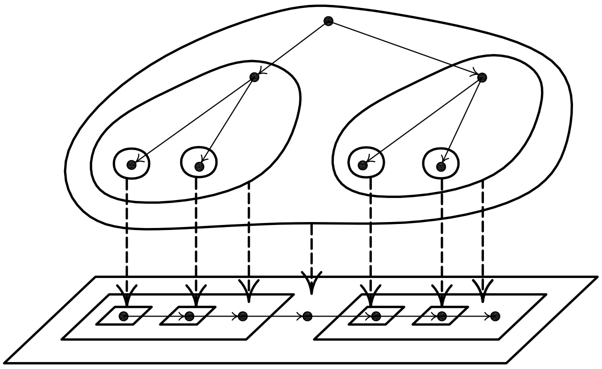


Figure 5.3: the projective linearization from Fig. 5.2(a) which preserves subtrees into intervals.

5.3.3 Projectivity through Syntagmata

Although *subtree* and *substring* are clear concepts we must formalize them. We can capture the notion of (total) subtree using patterns. Note that given a locus $\varphi \in \zeta^*$ the elements of the right ideal pattern (of arity one) $\zeta^*\varphi$ depict a subtree. An *interval* of u^* is a set of the form: $\{u^{p+1}, u^{p+2}, \dots, u^q\}$, for some integers $0 \le p \le q$. A *subtree* of a syntagma S is the set of loci $\mathrm{Spt}(S\zeta^*\varphi) = \zeta^*\varphi \cap \mathrm{Spt}(S)$ for some fixed $\varphi \in \zeta^*$, provided it is not the empty set. In the sequel our definition of projectivity will be the following:

Definition 5.10. Let S be a non-elliptic syntagma and let x be a string. A linearization $\ell : S \longrightarrow x$ is *projective* iff ℓ transforms subtrees into intervals. We call these linearizations *projections*.

5.4 Some Properties and Graphical Representations of Projections

♣ We study a couple of elementary properties and well-known graphical characterizations of projectivity. Since the issue is well-known the informed reader may skip this

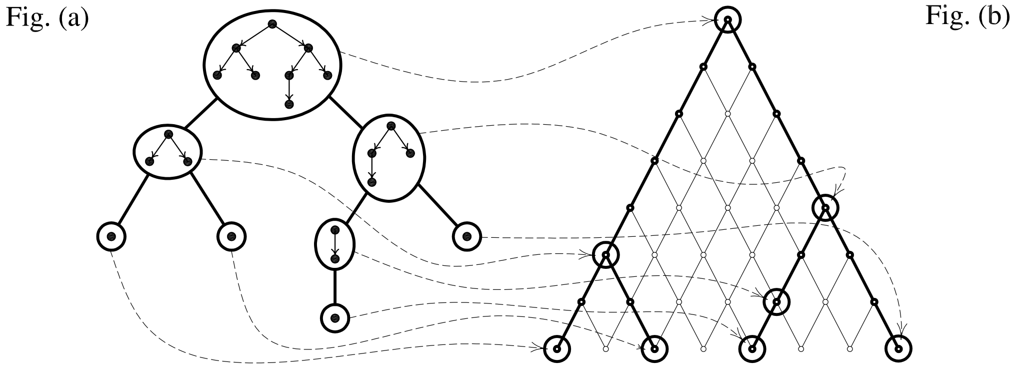


Figure 5.4: an example of a projective linearization read as a monomorphism of orders: (a) the set of all subtrees of a given syntagma ordered by inclusion; (b) the set of intervals ordered by inclusion.

section.

A nice way to read the above definition is through order theory. Notice on the one hand that the set of subtrees of a non-elliptic syntagma S, notated $\textbf{SubTrees}(S) = \{\mathrm{Spt}(S) \cap \zeta^*\varphi \mid \varphi \in \zeta^*\}$, can be ordered by inclusion, which gives a partial order. On the other hand the intervals of the support of a string x, notated $\textbf{Intervals}(x)$, form also a partial ordered set. We have that if ℓ is projective it induces a $\subseteq$-monomorphism of orders:

$$\ell : \textbf{SubTrees}(S) \longrightarrow \textbf{Intervals}(x),$$

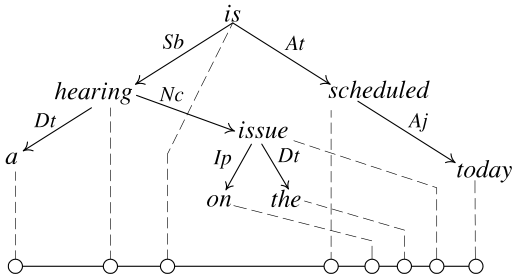
where we notate also ℓ the canonical set extension, $\ell(t) = \{\ell(x) \mid x \in t\}$ for any tree t. The proof is immediate: the induced mapping is well-defined because ℓ is projective; in addition since ℓ is injective the set extension is. However surjectivity is not preserved: there are many more intervals than subtrees. See Fig. 5.4 for an example.

The following is an easy result which bounds the distance between words in a projective linearization.

Lemma 5.11. *Given a projective linearization the distance in the linear order of two nodes of the dependency tree is strictly less than the size of the least subtree containing the nodes.*

Proof. Let t be the least subtree containing $\varphi, \psi \in \zeta^*$. Since the linearization is projective this tree is transformed into an interval, $\ell(t)$ which has length $|\ell(t)|$. In the worst case $\ell(\varphi)$ and $\ell(\psi)$ are placed at the extremes, and then the distance is $|\ell(t)| - 1$. So the distance is strictly less than $|\ell(t)|$. □

Example 15. An example of the metric property. Let us see an example of application of the last lemma. Consider the linearization ℓ of the following figure:



We want to prove that it is indeed non-projective. Although the picture shows crossing lines, in order to be completely sure that it is not a projective linearization we would have to check that redrawing the linearization cannot remove the crossing points.

Alternatively, in virtue of the previous lemma, we can consider the words *a* and *issue*, which are placed at the loci *Dt·Sb* and *Nc·Sb* respectively. Its distance in the linear order is 6. However the least tree containing both loci is given by root *Sb* (with the word *hearing*) which has size 5, which means that ℓ is not projective.

There exist two planar representations for linearizations, namely by *straight lines* and by *arcs*. Here we define and prove that both representations are equivalent. In the following chapters when we need to exhibit an individual linearization we will use the representation by straight lines.[7] For the rest of this section we are going to suppose that all syntagmata are non-elliptic.

Consider the first representation. We introduce a little geometric notation. $R^2 = \{(x_1, x_2) \in \mathbb{R}^2 \,|\, x_2 > 0\}$ is the superior half-plane and $R = \{(x_1, x_2) \in \mathbb{R}^2 \,|\, x_2 = 0\}$ the base line. π_1 and $\pi_2 : R^2 \longrightarrow R$ are the orthogonal projections $\pi_1(x_1, x_2) = x_1$, $\pi_2(x_1, x_2) = x_2$. $\|(x, y)\| = \sqrt{x^2 + y^2}$. Finally, given a pair A, B of points in the plane, we notate the open segment $\overline{AB} = \{A + t(B - A) \,|\, t \in (0, 1) \subset \mathbb{R}\}$.

Definition 5.12. A *straight representation* of a linearization ℓ is a pair of injective mappings f, g such that they make the following diagram commute:

$$\begin{tikzcd}
\mathrm{Spt}(S) \arrow[r, "f"] \arrow[d, "\ell"'] & R^2 \arrow[d, "\pi_1"] \\
\mathrm{Spt}(x) \arrow[r, "g"] & R
\end{tikzcd}$$

The segments $\overline{f(\varphi)f(\psi)}$ such that φ and ψ are adjacent are called *arrows*. The segments $\overline{f(\varphi)g(\ell(\varphi))}$ are called *projection lines*.[8]

[7] For more considerations on linearization under a geometric perspective see (Marcus, 1967).

[8] Notice that $\overline{f(\varphi)g(\ell(\varphi))} = \overline{f(\varphi)\pi_1(f(\varphi))}$.

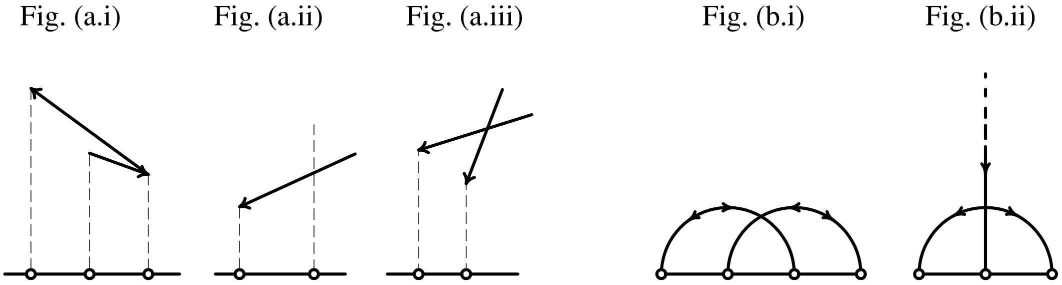


Figure 5.5: (a) prohibited configurations for a straight representation of a projective linearization; (b) prohibited configurations for a representation by arcs of a projective linearization.

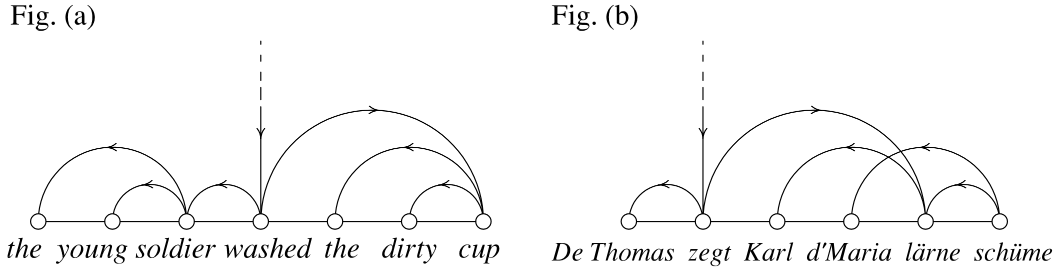


Fig. (a)
Fig. (b)
the young soldier washed the dirty cup
De Thomas zegt Karl d'Maria lärne schüme

Figure 5.6: representation by arcs of a projective linearization (a) and a non-projective linearization (b).

Given a linearization and astraight representation we establish the three conditions:

(a.i) if $\varphi \prec \psi$ then $\pi_2(f(\varphi)) < \pi_2(f(\psi))$, for any $\varphi, \psi \in \mathrm{Spt}(S)$;

(a.ii) arrows never intersect each other;

(a.iii) arrows and projection lines never intersect each other.

These three conditions are in fact defining prohibited configurations. Fig. 5.5(a.i), (a.ii) and (a.iii) show them. Fig. 5.2(a) depicts a straight representation of a projective linearization, while Fig. 5.2(b) is not projective. We see in a few moments that the three conditions characterize projectivity.

In the second representation there are no projection lines and the straight segments for the arrows are replaced by arcs. Given two points $A, B \in R$, we call the *arc* $\widehat{AB} = \{(x, y) \in R^2 \mid \left\|(x, y) - \frac{A+B}{2}\right\| = \frac{\|A-B\|}{2}\}$ i.e. the superior half of the unique circumference containing both A and B with center in R. Given a point A in the plane the *vertical ray* is the set $\overrightarrow{A} = \{(0, t) \mid t > 0\}$.

Definition 5.13. A *representation by arcs* of a linearization ℓ is an injective mapping $f : \mathrm{Spt}(S) \longrightarrow R$ such that if $\ell(\varphi) \prec \ell(\psi)$ then $f(\varphi) < f(\psi)$ for any $\varphi, \psi \in \mathrm{Spt}(x)$. We call *root ray* the set $\overrightarrow{f(1)}$, where recall that 1 is the root locus. Given two adjacent loci φ, ψ we call *arc-arrow* the arc: $\overparen{f(\varphi)f(\psi)}$.

Given a linearization and a representation by arcs we establish the two conditions:

(b.i) arc-arrows never intersect each other;

(b.ii) any arrow and the root ray never intersect.

In both straight and arc representations it is usual to draw the direction of the arrows. Similarly to the straight representation, these two conditions are in fact defining prohibited configurations. Fig. 5.5(b.i) and (b.ii) show them. Fig. 5.6(a) depicts a representation by arcs of a projective linearization, while Fig. 5.6(b) is not projective.

Theorem 5.14. *A linearization is projective if and only if there is a straight representation satisfying the conditions* (a.i), (a.ii), (a.iii). *Equivalently a linearization is projective if and only if there is a representation by arcs satisfying the conditions* (b.i), (b.ii). [9]

Proof. We sketch the proof. We see the first statement. (⇒) By induction on the depth d of the syntagma. The case $d = 0$ is trivial. We assume that the induction hypothesis is true for any subsyntagma with depth d, and consider a syntagma with depth $d + 1$. Any syntagma can be decomposed in its greatest proper subtrees together with the root. These subtrees have depth d and thus they satisfies the three conditions by induction hypothesis. We place them in the plane sufficiently separated and we add the root sufficiently high. Then the whole configuration satisfies the three conditions. (⇐). Consider a subtree. Geometrical conditions (a.i), (a.ii) and (a.iii) ensure that there is a polygon defined by the base line, arrows and projection lines which contains the subtree; see Fig. (a). Then since all the nodes of the subtree are in the polygon if we project them, they fall in the interval of the base line.

Fig. (a)

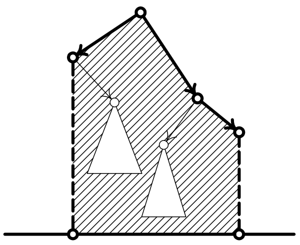


Fig. (b)

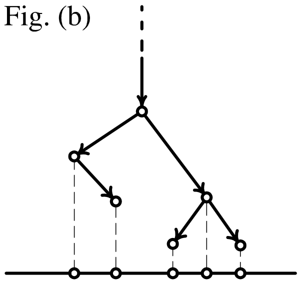
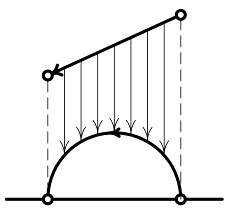


Fig. (c)



Now we prove that we have a straight representation satisfying (a.i), (a.ii), (a.iii) iff we have a representation by arcs satisfying (b.i), (b.ii). First add an extra special arrow

[9] If we remove the condition (b.ii) on the root ray we obtain the class of planar (or weakly non-projective) linearizations, (Temperley et al., 1993).

in the straight representation which targets the root and which sources from infinity, as in Fig. (b). It can be proved that if we replace the condition (a.i) by a condition (a'.i) which says that no arrow crosses the root arrow, then (a.i), (a.ii), (a.iii) are equivalent to (a'.i), (a.ii), (a.iii). So we only have to prove that these last conditions are equivalent to (b.i), (b.ii). In order to see this we consider the mapping of each arrow into an arc as in Fig. (c). Then the condition (a'.i) turns into condition (b.i), and conditions (a.ii) and (a.iii) turns into condition (b.ii). □

Although the representation by arcs is more concise and mathematically more elegant, it cannot separate the syntagma from the linear order and for large sentences it is difficult to grasp the structure, whereby in the sequel we adopt the straight representation.

6

Subsyntagmata Induced by Patterns

Substructure is a fundamental notion in linguistics: it is the basis of the understanding of many phenomena. We review first several notions of substructure.

This chapter presents as our notion of substructure subsyntagmata described by a pattern. This will allows us to control the parts of a syntagma in order to linearize it in the following chapters. Patterns are just a set of loci, not syntagmata, whereby we have to explain how a pattern induces a subsyntagma.

Since this construct is not immediate we study first the more intuitive case of subtrees or right ideal patterns. Then we give the general definitions in order to arrive at the definition of *subsyntagma induced by a pattern* or Γ-*subsyntagma*. In particular we prove that the subsyntagma induced is always unique and we show how to calculate it. We will provide several examples. In the last section we observe that induced subsyntagmata and symmetry commute perfectly.

6.1 A Note on the Importance of Substructures

The notion of substructure is as fundamental in linguistics as it is in algebra. In the field of algebra the set of substructures of a given structure informs about the nature of the latter; it can even identify it totally. Hence inquiry into an algebraic structure usually begins by studying substructures.

In linguistics the concept of substructure is still an object of debate today. There is much evidence that a sentence is not a monolithic unit. These arguments are the called *constituency tests* or more in general *diagnosis tests* and they appear frequently in traditional syntax monographs. Consider for example one such test, *pro-form substitution*:

(18) A man with dark glasses is following us ⇝ He is following us.

Since the substitution is successful the underlined substring is considered a *constituent* of the sentence. Hence it would seem that one could easily deduce an underlying syntactic structure. However in many situations tests fail or are not definitive and there is no a unequivocal procedure to recover the structure.[1,2]

The most basic and ancient division of a sentence, *subject-predicate*, goes back to Aristotle.[3] Nowadays it is not such a clear division: in dependency grammar it is not clearly accepted (however see in the last chapter §12.2 a proposal), while in constituency grammar and categorial grammar this is substituted by the opposition *noun phrase-verb phrase*. More in general the widespread notion of substructure is the *constituent* which is framed into the phrase-structure formalisms, also called constituency approaches.

In dependency grammar several proposals for substructures have been suggested: *(partial) subtrees* and *(complete) subtrees* (Hays, 1958), *word order domains* (Bröker, 1998), or more recently, *catena* (Osborne et al., 2012), which is a very relaxed notion. According to Osborne et al. (2012, pg 354) a catena is: "a word or combination of words that is continuous with respect to dominance". Here "continuous" in graph-theoretical terms means a connected subset of nodes of the dependency tree.[4] Catenae help to understand finer questions such as idioms, ellipses or morphology, cf. (Osborne et al., 2012; Groß, 2011). However in our opinion an eventually "discontinuous" substructure notion could help to understand better some constructions. Consider, for instance, the disconnected subjects in a sentence with subordinate clauses in some Germanic languages, which seem to behave as a unitary substructure in the linearization, see Example 19 later.

Here we will introduce substructures as subsyntagmata induced by a pattern which: (1) include a rich variety of combinations like in the notion of catena, but in addition (2) include "discontinuous" cases; (3) are defined algebraically, not geometrically; (4) can be located in a classificatory monoid, which helps us to discern phenomena.

6.2 Warm-Up: Subtrees

Since our main objects are syntagmata which we depict as trees, the first substructures we consider are subtrees. We already saw that when we depict the loci of the right ideal $\zeta^*\varphi$ we obtain a subtree. However this is not a syntagma but a set of loci.

The most abstract notion of substructure must satisfy a pair of requirements. The substructure must have the same status as the first structure (cf. subgroups are groups

[1] See Groß and Osborne (2015) and Osborne (2015) for some diagnosis tests in the dependency framework.

[2] As we have said, pro-form substitution states that if we substitute a substring by a pro-form with a grammatical result, then the substring is a constituent. But simple string manipulations are dangerous, as in the substitution *I think that the first answer is right* ⇝ *it is right*; the adequate constituent is *the first answer is right*, not the underlined substring. There are other similar wrong diagnoses, see for example (Phillips, 2003).

[3] See (Matthews, 1981).

[4] Hays had already defined "subtree" as "any connected set of nodes contained in a tree" (Hays, 1958, pg 261). In this thesis the term "subtree" means a complete subtree, i.e. is the set of loci which are dominated by a unique locus, also expressed by right ideal patterns.

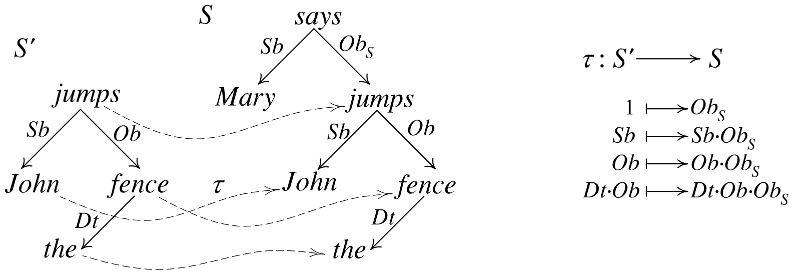


Figure 6.1: a subsyntagma S' of S defined by $S(x) = S(x \cdot Ob_S)$ and its embedding as a translation, $\tau : S' \longrightarrow S$, $\tau(x) = x \cdot Ob_S$.

or subgraphs are also graphs). In addition the substructure must be embeddable in the first structure, i.e. there must be a monomorphism.

For example the restricted subsyntagma $S\Gamma$ can be embedded into S. These requirements, which are framed in category theory, see for example Adámek et al. (2004), are a bit more general than the classical notion of a substructure in universal algebra, i.e. a set with certain inner operations, see for example Sankappanavar and Burris (1981). However we do not have operations in a syntagma to be preserved in a subsyntagma. $S\Gamma$ is just the result of removing loci outside the pattern Γ and this says nothing about the relation between $S\Gamma$ and S; in other words, the embedding is the trivial inclusion. We would like something similar, or better, something isomorphic to $S\Gamma$, where the embedding is not trivial.

In this relation consider again the notion of subtree as a right ideal $\Gamma = \zeta^*\varphi$. Note that all the loci of the support of $S\Gamma$ are of the form $x\varphi$ for some $x \in \zeta^*$. Consider the new syntagma given by $S'(x) = S(x\varphi)$. Now the support of S' is derived from the same set but we have deleted the constant φ at the end:

$$x \in \mathrm{Spt}(S') \iff x\varphi \in \mathrm{Spt}(S\,\zeta^*\varphi).$$

Moreover, we have an isomorphism of syntagmata:

$$\begin{tikzcd}
\mathrm{Spt}(S') \arrow[rr, "\tau_\varphi"] \arrow[dr, "S'"'] & & \mathrm{Spt}(S\zeta^*\varphi) \arrow[dl, "S\zeta^*\varphi"] \\
& \Sigma &
\end{tikzcd}$$

There are two ways to conceptualize S'. First, a geometric way where we can think of loci of $\mathrm{Spt}(S) \cap \zeta^*\varphi$ as translated upwards in such a way that now the root of S' (given by the identity 1) is the root of $\zeta^*\varphi$; see the example in Fig. 6.1. Indeed this translation

$\tau_\varphi(x) = x\varphi$ gives the monomorphism which embeds S' into S:

$$\begin{tikzcd}
\mathrm{Spt}(S') \arrow[rr, "\tau_\varphi"] \arrow[dr, "S'"'] & & \mathrm{Spt}(S) \arrow[dl, "S"] \\
& \Sigma &
\end{tikzcd}$$

The injectivity of τ is given by the cancellation property of free monoids. Thus $S' = S \circ \tau_\varphi$ is a subsyntagma of S in the above sense.

A second way to understand S' is combinatoric rather than geometric. We have the relation $x \in \mathrm{Spt}(S') \iff x\varphi \in \mathrm{Spt}(S\,\zeta^*\varphi)$. So we can consider that the φ's are superfluous, or better, redundant in the syntagma $S\Gamma$. If we delete them we will obtain the support of S'. Interestingly this second procedure is more productive and generalizable than the geometric interpretation.

Subtrees are closely related to right ideals (of arity one). Are these right ideals $\zeta^*\varphi$ the unique patterns which proportion substructures? Consider the simple construction $\varphi\zeta^*$ where we have just swapped the handedness. These left ideals should, by the same token, become substructures. In the end "subtre" is a geometrical notion depending on the handedness of the graphical representation. Right ideals draw a tree-shaped cloud of nodes while left ideals draw another distribution of the nodes which we can name co-subtrees. Fig. 6.2(a)-(d) shows the distribution of loci in some patterns. Importantly we will see later that right ideals are closely related to projective linearizations.

But indeed why should we stop here? We can take more complex patterns, like say $\varphi\zeta^*\psi\xi^*$. Can more general patterns induce subsyntagmata which yield other linearizations? This chapter provides algebraic conditions to answer positively these questions.

6.3 Induced Subsyntagmata: Definitions

In the following sections all the patterns will be of arity 1. Since we want to define certain mappings over the patterns, we need to be able to decompose $x \in \Gamma$ univocally. Recall that a pattern with its decomposition $\Gamma = \Gamma_1 \cdots \Gamma_k$ in basic patterns Γ_i is proper iff it is unambiguous (i.e. the product mapping $\pi : \bigoplus_{i=1}^k \Gamma_i \longrightarrow \Gamma$ is bijective) and it has no trivial factors. Let us introduce the following mapping:

Definition 6.1. Let $\bigoplus_{i=1}^k \Gamma_i$ be a factorization of a pattern Γ. We say that a mapping of the form:

$$\begin{aligned}
\partial : \bigoplus_{i=1}^{k} \Gamma_i &\longrightarrow \bigoplus_{i=1}^{k} \partial_i(\Gamma_i) \\
(x_1, \dots, x_k) &\longmapsto (\partial_1(x_1), \dots, \partial_k(x_k)),
\end{aligned}$$

is a *deletion* iff for each $i = 1, \dots, k$, ∂_i is the constant mapping 1 or the identity id_{Γ_i}. In other words ∂ deletes some components of the factorization making them 1.

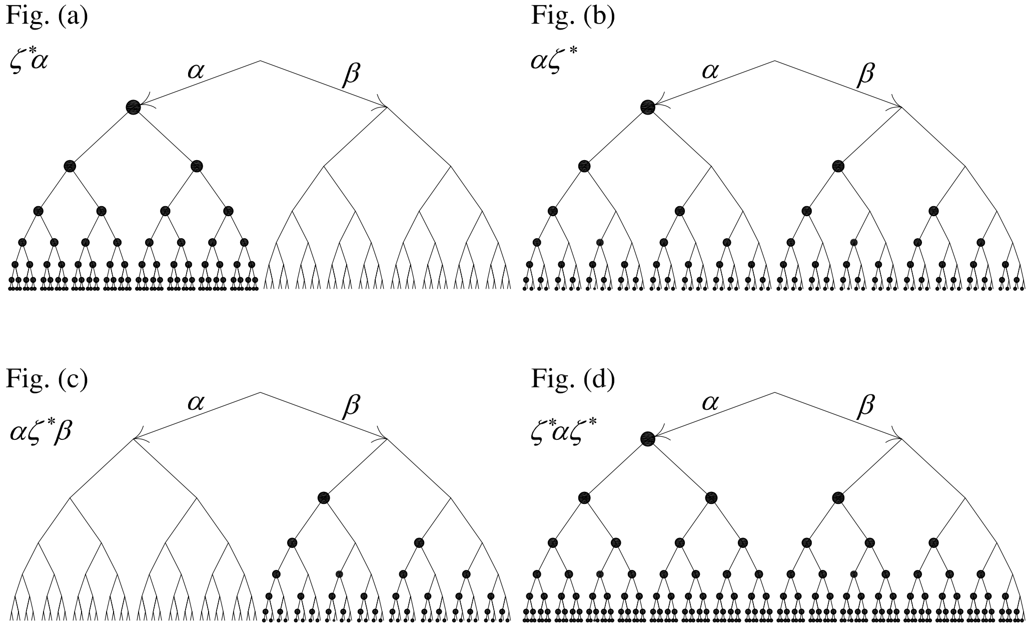


Figure 6.2: distribution of the loci for several patterns with $\zeta = \{\alpha, \beta\}$; (a) $\zeta^*\alpha$; (b) $\alpha\zeta^*$; (c) $\alpha\zeta^*\beta$; (d) $\zeta^*\alpha\zeta^*$.

Lemma 6.2. *Let $\pi : \bigoplus_{i=1}^k \Gamma_i \longrightarrow \Gamma$ be a proper factorization and let ∂ be a deletion. Then there exists a unique surjective mapping $\widetilde{\partial}$, a product mapping $\widetilde{\pi}$ and a pattern Θ, which make the following diagram commute:*

$$\begin{tikzcd}
\bigoplus_{i=1}^k \Gamma_i \arrow[r, "\partial"] \arrow[d, "\pi"'] & \bigoplus_{i=1}^k \partial_i(\Gamma_i) \arrow[d, dashed, "\widetilde{\pi}"] \\
\Gamma \arrow[r, dashed, "\widetilde{\partial}"] & \Theta
\end{tikzcd}$$

where $\widetilde{\pi} : \bigoplus_{i=1}^k \partial_i(\Gamma_i) \longrightarrow \Theta$ is a factorization. In other words, $\widetilde{\partial}$ deletes some factors of Γ being $\Theta = \widetilde{\partial}(\Gamma)$ the result of this deletion.

Proof. Consider the mapping $\widetilde{\pi} : \bigoplus_{i=1}^k \partial_i(\Gamma_i) \longrightarrow \widetilde{\pi}\big(\bigoplus_{i=1}^k \partial_i(\Gamma_i)\big)$ with $\widetilde{\pi}(y_1, \ldots, y_k) = y_1 \cdots y_k$. The set $\widetilde{\pi}\big(\bigoplus_{i=1}^k \partial_i(\Gamma_i)\big) = \prod_{i=1}^k \partial_i(\Gamma_i)$ is a pattern because each $\partial_i(\Gamma_i)$ is a constant or a submonoid.

Now since the mappings must make the diagram commute, $\widetilde{\partial} = \widetilde{\pi} \circ \partial \circ \pi^{-1}$ which exists because π is bijective and in addition this is the only possible choice for $\widetilde{\partial}$. Then we can calculate $\widetilde{\partial}(\Gamma) = \widetilde{\pi} \circ \partial \circ \pi^{-1}(\Gamma) = \widetilde{\pi} \circ \partial\big(\bigoplus_{i=1}^k \Gamma_i\big) = \widetilde{\pi}\big(\bigoplus_{i=1}^k \partial_i(\Gamma_i)\big)$. Clearly $\widetilde{\partial}$ is surjective. Now we can take $\Theta = \widetilde{\partial}(\Gamma)$. □

Definition 6.3. Let $\pi : \bigoplus_{i=1}^k \Gamma_i \longrightarrow \Gamma$ be a proper factorization and let S be a syntagma.

Consider also the j-projection:

$$\begin{aligned}\pi_j : \bigoplus_{i=1}^{k} \Gamma_i &\longrightarrow \Gamma_j \\ (x_1, \dots, x_k) &\longmapsto x_j.\end{aligned}$$

We define the *deletion associated to S and Γ and its factorization* as $\partial = (\partial_1, \dots, \partial_k)$ where for $j = 1, \dots, k$,

$$\begin{aligned}\partial_j \text{ is the constant mapping } 1 &\iff |\pi_j \circ \pi^{-1}(\mathrm{Spt}(S\Gamma))| \leq 1; \\ \partial_j \text{ is the identity } \mathrm{id}_{\Gamma_j} &\iff |\pi_j \circ \pi^{-1}(\mathrm{Spt}(S\Gamma))| > 1.\end{aligned}$$

Given a syntagma S and a proper factorization of Γ, we say that a syntagma S' is a *subsyntagma induced by the pattern* Γ (or in short a Γ-*subsyntagma*) of S iff its associated deletion mapping ∂ yields an isomorphism $S\Gamma \cong_{\widetilde{\partial}} S'$; that is, we have the isomorphism of syntagmata:

$$\begin{tikzcd}
\mathrm{Spt}(S\Gamma) \arrow[rr, "\widetilde{\partial}"] \arrow[dr, "S\Gamma"'] & & \mathrm{Spt}(S') \arrow[dl, "S'"] \\
& \Sigma &
\end{tikzcd}$$

Remark 6.4. In order to be absolutely formal we should say that S' is a $\bigoplus_i \Gamma_i$-subsyntagma or that S' is the subsyntagma induced by the factorization of Γ. However by a mild abuse of terminology, we will suppose that the factorization is given implicitly and we will say Γ-subsyntagma; here "implicitly" means that when we write $\Gamma = \prod_i \Gamma_i$ we understand the obvious factorization $\bigoplus_i \Gamma_i$.

Lemma 6.5. *Given a syntagma S and a pattern Γ with a proper factorization, there is at most one, unique, subsyntagma induced by Γ.*

Proof. Consider two syntagmata S', S'' both being Γ-subsyntagmata of S. By definition $\widetilde{\partial}$ is surjective, so that $\widetilde{\partial}(\mathrm{Spt}(S\Gamma)) = \mathrm{Spt}(S')$ and $\widetilde{\partial}(\mathrm{Spt}(S\Gamma)) = \mathrm{Spt}(S'')$, and both supports coincide. By definition $S' \circ \widetilde{\partial} = S$ and $S'' \circ \widetilde{\partial} = S$, so we have the diagram:

$$\begin{tikzcd}
\mathrm{Spt}(S) \cap \Gamma \arrow[rr, "\widetilde{\partial}"] \arrow[dr, "S"'] & & \mathrm{Spt}(S') = \mathrm{Spt}(S'') \arrow[dl, bend right, "S'"'] \arrow[dl, bend left, "S''"] \\
& \Sigma &
\end{tikzcd}$$

Since $\widetilde{\partial}$ is surjective, $S' \circ \widetilde{\partial} = S'' \circ \widetilde{\partial} \implies S' = S''$. Notice that this equality works for S' and S'' on the domain set defined in the diagram, $\mathrm{Spt}(S') = \mathrm{Spt}(S'')$. We have to check in addition that they coincide outside the set. But this is immediate because when $x \notin \mathrm{Spt}(S') = \mathrm{Spt}(S'')$, $S'(x) = S''(x) = 0$. □

Since the Γ-subsyntagma is unique when it exists, we can define:

Definition 6.6. Given a syntagma S and a pattern Γ with a proper factorization, when there exists the Γ-subsyntagma, we will write it as S_Γ.

Now we see that Γ-subsyntagmata are indeed subsyntagmata, and how to calculate them.

Proposition 6.7. *Let S be a syntagma, Γ be a pattern with a proper factorization, and ∂ be its associated deletion. We have the following properties:*

(i) If S_Γ exists, $S_\Gamma \cong S\Gamma$ and then S_Γ is a subsyntagma of S.

(ii) If S_Γ exists, it can be calculated as:

$$S_\Gamma(x) = \begin{cases} S \circ \widetilde{\partial}^{-1}(x) & \textit{if } x \in \widetilde{\partial}(\mathrm{Spt}(S\Gamma)); \\ 0 & \textit{otherwise.} \end{cases}$$

(iii) S_Γ exists if and only if the mapping $\widetilde{\partial} : \mathrm{Spt}(S\Gamma) \longrightarrow \widetilde{\partial}(\mathrm{Spt}(S\Gamma))$ is bijective.

(iv) S_Γ exists if and only if $|\mathrm{Spt}(S\Gamma)| = |\widetilde{\partial}(\mathrm{Spt}(S\Gamma))|$.

Proof. (i) Trivial from the definitions and the fact that $S\Gamma$ is a subsyntagma of S. (ii) We consider the syntagma S' defined by:

$$S'(x) = \begin{cases} S \circ \widetilde{\partial}^{-1}(x) & \text{if } x \in \widetilde{\partial}(\mathrm{Spt}(S\Gamma)); \\ 0 & \text{otherwise.} \end{cases}$$

First we check that S' is well defined. We use the fact that $\widetilde{\partial}$ is an injective mapping over the set $\mathrm{Spt}(S\Gamma)$. The injectivity is a property of a mapping which depends only on the domain. By hypothesis we are assuming that there exists a Γ-subsyntagma. So $\widetilde{\partial}$ defined over $\mathrm{Spt}(S\Gamma) = \mathrm{Spt}(S) \cap \Gamma$ is injective. Hence the expression $S \circ \widetilde{\partial}^{-1}$ makes sense and S' is well defined.

Secondly, we calculate its support. From the definition of S' we have that $\mathrm{Spt}(S) \subseteq \widetilde{\partial}(\mathrm{Spt}(S\Gamma)) = \widetilde{\partial}(\mathrm{Spt}(S) \cap \Gamma)$. The following equalities show that this inclusion is indeed an equality:

$$\begin{aligned} \mathrm{Spt}(S') =& \{x \in \zeta^* \mid S'(x) \neq 0\} \\ =& \{x \in \widetilde{\partial}(\mathrm{Spt}(S\Gamma) \mid S \circ \widetilde{\partial}^{-1}(x) \neq 0\} \\ =& \{x \in \widetilde{\partial}(\mathrm{Spt}(S\Gamma) \mid \widetilde{\partial}^{-1}(x) \in \mathrm{Spt}(S)\} \\ =& \{x \in \widetilde{\partial}(\mathrm{Spt}(S) \cap \Gamma) \mid x \in \widetilde{\partial}(\mathrm{Spt}(S))\} \\ =& \widetilde{\partial}(\mathrm{Spt}(S) \cap \Gamma) \\ =& \widetilde{\partial}(\mathrm{Spt}(S\Gamma)). \end{aligned}$$

Finally, we check that S' satifies the definition of Γ-subsyntagma. On the one hand S' makes the following diagram commute:

$$\begin{tikzcd}
\mathrm{Spt}(S\Gamma) \arrow[rr, "\widetilde{\partial}"] \arrow[dr, "S"'] & & \mathrm{Spt}(S') = \widetilde{\partial}(\mathrm{Spt}(S\Gamma)) \arrow[dl, "S'=S\circ\widetilde{\partial}^{-1}"] \\
& \Sigma &
\end{tikzcd}$$

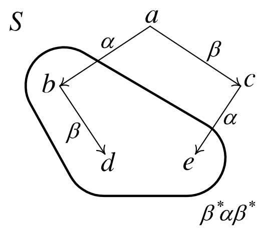
On the other hand $\widetilde{\partial}$ is bijective in this diagram. So S' is a Γ-subsyntagma of S; but since this is unique we conclude $S_\Gamma = S'$.

(iii) If S_Γ exists then $\mathrm{Spt}(S_\Gamma) = \widetilde{\partial}(\mathrm{Spt}(S\Gamma))$ and then the mapping $\widetilde{\partial} : \mathrm{Spt}(S\Gamma) \longrightarrow \widetilde{\partial}(\mathrm{Spt}(S\Gamma))$ is bijective. Let us see the other direction. If we suppose that the last mapping is bijective then we can construct an S' as in (ii) which satisfies the definition of Γ-subsyntagma, therefore $S' = S_\Gamma$ exists.

(iv) This is trivial from (iii) because the sets involved are finite. □

6.4 Examples and Ideal Subsyntagmata

Example 16. A non-induced subsyntagma. Consider the proper pattern $\Gamma = \beta^*\alpha\beta^*$ (with the factorization $\beta^* \oplus \alpha \oplus \beta^*$) and the syntagma S as in the following figure:



We calculate the sets:

$$\begin{aligned}
\Gamma &= \{\alpha, \beta\alpha, \alpha\beta, \beta\alpha\beta, \beta^2\alpha\beta, \beta\alpha\beta^2, \ldots\},\\
\mathrm{Spt}(S) &= \{1, \alpha, \beta, \beta\alpha, \alpha\beta\},\\
\mathrm{Spt}(S\Gamma) &= \{\alpha, \beta\alpha, \alpha\beta\}.
\end{aligned}$$

We are going to calculate the associated deletion $\partial = (\partial_1, \partial_2, \partial_3)$. From the following table:

$\mathrm{Spt}(S\Gamma)$	π^{-1}	$\pi_1 \circ \pi^{-1}$	$\pi_2 \circ \pi^{-1}$	$\pi_3 \circ \pi^{-1}$
α	$(1, \alpha, 1)$	1	α	1
$\beta\alpha$	$(\beta, \alpha, 1)$	β	α	1
$\alpha\beta$	$(1, \alpha, \beta)$	1	α	β

we have that:

$$
\begin{aligned}
|\pi_1 \circ \pi^{-1}(\mathrm{Spt}(S\Gamma))| &= |\{1, \beta\}| = 2 \implies \partial_1 = \mathrm{id};\\
|\pi_2 \circ \pi^{-1}(\mathrm{Spt}(S\Gamma))| &= \ \ |\{\alpha\}| \ \ = 1 \implies \partial_2 = 1;\\
|\pi_3 \circ \pi^{-1}(\mathrm{Spt}(S\Gamma))| &= |\{1, \beta\}| = 2 \implies \partial_3 = \mathrm{id}.
\end{aligned}
$$

So $\partial = (\mathrm{id}, 1, \mathrm{id})$ and then we have $\widetilde{\partial} = \pi' \circ \partial \circ \pi^{-1}$. We check whether $\widetilde{\partial}$ is bijective. On the one hand $\mathrm{Spt}(S\Gamma) = \{\alpha, \beta\alpha, \alpha\beta\}$ and on the other hand $\widetilde{\partial}(\mathrm{Spt}(S\Gamma)) = \{\cancel{\alpha}, \beta\cancel{\alpha}, \cancel{\alpha}\beta\} = \{1, \beta\}$. Thus $\widetilde{\partial}$ is not bijective and S_Γ does not exist. The following table summarizes the calculations:

$\mathrm{Spt}(S\Gamma)$	π^{-1}	$\pi_1 \circ \pi^{-1}$	$\pi_2 \circ \pi^{-1}$	$\pi_3 \circ \pi^{-1}$	
α	$(1, \alpha, 1)$	1	α	1	
$\beta\alpha$	$(\beta, \alpha, 1)$	β	α	1	
$\alpha\beta$	$(1, \alpha, \beta)$	1	α	β	
		$\{1, \beta\}$	$\{\alpha\}$	$\{1, \beta\}$	$\pi_j \circ \pi^{-1}(\mathrm{Spt}(S\Gamma)$
		2	1	2	$\|\pi_j \circ \pi^{-1}(\mathrm{Spt}(S\Gamma)\|$
		id	1	id	∂_j

Example 17. An induced subsyntagma. Consider now the same proper pattern $\Gamma = \beta^*\alpha\beta^*$ (with the factorization $\beta^* \oplus \alpha \oplus \beta^*$) and the syntagma as in Fig. (a). First we calculate $\mathrm{Spt}(S\Gamma)$:

$$
\begin{aligned}
\Gamma &= \{\alpha, \beta\alpha, \alpha\beta, \beta\alpha\beta, \beta^2\alpha\beta, \beta\alpha\beta^2, \ldots\},\\
\mathrm{Spt}(S) &= \{1, \beta, \beta^2, \alpha\beta^2, \beta\alpha\beta^2\},\\
\mathrm{Spt}(S\Gamma) &= \{\alpha\beta^2, \beta\alpha\beta^2\}.
\end{aligned}
$$

Fig. (a)

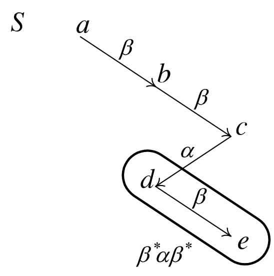
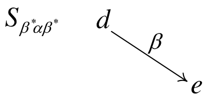


Fig. (b)



We calculate its associated deletion:

$\mathrm{Spt}(S\Gamma)$	π^{-1}	$\pi_1 \circ \pi^{-1}$	$\pi_2 \circ \pi^{-1}$	$\pi_3 \circ \pi^{-1}$	
$\alpha\beta^2$	$(1, \alpha, \beta^2)$	1	α	β^2	
$\beta\alpha\beta^2$	(β, α, β^2)	β	α	β^2	
		$\{1, \beta\}$	$\{\alpha\}$	$\{\beta^2\}$	$\pi_j \circ \pi^{-1}(\mathrm{Spt}(S\Gamma)$
		2	1	1	$\|\pi_j \circ \pi^{-1}(\mathrm{Spt}(S\Gamma)\|$
		id	1	1	∂_j

Then $\partial = (\mathrm{id}, 1, 1)$ and $\widetilde{\partial}\big(\mathrm{Spt}(S\Gamma) = \{\cancel{\alpha\beta^2}, \beta\cancel{\alpha\beta^2}\} = \{1, \beta\}$. Since $|\mathrm{Spt}(S\Gamma)| = |\widetilde{\partial}\big(\mathrm{Spt}(S\Gamma)|$, S_Γ exists; see Fig. (b).

Example 18. Another induced subsyntagma: the subject subsyntagma of a sentence. Consider now a linguistic example from the syntagma in Fig. (a).

Fig. (a)

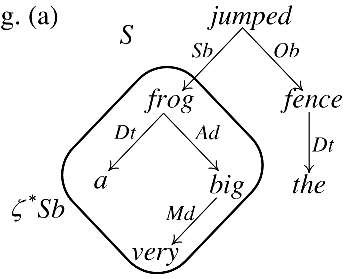
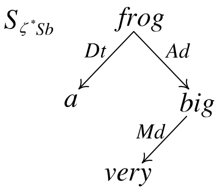


Fig. (b)



We take the pattern $\Gamma = \zeta^* Sb$ with the factorization $\zeta^* \oplus Sb$. We repeat the same calculations:

$$\begin{aligned} \Gamma &= \{Sb, Sb{\cdot}Sb, Ob{\cdot}Sb, Dt{\cdot}Sb, Ad{\cdot}Sb, Sb{\cdot}Sb{\cdot}Sb, Dt{\cdot}Dt{\cdot}Sb, \ldots\}, \\ \mathrm{Spt}(S) &= \{1, Sb, Ob, Dt{\cdot}Sb, Ad{\cdot}Sb, Ad{\cdot}Sb\}, \\ \mathrm{Spt}(S\Gamma) &= \{Sb, Dt{\cdot}Sb, Ad{\cdot}Sb, Md{\cdot}Ad{\cdot}Sb\}. \end{aligned}$$

We calculate its associated deletion:

$\mathrm{Spt}(S\Gamma)$	π^{-1}	$\pi_1 \circ \pi^{-1}$	$\pi_2 \circ \pi^{-1}$	
Sb	$(1, Sb)$	1	Sb	
$Dt{\cdot}Sb$	(Dt, Sb)	Dt	Sb	
$Ad{\cdot}Sb$	(Ad, Sb)	Ad	Sb	
$Md{\cdot}Ad{\cdot}Sb$	$(Md{\cdot}Ad, Sb)$	$Md{\cdot}Ad$	Sb	
		$\{1, Dt, Ad, Md{\cdot}Ad\}$	$\{Sb\}$	$\pi_j \circ \pi^{-1}(\mathrm{Spt}(S\Gamma)$
		4	1	$\|\pi_j \circ \pi^{-1}(\mathrm{Spt}(S\Gamma)\|$
		id	1	∂_j

Then $\partial = (\mathrm{id}, 1)$ and $\widetilde{\partial}\big(\mathrm{Spt}(S\Gamma) = \{\cancel{Sb}, Dt{\cdot}\cancel{Sb}, Ad{\cdot}\cancel{Sb}, Md{\cdot}Ad{\cdot}\cancel{Sb}\} = \{Dt, Ad, Md{\cdot}Ad\}$. Since $|\mathrm{Spt}(S\Gamma)| = |\widetilde{\partial}\big(\mathrm{Spt}(S\Gamma)|$, S_Γ exists; see Fig. (b).

Proposition 6.8. *Let S be a syntagma and Γ a proper pattern such that $|S\Gamma| = 0$; then the induced subsyntagma exists and $S_\Gamma = 0^\bullet$. In particular $0^\bullet_\Gamma = 0^\bullet$.*

Proof. We understand by convention that a mapping $\emptyset \longrightarrow \emptyset$ is indeed bijective. So $\mathrm{Spt}(S\Gamma) = \emptyset$ and then $\widetilde{\partial}(\emptyset) = \emptyset$ whereby:

$$S_\Gamma(x) = \begin{cases} S \circ \widetilde{\partial}^{-1}(x) & \text{if } x \in \emptyset; \\ 0 & \text{otherwise.} \end{cases} = 0. \quad \square$$

When the intersection of the pattern and the syntagma consists in only one locus, the induced syntagma is the atomic syntagma of this locus.

Proposition 6.9. *Let S be a syntagma and Γ a proper pattern such that $|S\Gamma| = 1$; then the induced subsyntagma exists and $S_\Gamma = (S(\varphi))^\bullet$ where $\{\varphi\} = \mathrm{Spt}(S\Gamma)$. In particular $S_\varphi = (S(\varphi))^\bullet$ for any constant $\varphi \in \mathbf{k}^p$.*

Proof. Let $\bigoplus_{i=1}^k \Gamma_i$ be a proper factorization of Γ. We want to calculate the associated deletion $\partial = (\partial_1, \ldots, \partial_k)$. Note that since $|S\Gamma| = |\mathrm{Spt}(S\Gamma)| = 1$, $|\pi_j \circ \pi^{-1}(\mathrm{Spt}(S\Gamma)| = 1$. Thus $\partial = (1, \ldots, 1)$. But then $\widetilde{\partial}(\mathrm{Spt}(S\Gamma)) = \{1\}$. So if $\{\varphi\} = \mathrm{Spt}(S\Gamma)$ then $\widetilde{\partial}(\varphi) = 1$. Thus:

$$S_\Gamma(x) = \begin{cases} S \circ \widetilde{\partial}^{-1}(x) & \text{if } x \in \{1\}; \\ 0 & \text{otherwise.} \end{cases} = \begin{cases} S(\varphi) & \text{if } x \in \{1\}; \\ 0 & \text{otherwise.} \end{cases} = S(\varphi)^\bullet. \quad \square$$

Now we calculate some subsyntagma induced by patterns which will be the most common in the following chapters and which have types: $\mathbf{k}^p$, $\mathbb{M}$, $\mathbf{k}^p\mathbb{M}$, $\mathbb{M}\mathbf{k}^q$, and $\mathbf{k}^p\mathbb{M}\mathbf{k}^q$.

Proposition 6.10. *Let S be a syntagma over ζ and Γ be a pattern such that $|S\Gamma| > 1$. For the patterns $\Gamma = \zeta^*, \xi^*, \zeta^*\psi, \varphi\zeta^*, \varphi\zeta^*\psi, \xi^*\psi, \varphi\xi^*, \varphi\xi^*\psi$, where $\varphi \in \mathbf{k}^p, \psi \in \mathbf{k}^q$ and $\xi \subseteq \zeta$, the following induced subsyntagma exist and we can calculate them as shown:*

(i) $S_{\zeta^*} = S$;

(ii) $S_{\xi^*} = S\xi^*$;

(iii) $S_{\zeta^*\psi}(x) = S(x\psi)$;

(iv) $S_{\varphi\zeta^*}(x) = S(\varphi x)$;

(v) $S_{\varphi\zeta^*\psi}(x) = S(\varphi x\psi)$;

(vi) $S_{\xi^*\psi}(x) = S(x\psi)$, $S_{\varphi\xi^*}(x) = S(\varphi x)$, $S_{\varphi\xi^*\psi}(x) = S(\varphi x\psi)$ *when* $x \in \xi^*$, *otherwise they are* 0.

Proof.

(i) The pattern $\Gamma = \zeta^*$ has the obvious proper factorization with only one factor: ζ^*. Then $\pi = \mathrm{id}$ and $\pi_1 = \mathrm{id}$. Since $|S\Gamma| > 1$:

$$\begin{aligned}
&\Longrightarrow |\pi_1 \circ \pi^{-1}(\mathrm{Spt}(S\Gamma)| = |\mathrm{Spt}(S\Gamma)| > 1\\
&\Longrightarrow \partial = (id)\\
&\Longrightarrow \widetilde{\partial}(\mathrm{Spt}(S\Gamma)) = \mathrm{Spt}(S\Gamma) = \mathrm{Spt}(S) \cap \Gamma = \mathrm{Spt}(S) \cap \zeta^* = \mathrm{Spt}(S).
\end{aligned}$$

Thus

$$S_\Gamma(x) = \begin{cases} S \circ \widetilde{\partial}^{-1}(x) & \text{if } x \in \mathrm{Spt}(S\Gamma);\\ 0 & \text{otherwise.} \end{cases} = \begin{cases} S(x) & \text{if } x \in \mathrm{Spt}(S);\\ 0 & \text{otherwise.} \end{cases} = S(x).$$

(ii) The pattern $\Gamma = \xi^*$ has the proper factorization with only one factor: ξ^*. Then $\pi = \mathrm{id}$ and $\pi_1 = \mathrm{id}$. Since $|S\Gamma| > 1$:

$$\begin{aligned}
&\Longrightarrow |\pi_1 \circ \pi^{-1}(\mathrm{Spt}(S\Gamma)| = |\mathrm{Spt}(S\Gamma)| > 1\\
&\Longrightarrow \partial = (id)\\
&\Longrightarrow \widetilde{\partial}(\mathrm{Spt}(S\Gamma)) = \mathrm{Spt}(S\Gamma).
\end{aligned}$$

Thus

$$S_\Gamma(x) = \begin{cases} S \circ \widetilde{\partial}^{-1}(x) & \text{if } x \in \mathrm{Spt}(S\Gamma);\\ 0 & \text{otherwise.} \end{cases} = \begin{cases} S(x) & \text{if } x \in \mathrm{Spt}(S\Gamma);\\ 0 & \text{otherwise.} \end{cases} = S\Gamma(x).$$

(iii) This is a particular case of (v) when $\varphi = 1$.

(iv) This is a particular case of (v) when $\psi = 1$.

(v) Let the pattern be $\Gamma = \varphi\zeta^*\psi$. Suppose $\varphi = \varphi_1 \cdots \varphi_p$ and $\psi = \psi_1 \cdots \psi_q$. Then we consider the proper factorization:

$$\bigoplus_{i=1}^{p} \varphi_i \oplus \zeta^* \oplus \bigoplus_{i=1}^{q} \psi_i.$$

So ∂ has the form $\partial = (\partial_1, \ldots, \partial_p, \partial_{p+1}, \partial_{p+2}, \ldots, \partial_{p+q+2})$. We have the following cardinalities:

$$|\pi_j \circ \pi^{-1}(\mathrm{Spt}(S\Gamma))| \text{ is } \begin{cases} = 1 & \text{if } 1 \le j \le p;\\ > 1 & \text{if } j = p+1;\\ = 1 & \text{if } p+2 \le j \le q. \end{cases}$$

Let us see this in a little more detail. When $1 \leq j \leq p$ then $\pi_j \circ \pi^{-1}(\mathrm{Spt}(S\Gamma)) = \{\varphi_j\}$. Equally when $p+2 \leq j \leq q$, $\pi_j \circ \pi^{-1}(\mathrm{Spt}(S\Gamma)) = \{\psi_j\}$. So

$$|\pi_j \circ \pi^{-1}(\mathrm{Spt}(S\Gamma))| \;=\; \begin{cases} 1 & \text{if } 1 \leq j \leq p; \\ 1 & \text{if } p+2 \leq j \leq q. \end{cases}$$

Now we see $|\pi_{p+1} \circ \pi^{-1}(\mathrm{Spt}(S\Gamma))|$. We appeal to an easy general set-theoretical fact. If we have finite sets such that $A \subseteq A_1 \times \cdots \times A_k$ then $|A| \leq |\pi_1(A)| \cdots |\pi_k(A)|$ (this is a consequence of $A \subseteq \pi_1(A) \times \cdots \times \pi_k(A)$). Applying this we have:

$$\begin{aligned} 1 < |\mathrm{Spt}(S\Gamma)| &= |\pi^{-1}(\mathrm{Spt}(S\Gamma))| \\ &\leq |\pi_1 \circ \pi^{-1}(\mathrm{Spt}(S\Gamma))| \cdots |\pi_k \circ \pi^{-1}(\mathrm{Spt}(S\Gamma))| \\ &= 1 \cdots 1 \cdot |\pi_{p+1} \circ \pi^{-1}(\mathrm{Spt}(S\Gamma))| \cdot 1 \cdots 1 = |\pi_{p+1} \circ \pi^{-1}(\mathrm{Spt}(S\Gamma))|. \end{aligned}$$

The first inequality is given by assumption in the proposition we are proving. So we have:

$$\partial = (1, \ldots, 1, \mathrm{id}, 1, \ldots, 1).$$

Notice that every element in $S\Gamma$ can be written as $\varphi x \psi$ and then $\widetilde{\partial}(\varphi x \psi) = x$ which gives the bijection $\widetilde{\partial} : \mathrm{Spt}(S\Gamma) \longrightarrow \widetilde{\partial}(\mathrm{Spt}(S\Gamma))$. Hence S_Γ exists. We calculate it:

$$\begin{aligned} S_\Gamma(x) &= \begin{cases} S \circ \widetilde{\partial}^{-1}(x) & \text{if } x \in \widetilde{\partial}(\mathrm{Spt}(S\Gamma)); \\ 0 & \text{otherwise.} \end{cases} \\ &= \begin{cases} S(\varphi x \psi) & \text{if } \varphi x \psi \in \mathrm{Spt}(S); \\ 0 & \text{otherwise.} \end{cases} \;=\; S(\varphi x \psi). \end{aligned}$$ □

(vi) We combine (ii) and (v).

Example 19. Co-subtrees and chains of subjects in Germanic languages. When a pattern takes a simple form such as ξ^*, φ or $\zeta^*\varphi$ then the induced subsyntagma has an immediate geometric interpretation. The pattern ξ^* only takes the part of the syntagma in ξ^* and forget the rest; φ takes just a locus; and $\zeta^*\varphi$ takes a subtree.

Nevertheless in the case of left ideals $\varphi\zeta^*$ the loci in $\zeta^*\varphi \cap \mathrm{Spt}(S)$ form a disconnected set. We call this distribution of loci a *co-subtree*; in Fig. 6.2(b) there is an example of a co-subtree with $\zeta = \{\alpha, \beta\}$. From the previous proposition the induced subsyntagma is given by $S_{\varphi\zeta^*}(x) = S(\varphi x)$. These patterns are useful to understand linearization in Germanic languages. These languages display verbal clusters and their arguments in very particular configurations.

We will see in the next chapter that context-freeness and right ideals are closely related. Since Shieber argued that Germanic cross-serial dependencies are not context-free (cf. §4.3.1), we should not be restricted to using right ideals. Indeed, the adequate patterns to treat the subject arguments are left ideals. Consider the subordinate clause in Dutch. In the following example the sentence exhibits an object argument in the deepest clause, *de nijlpaarden* (compare with the Dutch sentence in §4.3.1), which makes the analysis more complex:

(19)	. . . omdat	ik	Cecilia	henk	de	nijlpaarden	zag	helpen	voeren.
	. . . because	I	Cecilia	Henk	the	hippopotamuses	saw	help	feed
	'. . . because I saw Cecilia help Henk feed the hippos.'								

Fig. (a)

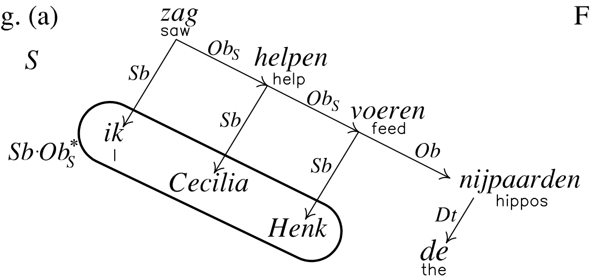


Fig. (b)

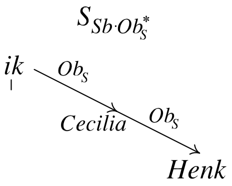


Fig. (a) shows the dependency analysis. When we consider the pattern $Sb \cdot Ob_S^*$ (where recall that Ob_S is simply a syntactic function which introduces a subsyntagma representing a subordinate clause which plays the role of a propositional object) we are grouping all the loci which end with a subject, or simply put we are grouping the subjects. Now consider the induced subsyntagma $S_{Sb \cdot Ob_S^*}$ in Fig. (b). The subjects are linked by arrows in its graphical representation, which did not exist in S. These new connections will be the key to giving a recursive system to linearize the syntagma.

Notice that co-subtrees are the dual form of subtrees. That is, a subtree in S becomes a co-subtree in $S^{\mathfrak{R}}$. In the linguistic framework co-subtrees are the alternative form for Germanic languages to linearize. However these languages use patterns even more complex than left ideals. When the subjects of the subordinated clauses contain modifiers we have to expand the pattern to $\zeta^* \cdot Sb \cdot Ob_S^*$.

6.5 Symmetric Induced Subsyntagma

We have seen that symmetry makes manifolds commute with their patterns (cf. Lema 5.6). Similarly we have the following result; this lemma is necessary to understand how the duality of symmetry extends throughout the components in our formalism.

Lemma 6.11. *Let S be a syntagma, and Γ be a pattern. Suppose that S_Γ exists, then $S^{\mathfrak{R}}_{\Gamma^R}$ exists, and we have:*

$$(S_\Gamma)^{\mathfrak{R}} = S^{\mathfrak{R}}_{\Gamma^R}.$$

Proof. Recall that we defined the symmetric syntagmata as $S^{\mathfrak{R}}(x) = S(x^R)$. For reasons of readability we will also use sometimes the notation $R(x) = x^R$. Even though R is an involutive mapping, and therefore $R^{-1} = R$, we keep the exponent -1 to indicate the direction of the arrows in the following diagram the mappings of which will be introduced during the proof:

$$\begin{tikzcd}
 & \bigoplus_{i=1}^k \Gamma^R_{k-i} \arrow[rr, "\partial^{\mathfrak{R}}"] \arrow[dd, "\pi^{\mathfrak{R}}"] & & \bigoplus_{i=1}^k \partial_{k-i}(\Gamma^R_{k-i}) \arrow[dd, "\widetilde{\pi^{\mathfrak{R}}}"] \\
\bigoplus_{i=1}^k \Gamma_i \arrow[ur, "\overline{R}"] \arrow[rr, "\partial"] \arrow[dd, "\pi"] & & \bigoplus_{i=1}^k \partial_i(\Gamma_i) \arrow[ur, "\overline{R}"] \arrow[dd, "\pi"] & \\
 & \Gamma^R \arrow[rr, "\widetilde{\partial^{\mathfrak{R}}}"] & & \widetilde{\partial^{\mathfrak{R}}}(\Gamma^R) \\
\Gamma \arrow[ur, "R"] \arrow[rr, "\widetilde{\partial}"] & & \widetilde{\partial}(\Gamma) \arrow[ur, "R"] &
\end{tikzcd}$$

The first goal is proving that this commutes (steps (1)-(6)):

(1) If $\pi : \bigoplus_{i=1}^k \Gamma_i \longrightarrow \Gamma$ is a proper factorization then $\pi^{\mathfrak{R}} : \bigoplus_{i=1}^k \Gamma^R_{k-i} \longrightarrow \Gamma^R$ is a proper factorization.

We define the mapping $\overline{R}(x_1, \ldots, x_k) = (x_k^R, \ldots, x_1^R)$ which is clearly bijective. Now we prove that $\pi \circ \overline{R}^{-1} = R^{-1} \circ \pi^{\mathfrak{R}}$:

$$\pi \circ \overline{R}^{-1}(x_k, \ldots, x_1) = \pi(x_1^R, \ldots, x_k^R) = x_1^R \cdots x_k^R,$$

$$R^{-1} \circ \pi^{\mathfrak{R}}(x_k, \cdots, x_1) = R^{-1}(x_k \cdots x_1) = x_1^R \cdots x_k^R.$$

Then $\pi \circ \overline{R}^{-1} = R^{-1} \circ \pi^{\mathfrak{R}} \implies \pi^{\mathfrak{R}} = R \circ \pi \circ \overline{R}^{-1}$. Hence if π is bijective then $\pi^{\mathfrak{R}}$ is too.

(2) We know that if Γ is a basic pattern then $\Gamma^R = \Gamma$. By (1) if $\bigoplus_{i=1}^k \Gamma_i$ is a proper factorization of Γ then $\bigoplus_{i=1}^k \Gamma_{k-i}$ is a proper factorization of Γ^R.

(3) $(S\Gamma)^{\mathfrak{R}} = S^{\mathfrak{R}}\Gamma^R$.

$$(S\Gamma)^{\mathfrak{R}}(x) \;=\; S\Gamma(x^R) = \begin{cases} S(x^R) & \text{if } x^R \in \Gamma; \\ 0 & \text{otherwise.} \end{cases} = \begin{cases} S^{\mathfrak{R}}(x) & \text{if } x \in \Gamma^R; \\ 0 & \text{otherwise.} \end{cases} = S^{\mathfrak{R}}\Gamma^R.$$

(4) If $\partial = (\partial_1, \ldots, \partial_k)$ is the deletion associated with S and Γ, and if $\partial^{\mathfrak{R}} = (\partial_1^{\mathfrak{R}}, \ldots, \partial_k^{\mathfrak{R}})$ is the deletion associated with $S^{\mathfrak{R}}$ and Γ^R then we have that $\partial^{\mathfrak{R}} = (\partial_k, \ldots, \partial_1)$.

First we define $\pi_j^{\mathfrak{R}}$ as the j-projection $\pi_j^{\mathfrak{R}} : \bigoplus_{i=1}^k \Gamma_{k-i} \longrightarrow \Gamma_j$, $\pi_j^{\mathfrak{R}}(x_k, \ldots, x_1) = x_{k-j}$. Now we check the following identity:

$$\pi_j \circ \pi^{-1} = R \circ \pi_{k-j}^{\mathfrak{R}} \circ (\pi^{\mathfrak{R}})^{-1} \circ R$$

We see that:

$$\pi_j \circ \pi^{-1}(x) = \pi_j(x_1, \ldots, x_k) = x_j,$$

and that:

$$\begin{aligned} R \circ \pi_{k-j}^{\mathfrak{R}} \circ (\pi^{\mathfrak{R}})^{-1} \circ R(x) &= R \circ \pi_{k-j}^{\mathfrak{R}} \circ (\pi^{\mathfrak{R}})^{-1}(x^R) \\ &= R \circ \pi_{k-j}^{\mathfrak{R}}(x_k^R, \ldots, x_1^R) = R(x_j^R) = x_j. \end{aligned}$$

Now we prove that $R\Big(\pi_j \circ \pi^{-1}(\mathrm{Spt}(S\Gamma))\Big) = \pi_j^{\mathfrak{R}} \circ (\pi^{\mathfrak{R}})^{-1}\Big(\mathrm{Spt}(S^{\mathfrak{R}}\Gamma^R)\Big)$:

$$\begin{aligned} R\Big(\pi_j \circ \pi^{-1}(\mathrm{Spt}(S\Gamma))\Big) &= \pi_j^{\mathfrak{R}} \circ (\pi^{\mathfrak{R}})^{-1}\Big((\mathrm{Spt}(S\Gamma)^{\mathfrak{R}}\Big) \\ &= \pi_j^{\mathfrak{R}} \circ (\pi^{\mathfrak{R}})^{-1}\Big(\mathrm{Spt}((S\Gamma)^{\mathfrak{R}}\Big) \\ &= \pi_j^{\mathfrak{R}} \circ (\pi^{\mathfrak{R}})^{-1}\Big(\mathrm{Spt}(S^{\mathfrak{R}}\Gamma^R)\Big). \end{aligned}$$

We take cardinalities:

$$\begin{aligned} &\Big|R\Big(\pi_j \circ \pi^{-1}(\mathrm{Spt}(S\Gamma))\Big)\Big| = \Big|\pi_j^{\mathfrak{R}} \circ (\pi^{\mathfrak{R}})^{-1}\Big(\mathrm{Spt}(S^{\mathfrak{R}}\Gamma^{\mathfrak{R}})\Big)\Big| \\ \Longleftrightarrow\ &\big|\pi_j \circ \pi^{-1}(\mathrm{Spt}(S\Gamma))\big| = \Big|\pi_j^{\mathfrak{R}} \circ (\pi^{\mathfrak{R}})^{-1}\Big(\mathrm{Spt}(S^{\mathfrak{R}}\Gamma^R)\Big)\Big|. \end{aligned}$$

Recall that we defined ∂_j by:

$$\partial_j \text{ is the constant } 1 \iff \big|\pi_j \circ \pi^{-1}(\mathrm{Spt}(S\Gamma))\big| \leq 1,$$

otherwise ∂_j is the identity. Consider the deletion associated with $S^{\mathfrak{R}}$ and Γ^R, namely, $\partial^{\mathfrak{R}} = (\partial_1^{\mathfrak{R}}, \ldots, \partial_k^{\mathfrak{R}})$. If we apply this definition to this deletion we have

$$\partial_j^{\mathfrak{R}} \text{ is the constant } 1 \iff \Big|\pi_j^{\mathfrak{R}} \circ (\pi^{\mathfrak{R}})^{-1}(\mathrm{Spt}(S^{\mathfrak{R}}\Gamma^R))\Big| \leq 1.$$

But we have proved that sets involved in the definition of ∂_j and $\partial_{k-j}^{\mathfrak{R}}$ have the same cardinality. Thus $\partial_j = \partial_{k-j}^{\mathfrak{R}}$ and $\partial^{\mathfrak{R}} = (\partial_k, \ldots, \partial_1)$.

(5) $\partial = \overline{R}^{-1} \circ \partial^{\mathfrak{R}} \circ \overline{R}$.

First we need to prove that $\partial_i(x_i^R) = \partial_i(x_i)^R$ for any $i = 1, \dots, k$. But this is easy because ∂_i is the identity mapping or the constant mapping 1 and in both cases it commutes with R. Then we have:

$$\begin{aligned}
\overline{R}^{-1} \circ \partial^{\mathfrak{R}} \circ \overline{R}(x_1, \dots, x_k) &= \overline{R}^{-1} \circ \partial^{\mathfrak{R}}(x_k^R, \dots, x_1^R) \\
&= \overline{R}^{-1}\Big(\big(\partial_k(x_k^R), \dots, \partial_1(x_1^R)\big)\Big) \\
&= \overline{R}^{-1}\Big(\big(\partial_k(x_k)^R, \dots, \partial_1(x_1)^R\big)\Big) \\
&= \big((\partial_1(x_1)^R)^R, \dots, (\partial_k(x_k)^R)^R\big) \\
&= \big(\partial_1(x_1), \dots, \partial_k(x_k)\big) \\
&= \partial(x_1, \dots, x_k),
\end{aligned}$$

where we have used part (5) to reverse the vector.

(6) $\widetilde{\partial} = R^{-1} \circ \widetilde{\partial^{\mathfrak{R}}} \circ R$.

By definition $\widetilde{\partial} = \widetilde{\pi} \circ \partial \circ \pi^{-1}$ and $\widetilde{\partial^{\mathfrak{R}}} = \widetilde{\pi^{\mathfrak{R}}} \circ \partial^{\mathfrak{R}} \circ (\pi^{\mathfrak{R}})^{-1}$. We also need the commutations $\overline{R} \circ \pi^{-1} = (\pi^{\mathfrak{R}})^{-1} \circ R$ and $\widetilde{\pi} \circ \overline{R}^{-1} = R^{-1} \circ \widetilde{\pi^{\mathfrak{R}}}$.

The first arises from inverting $\pi \circ \overline{R}^{-1} = R^{-1} \circ \pi^{\mathfrak{R}}$ which was proved in part (1). The second is very similar (we only substitute $\widetilde{\pi}$ for π). Then if we combine these and we use part (5) we will obtain:

$$\begin{aligned}
\widetilde{\partial} &= \widetilde{\pi} \circ \partial \circ \pi^{-1} \\
&= \widetilde{\pi} \circ \left(\overline{R}^{-1} \circ \partial^{\mathfrak{R}} \circ \overline{R}\right) \circ \pi^{-1} \\
&= \left(\widetilde{\pi} \circ \overline{R}^{-1}\right) \circ \partial^{\mathfrak{R}} \circ \left(\overline{R} \circ \pi^{-1}\right) \\
&= \left(R^{-1} \circ \widetilde{\pi^{\mathfrak{R}}}\right) \circ \partial^{\mathfrak{R}} \circ \left((\pi^{\mathfrak{R}})^{-1} \circ R\right) \\
&= R^{-1} \circ \left(\widetilde{\pi^{\mathfrak{R}}} \circ \partial^{\mathfrak{R}} \circ (\pi^{\mathfrak{R}})^{-1}\right) \circ R \\
&= R^{-1} \circ \widetilde{\partial^{\mathfrak{R}}} \circ R.
\end{aligned}$$

Notice that parts (1) to (6) prove that the whole cube commutes.

(7) If $S^{\mathfrak{R}}_{\Gamma^R}$ exists we have the isomorphism $S^{\mathfrak{R}}\Gamma^R \cong_{\widetilde{\partial^{\mathfrak{R}}}} S^{\mathfrak{R}}_{\Gamma^R}$.

There is nothing to prove: this is just the definition of the subsyntagma of $S^{\mathfrak{R}}$ induced by Γ^R.

(8) The functor $\mathfrak{R}$ preserves isomorphisms of syntagmata. So if $S\Gamma \cong_{\widetilde{\partial}} S_\Gamma$ then we have the isomorphism $(S\Gamma)^{\mathfrak{R}} \cong_{R\circ\widetilde{\partial}\circ R^{-1}} (S_\Gamma)^{\mathfrak{R}}$.

We saw this in Lemma 5.5.

(9) $S^{\mathfrak{R}}\Gamma^R \cong_{\widetilde{\partial^{\mathfrak{R}}}} (S_\Gamma)^{\mathfrak{R}}$.

By part (3) we have $S^{\mathfrak{R}}\Gamma^R = (S\Gamma)^{\mathfrak{R}}$ so we have by part (8) that $S^{\mathfrak{R}}\Gamma^R \cong_{R\circ\widetilde{\partial}\circ R^{-1}} (S_\Gamma)^{\mathfrak{R}}$. By part (6) we have $S^{\mathfrak{R}}\Gamma^R \cong_{\widetilde{\partial^{\mathfrak{R}}}} (S_\Gamma)^{\mathfrak{R}}$.

Now we have all the ingredients. We see that we have two isomorphisms. The first, by part (7), $S^{\mathfrak{R}}\Gamma^R \cong_{\widetilde{\partial^{\mathfrak{R}}}} S^{\mathfrak{R}}_{\Gamma^R}$. The second is given by part (9), $S^{\mathfrak{R}}\Gamma^R \cong_{\widetilde{\partial^{\mathfrak{R}}}} (S_\Gamma)^{\mathfrak{R}}$. So by the definition of induced syntagmata we have two candidates to be subsyntagma. However the induced subsyntagma is unique and thus $S^{\mathfrak{R}}_{\Gamma^R} = (S_\Gamma)^{\mathfrak{R}}$. Now we prove the existence. If S_Γ exists then $(S_\Gamma)^{\mathfrak{R}}$ exists always. By the last equality $S^{\mathfrak{R}}_{\Gamma^R}$ exists. □

7

The Model and the Bi-Hierarchies

The present chapter represents the core of the thesis. We introduce the model of *algebraic dependency grammar* in terms of the concepts developed in the previous chapters. An algebraic dependency grammar consists in a manifold (already discussed in Chapter 4) and a global linearization given by a system of *arrangements*. Two methodological considerations are made before the formalization.

First we notice that the classical notion of *projectivity* is quintessentially that certain substructures of a dependency tree always form an interval in its linearization. So we have to establish well what is a substructure. We saw in the last chapter that patterns proportion the key: a *subsyntagma* is a substructure of a syntagma which is induced by a pattern. This generalizes the notion of projectivity. By decomposing a syntagma in subsyntagmata and specifying how these subsyntagmata must be ordered, and repeating this action for each subsyntagma, we establish recursive global linearization procedures.

Notice that patterns sustain both manifolds and linearizations. We study their interrelation. We can classify languages by a pair of descriptions X/Y in a classificatory monoid; one for the manifold X, and another for the linearization Y. Since we have two classificatory axes, we call this a *bi-hierarchy* which depends on the classificatory monoid. We see several initial examples.

7.1 Two Methodological Considerations

7.1.1 Extending the Monoidal Hypothesis (Individual Linearizations)

An example of so-called *discontinuity* is the possibility of extracting a noun complement and locating it to the right. This construction is named *extraposition* and it will be revisited when we review the applications of the theory in future chapters. Consider the pair of sentences: [1]

(20) a. A hearing on the issue is scheduled today.
b. A hearing is scheduled on the issue today.

[1] Example from Kuhlmann (2013).

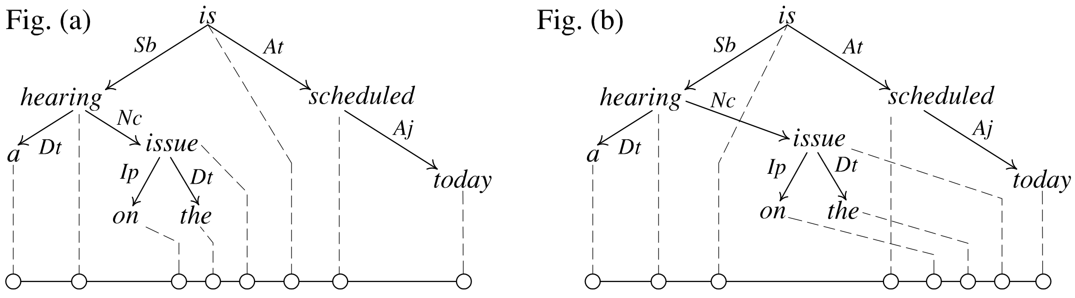


Figure 7.1: (a) projective individual linearization for the sentence *a hearing on the issue is scheduled today*; (b) non-projective individual linearization for the sentence *a hearing is scheduled on the issue today*.

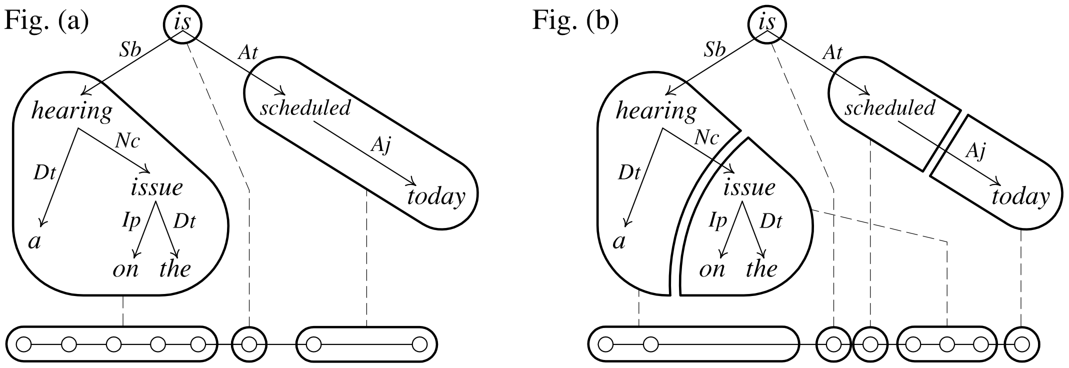


Figure 7.2: the same linearizations as in Figure 7.1 with attention to the substructures rather than on the words.

The first is the non-problematic version and it enjoys a projective linearization. The second linearization is not a projection due to the metric property for projections; see Example 15. Traditionally in this situation it is said that a constituent (*on the issue*) has been "moved" to the right in the sentence. See Figures 7.1(a) and (b).

Instead of looking where the linearization places each word, let us focus on how the linearization deals with bigger substructures. We focus just on the biggest proper parts of the structures. Fig. 7.2(a) and Fig. 7.2(b) show the effect of the linearization on these parts. It appears that the substructures in the second figure are finer than in the first and that, thus, the speaker has a finer control on the parts of syntactic structure. This is an interpretation without appeal to "movement".

On this perspective we can say that individual linearizations transform certain substructures into intervals under a generalization of projectivity. We have defined such a general concept of substructure induced by a pattern in the last chapter.

This leads us to an extension of the *monoidal hypothesis*, whereby let us recall what we stated after studying a number of natural language examples: *for any natural language the set of the places where agreements occur can be described as a monoidal* *pattern.* Now we would like to extend this hypothesis:

> *Claim* 3. MONOIDAL PATTERN HYPOTHESIS (CONTINUATION). (...) and furthermore natural languages in their linearization decompose the structures in certain substructures also described by monoidal patterns which are transformed into prosodic intervals.

Nevertheless, this hypothesis, which only tries to capture the underlying notion, does not inform as how these "certain" structures must be chosen. This is a consideration on linearizations taken individually and it says nothing about how a global linearization is orchestrated. Next section starts out on the way to do this.

7.1.2 Recurrences on Syntagmata (Global Linearizations)

Let us make some mathematical considerations on recurrences. A typical recurrence definition of a function is: $n! = n \cdot (n-1)!$, when $n > 0$, and $n! = 1$, when $n = 0$. The case $n = 0$ is called the *base case* which starts the recurrence. More than one equation and variable can be used in a recurrence:

$$\begin{cases} X_n = Y_{n-1} + X_{n-1} & \text{if } n > 0; \\ Y_n = X_{n-1} - Y_{n-1} & \text{if } n > 0; \\ X_n = 1 & \text{if } n = 0; \\ Y_n = 0 & \text{if } n = 0. \end{cases}$$

The first values of this pair of sequences are $X_n = 1, 1, 2, 2, 4, 4, 8, 8, \ldots$ and $Y_n = 0, 1, 0, 2, 0, 4, 0, 8, \ldots$.

In a similar way we can describe recursively a mapping Π over syntagma. Consider syntagmata **Synt** with syntactic functions $\zeta = \{\alpha, \beta\}$. The following example defines a mapping over syntagmata into natural numbers, $\Pi : \mathbf{Synt} \longrightarrow \mathbb{N}_+$ given by the recurrences:

$$\begin{cases} \Pi(S) = \Pi(S_{\zeta^*\alpha}) + \Pi(S_{\zeta^*\beta}) & \text{if } S \text{ is not atomic;} \\ \Pi(S) = 1 & \text{if } S \text{ is atomic;} \\ \Pi(S) = 0 & \text{if } S \text{ is the null syntagma.} \end{cases}$$

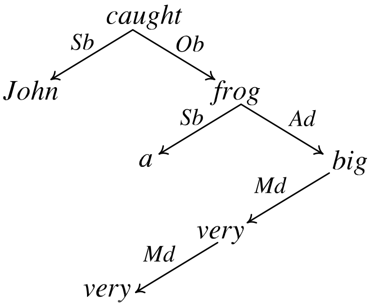
This mapping is calculating $\Pi(S) = |\mathrm{Lvs}(S)|$. Notice that the base case is given by both atomic syntagmata and the null syntagma. Let us see now a mapping $\Pi : \mathbf{Synt} \longrightarrow \Sigma^*$:

$$\begin{cases} \Pi(S) = a + b + \Pi(S_{\zeta^*\alpha}) & \text{if } S \text{ is not atomic;} \\ \Pi(S) = c & \text{if } S \text{ is atomic;} \\ \Pi(S) = c & \text{if } S \text{ is the null syntagma.} \end{cases}$$

The result of apply it to a syntagma S is $\Pi(S) = a + b + \cdots + a + b + c$. The image of this application $\Pi(\mathbf{Synt})$ is the language $\{(ab)^n c \mid n \in \mathbb{N}_+\}$, where n is the number of subtrees $\zeta^*\alpha^k$ successively nested. We can use recurrence equations in order to define a global linearization Π. We need secondary auxiliary functions, Π', Π'':

$$\begin{cases} \Pi(S) = \Pi'(S_{\zeta^*Sb}) + \Pi'(S_1) + \Pi'(S_{\zeta^*Ob}) & \text{if } S \text{ is not atomic;} \\ \Pi'(S) = \Pi''(S_{\zeta^*Dt}) + \Pi''(S_{\zeta^*Ad}) + \Pi''(S_1) & \text{if } S \text{ is not atomic;} \\ \Pi''(S) = \Pi''(S_{\zeta^*Md}) + \Pi''(S_1) & \text{if } S \text{ is not atomic;} \\ \Pi(S), \Pi'(S), \Pi''(S) = a & \text{if } S = a^\bullet \text{ is atomic or null;} \end{cases}$$

Notice that the right side of the recurrence equations do not add nor substract material of the syntagma. They just split a syntagma and reorder its parts. Let us see an example of calculation from the following syntagma S.



$$\begin{aligned} \Pi(S) &= \Pi'(\zeta^*Sb) + \text{proved} + \Pi'(\zeta^*Ob) \\ &= (\Pi''(\zeta^*Dt) + \Pi''(\zeta^*Ad) + \text{John}) + \text{proved} \ + \Pi'(\zeta^*Ob) \\ &= (0 + 0 + \text{John}) + \text{proved} \ + \Pi'(\zeta^*Ob) \\ &= \text{John} + \text{proved} + (\Pi''(\zeta^*Dt) + \Pi''(\zeta^*Ad) + \text{theorem}) \\ &= \text{John} + \text{proved} + (\text{a} + (\Pi''(\zeta^*Md) + \text{hard}) + \text{theorem}) \\ &= \text{John} + \text{proved} + \text{a} + (\Pi''(\zeta^*Md) + \text{very}) + \text{hard} + \text{theorem} \\ &= \text{John} + \text{proved} + \text{a} + (\Pi''(\zeta^*Md) + \text{very}) + \text{very} + \text{hard} + \text{theorem} \\ &= \text{John} + \text{proved} + \text{a} + 0 + \text{very} + \text{very} + \text{hard} + \text{theorem} \\ &= \text{John} + \text{proved} + \text{a} + \text{very} + \text{very} + \text{hard} + \text{theorem.} \end{aligned}$$

In order to avoid subsyntagmata of subsyntagmata, e.g. $(S_{\zeta^*Sb})_{\zeta^*Dt}$, we have just written the last pattern involved in the computation, $\Pi(\zeta^*Dt)$. Notice that the recurrences correspond with the intuitive order arrangements that one can read in a traditional grammar:

$$\begin{aligned} \Pi &\leftrightarrow \textbf{\textit{Subject + Verb + Object}}; \\ \Pi' &\leftrightarrow \textbf{\textit{Determiner + Adjective + Noun}}; \\ \Pi'' &\leftrightarrow \textbf{\textit{Modifier + Adjective}}. \end{aligned}$$

This recurrence uses right ideal patterns (which represent subtrees). However, in the last chapter we have defined a rich zoo of substructures and we can use them. A recurrence system as in the last example defines a global linearization $\Pi : W \longrightarrow \Sigma^*$. If we fix a manifold and such a recurrence system we will have a grammar.

However, in natural languages, linearizations are not mappings, but relations. In dependency grammar it is widely admitted that the set of sentences:[2]

(21) a. Fortunately, the man left the room.
b. The man, fortunately, left the room.
c. The man left, fortunately, the room.
d. The man left the room, fortunately.

must come from a unique syntactic structure. So instead of a mapping $\Pi : \textbf{Synt} \longrightarrow \Sigma^*$ defined by recurrences we have to deal with relations defined by recurrences, or what amounts to the same thing, a mapping $\Pi : \textbf{Synt} \longrightarrow \wp(\Sigma^*)$. Notice that the essence of the above recurrences resides in indicating in the right part of the equations which is the new equation to apply or invoke for each subsyntagma. Fig. 7.3(a) shows a scheme of invocations for the last recurrence equations. An arrow sourcing from a pattern indicates which equation must follow. In the more general case with $\Pi : \textbf{Synt} \longrightarrow \wp(\Sigma^*)$, we must allow invocation of several equations.

By way of a minimal example of relational recurrence consider shifting movement on certain adjectives in Romance languages:

(22) Un magnífic espectacle / un espectacle magnífic
a great play / a play great
'a great play'

This can be captured by a "non-deterministic" diagram, see Fig. 7.3(b).[3]

So we can summarize all this as:

> *Claim* 4. Monoidal Pattern Hypothesis (continuation). (...) This linearization is organized by a finite system of recurrences or word arrangements. Every recurrence tells how to order the substructures and how to decompose these further.

What follows is a formalization of these ideas.

[2] These constructions are called *parentheticals*; the example is from Morrill (2011).

[3] The example is in Catalan. The possibility of shifting adjectives in Romance languages does not involve every adjective. Just a few adjectives can be placed in both positions (*prenominal, postnominal*) without changing the meaning of the phrase or even making it incorrect. Thus, to be more exact we should use a specific function *Ad′* similar to the ordinary adjective *Ad* which only would select those few adjectives.

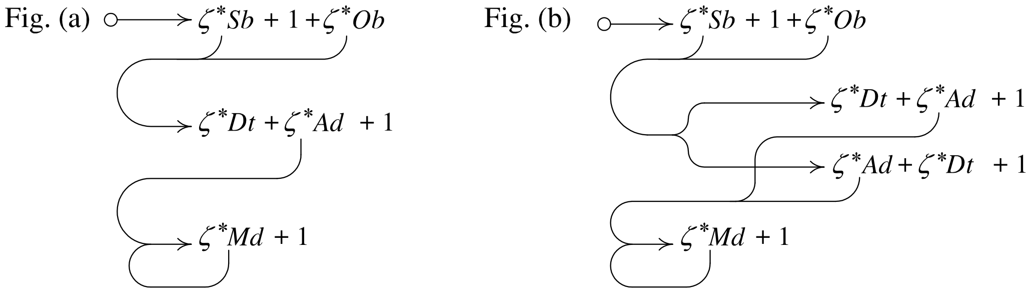


Fig. (a)
$\zeta^*Sb + 1 + \zeta^*Ob$
$\zeta^*Dt + \zeta^*Ad + 1$
$\zeta^*Md + 1$
Fig. (b)
$\zeta^*Sb + 1 + \zeta^*Ob$
$\zeta^*Dt + \zeta^*Ad + 1$
$\zeta^*Ad + \zeta^*Dt + 1$
$\zeta^*Md + 1$

Figure 7.3: (a) and (b) diagrams invoking equations of a recurrence.

7.2 Algebraic Dependency Grammar

Definition 7.1. Given a finite set of patterns $\Gamma_1, \ldots, \Gamma_m$ we say that it forms a *partition* of the syntagma S iff $\mathrm{Spt}(S\Gamma_1), \ldots, \mathrm{Spt}(S\Gamma_m)$ is a partition of $\mathrm{Spt}(S)$, i.e:

(i) $\mathrm{Spt}(S) = \bigcup_{i=1}^{m} \mathrm{Spt}(S\Gamma_i)$;

(ii) if $\mathrm{Spt}(S\Gamma_i) \neq \mathrm{Spt}(S\Gamma_j)$ then $\mathrm{Spt}(S\Gamma_i) \cap \mathrm{Spt}(S\Gamma_j) = \emptyset$, for every $i, j = 1, \ldots, m$.

Definition 7.2. We say that the set of patterns $\Gamma_1, \ldots, \Gamma_m$ *is applicable to* a syntagma S when:

(i) $\Gamma_1, \ldots, \Gamma_m$ form a partition of S;

(ii) the induced subsyntagma S_{Γ_i} exists, for every $i = 1, \ldots, m$;

(iii) $|S_{\Gamma_i}| < |S|$, for every $i = 1, \ldots, m$, provided that S is not atomic, nor null.

Given a mapping $\Pi : \mathbf{Synt} \longrightarrow \wp(\Sigma^*)$ we notate its set extension by the same Π, i.e. $\Pi(\{S_1, \ldots, S_k, \ldots\}) = \{\Pi(S_1), \ldots, \Pi(S_k), \ldots\}$. Given a set of mappings $\Pi_1, \ldots, \Pi_n$, and $\omega \subseteq \{1, \ldots, n\}$, we notate $\Pi_\omega(X) = \bigcup_{i \in \omega} \Pi_i(X)$.

Definition 7.3. Let there be a finite set of patterns doubly indexed thus:

$$\begin{cases} \Gamma_{1,1}, \ldots, \Gamma_{m_1,1}, \\ \Gamma_{1,2}, \ldots, \Gamma_{m_2,2}, \\ \qquad \vdots \\ \Gamma_{1,n}, \ldots, \Gamma_{m_n,n}; \end{cases}$$

with $n \geq 1$ and $m_1, \ldots, m_n \geq 1$. Consider a mapping:

$$\omega : \{(i, j) \,|\, i = 1, \ldots, m_j, j = 1, \ldots, n\} \cup \{(0, 0)\} \longrightarrow \wp(\{1, \ldots, n\}) \setminus \{\emptyset\}.$$

We call a *system of arrangements* a finite set of doubly indexed patterns as above together with the mapping ω. We say that a partial mapping $\Pi : \mathbf{Synt} \longrightarrow \wp(\Sigma^*)$ is a *solution of the system of arrangements* iff there are n partial mappings $\Pi_1, \ldots, \Pi_n : \mathbf{Synt} \longrightarrow \wp(\Sigma^*)$ such that, for all $1 \leq j \leq n$:

(i) $\Pi = \Pi_{\omega(0,0)}$;

(ii) $\Pi_j(a^\bullet) = \{a\}$ for all $a \in \Sigma_+$;

(iii) for all $S \in \mathrm{dom}(\Pi_j)$, the set of patterns $\Gamma_{1,j}, \ldots, \Gamma_{m_j,j}$ is applicable to S and:

$$\Pi_j(S) = \Pi_{\omega(1,j)}(S_{\Gamma_{1,j}}) + \cdots + \Pi_{\omega(m_j,j)}(S_{\Gamma_{m_j,j}}).$$

Numbers in $\{1, \ldots, n\}$ represent equations. The first condition just says that the calculation begins with the equations invoked by $\omega(0,0)$. The second condition expounds the base cases while the third condition establishes the recurrences. Conditions on applicability permit us to prove inductively the correct working of the system. When $|\omega(i,j)| = 1$ for each i, j the system becomes *deterministic*, and for each syntagma Π yields a unique string.

Notice that if Π is a solution of a system of arrangements with domain $\mathrm{dom}(\Pi)$ we can obtain another solution Π' by picking any subdomain $\mathrm{dom}(\Pi') \subseteq \mathrm{dom}(\Pi)$, however we can take always the largest solution:

Lemma 7.4. *Given a system of arrangements, there is a unique solution which is maximal over the domain.*

Proof. Let Π and Π' be two solutions of a system of arrangements. We notate $D = \mathrm{dom}(\Pi)$ and $D' = \mathrm{dom}(\Pi')$ (might be empty) and we prove that $\Pi(S) = \Pi(S')$ for any $S \in D \cap D'$. Let $\Pi_1, \ldots, \Pi_n$ be the mappings of the solution Π with domains $D_1, \ldots, D_n$ and $\Pi'_1, \ldots, \Pi'_n$ be the mappings of the solution Π' with domains $D'_1, \ldots, D'_n$. We proceed by induction on the size s of S. By the point (ii) in the definition, if S is atomic or null the statement is trivial, $\Pi_j(S) = \Pi'_j(S)$, for each $j = 1, \ldots, n$ and then $\Pi(S) = \Pi'(S)$. We suppose that the statement is true for syntagmata with size $< s$. Let $S \in D_j \cap D'_j$ be a syntagma with $|S| = k$. Then we can suppose that for every i, j, we have $S_{\Gamma_{i,j}} \in \bigcup_{k \in \omega(i,j)} D_k \cap D'_k$, otherwise one of the mappings Π_j or Π'_j is not defined for S, and then Π or Π' is not defined, which is a contradiction. Then, since the involved patterns are applicable (point (iii) in the definition), $|S_{\Gamma_{i,j}}| < |S|$ and we can use the hypothesis of induction:

$$\begin{aligned}\Pi'_j(S) &= \Pi'_{\omega(1,j)}(S_{\Gamma_{1,j}}) + \cdots + \Pi'_{\omega(m_j,j)}(S_{\Gamma_{m_j,j}}) \\ &= \Pi_{\omega(1,j)}(S_{\Gamma_{1,j}}) + \cdots + \Pi_{\omega(m_j,j)}(S_{\Gamma_{m_j,j}}) = \Pi_j(S).\end{aligned}$$

Since this works for any j, $\Pi'(S) = \Pi'_{\omega(0,0)}(S) = \Pi_{\omega(0,0)}(S) = \Pi(S)$. Now it is easy to see that there is a unique maximal solution. Since given two solutions Π and Π' they coincide in the domain $D \cap D'$, we can construct a new solution say Π'' with domain $D \cup D'$, defined as $\Pi''(S) = \Pi(S)$ if $S \in D$ and $\Pi''(S) = \Pi'(S)$ if $S \in D'$. Thus to find the maximal solution we only have to join all the possible solutions. □

In the sequel "the solution of a system" will stands for "the maximal solution". When there is no possible confusion, we write Π to refer to the solution or the system in itself.

Definition 7.5 (Algebraic Dependency Grammar)**.** Let Π be a system of arrangements as in the above definition, and let W be a manifold with effectively bounded ellipticity such that $W \subseteq \mathrm{dom}(\Pi)$. We define an *algebraic dependency grammar* as a pair: $\mathscr{G} = (W, \Pi)$, and its associated language is:

$$\mathscr{L}(\mathscr{G}) = \bigcup_{S \in W} \Pi(S).$$

We say that a language L is an *algebraic dependency language* iff there exists an algebraic dependency grammar $\mathscr{G}$ such that $L = \mathscr{L}(\mathscr{G})$.

We ensure that the definition is sound:

Proposition 7.6. *Let $\mathscr{G} = (W, \Pi)$ be an algebraic dependency grammar.*

(i) For each $S \in W$, $\Pi(S)$ is a finite set, and can be effectively computed.

(ii) For each $x \in \Pi(S)$, $S \in W$ and there is an individual linearization $\ell : S \longrightarrow x$.

Proof. (i) By induction on the size of S. When the syntagma is atomic or null, then trivially we can compute $\Pi(a^\bullet) = a$. We suppose that the statement is true for any syntagma with size $< s$ and let S be a syntagma with $|S| = s$. We prove that every mapping Π_j can be calculated in a finite number of steps. First notice that given a syntagma and a pattern we can effectively calculate its induced subsyntagma. As in the above proof, $|S_{\Gamma_{i,j}}| < |S| = s$, and then according to the recurrence:

$$\Pi_j(S) = \Pi_{\omega(1,j)}(S_{\Gamma_{1,j}}) + \cdots + \Pi_{\omega(m_j,j)}(S_{\Gamma_{m_j,j}})$$

Π_j can be calculated in a finite number of steps, since every $\Pi_{\omega(i,j)}(S_{\Gamma_{i,j}})$ can be calculated in a finite number of steps. So Π is calculated in a finite number of steps and in addition $\Pi(S)$ is a finite set.

(ii) We notice that since since $S_{\Gamma_{1,j}}, \ldots, S_{\Gamma_{m_j,j}}$ is an arrangement of S, every equation is well balanced in the following sense. Let us write $|S|_a = |S^{-1}(a)|$. $S_{\Gamma_{1,j}}, \ldots, S_{\Gamma_{m_j,j}}$

must form a partition of S, whereby $\mathrm{Spt}(S) = \bigcup_{i=1}^{m_j} \mathrm{Spt}(S\Gamma_{i,j})$. Since the sets are pairwise disjoints, for every $a \in \Sigma$ we have:

$$|S|_a = \sum_{i=1}^{m_j} |S\Gamma_{i,j}|_a = \sum_{i=1}^{m_j} |S_{\Gamma_{i,j}}|_a.$$

Then the recurrences satisfies that:

$$|\Pi_j(S)|_a = |\Pi_{\omega(1,j)}(S_{\Gamma_{1,j}})|_a + \cdots + |\Pi_{\omega(m_j,j)}(S_{\Gamma_{m_j,j}})|_a,$$

that is, equations do not add and do not remove the whole number of each letter. In consequence, if we take an $x \in \Pi(S)$, the letters of x and the letters of S are the same. So there is a bijection $\ell : \mathrm{Spt}(S) \longrightarrow \mathrm{Spt}(x)$ which is an individual linearization. □

Remark 7.7. Linearizations so defined work "continuously": when we consider a subsyntagma $S_{\Gamma_{i,j}}$ of S, then $\Pi_{\omega(i,j)}(S_{\Gamma_{i,j}})$ becomes a substring (or an interval) of $\Pi_j(S)$. What can be discontinuous are the patterns $\Gamma_{i,j}$. This will be the way in which we obtain discontinuous examples in natural languages.

7.3 Some Comments on the Definition

Notation

We adopt the following conventions on vocabulary and notation. Given a system of arrangements Π, for each recurrence equation:

$$\Pi_j(S) = \Pi_{\omega(1,j)}(S_{\Gamma_{1,j}}) + \cdots + \Pi_{\omega(m_j,j)}(S_{\Gamma_{m_j,j}}),$$

we take the following simpler expression which we call an *arrangement*:

$$A_j = \Gamma_{1,j} + \cdots + \Gamma_{m_j,j}.$$

We identify recurrence equations with arrangements (hence the term "system of arrangements"). We say that the pattern $\Gamma_{i,j}$ *invokes* the arrangement (equation) k iff $k \in \omega(i,j)$. The function ω can be represented graphically as in Fig. 7.3(a) and (b) where each arrow represents the patterns invoked by ω. For practical issues we need a notational system to explicate ω. Given an arrangement A_j we will write under each induced subsyntagma the value of ω:

$$\begin{cases} A_1 = \underset{\omega(1,1)}{\Gamma_{1,1}} + \cdots + \underset{\omega(m_1,1)}{\Gamma_{m_1,1}}, \\ \quad\vdots \\ A_n = \underset{\omega(1,n)}{\Gamma_{1,n}} + \cdots + \underset{\omega(m_n,n)}{\Gamma_{m_n,n}}. \end{cases}$$

So a system of arrangements Π can be described as a finite set of arrangements with the mapping ω, i.e. $(\{A_1, \ldots, A_m\}, \omega)$. When we need to separate the solution from the system in itself we denote the solution as $\Pi_{\mathcal{A}}$, where $\mathcal{A} = (\{A_1, \ldots, A_m\}, \omega)$.

On Computational Implementation

In a computational implementation a system of arrangements can be treated as a rewriting process in the free monoid **Synt***. We start with a syntagma $S \in W$ and we choose an arrangement

$$A_k = \Gamma_{1,k} + \cdots + \Gamma_{m_k,k},$$

such that $k \in \omega(0, 0)$. Then we remove S and we write the string of syntagmata

$$S_{\Gamma_{1,k}} + \cdots + S_{\Gamma_{m_k,k}}.$$

We notate this string $A_k(S)$ and we call it *the application of the arrangement A_k to the syntagma S*. In the second step we pick some syntagma $S_{\Gamma_{i,j}}$ from this string and we choose some other arrangement $A_{k'}$ with $k' \in \omega(i, j)$. We remove $S_{\Gamma_{i,j}}$ and put in its place the string of syntagmata $A_{k'}(S_{\Gamma_{i,j}})$. We repeat the last step as many times as is necessary to obtain a string of atomic syntagmata.

We call this process a *derivation*. Thus $\Pi(S)$ is the set of all the possible derivations of S. Notice that the function ω is indicating which arrangements can be invoked for a syntagma depending just on the result of the last arrangement applied to this. So the system only remembers the last step and forgets the previous steps. I.e. there are no “transderivational” constraints. In addition this procedure allows us to know whether given a syntagma S, $S \in^? \mathrm{dom}(\Pi_{\mathcal{A}})$, where $\Pi_{\mathcal{A}}$ is the solution of the system $\mathcal{A}$. If for all the derivations the induced subsyntagmata exist and patterns are applicable (which can be decided in linear time in the size of S), then $S \in^? \mathrm{dom}(\Pi_{\mathcal{A}})$, otherwise the computation stops and $S \notin \mathrm{dom}(\Pi_{\mathcal{A}})$.

Notice that when in a derivation appears a null or an atomic syntagma it persists during all the computation until the end. If we apply an arrangement to it we will obtain the same null or atomic syntagma. The possible arrangements invoked by a constant patterns, which always induce atomic syntagmata, are innocuous. For this reason, for constant patterns we do not specify ω.

Another special case is when the set of arrangements only contains one arrangement $\{A_1\}$. For these linearizations ω can only take the value $\omega(i, j) = \{1\}$. So we will specify nothing.

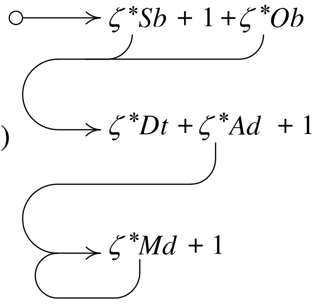
Some theorems are proved more easily with the original definition, while others in this last fashion. In order to show examples we use this rewriting style. Every time that we apply an arrangement we write $\rightsquigarrow$ and we notate at the right margin which arrangement has been applied. To avoid writing subsyntagmata of subsyntagmata, for example $(S_{\zeta^* Sb})_{\zeta^* Dt}$, we only write the last pattern in the computation, $\zeta^* Dt$.

Some Examples of Linearizations

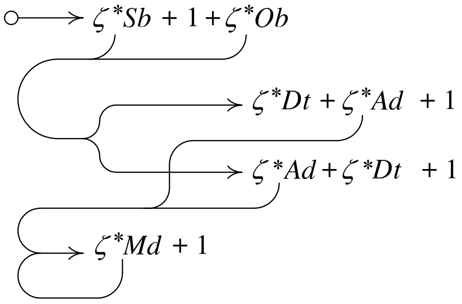
Example 20. An example of notation. We see two examples of systems of arrangements. First in the notation commented above and second in diagrammatic form. The system (I) only offers one possible order, while the system (II) accepts the adjective in *prenominal* or *postnominal* position.

$$\text{(I)} \begin{cases} A_1 = & \underset{2}{\zeta^*}Sb + 1 + \underset{2}{\zeta^*}Ob; \\ A_2 = & \zeta^*Dt + \underset{3}{\zeta^*}Ad + 1; \\ A_3 = & \underset{3}{\zeta^*}Md + 1. \end{cases} \qquad \text{(II)} \begin{cases} A_1 = & \underset{2,3}{\zeta^*}Sb + 1 + \underset{2,3}{\zeta^*}Ob; \\ A_2 = & \zeta^*Dt + \underset{4}{\zeta^*}Ad + 1; \\ A_3 = & \zeta^*Dt + 1 + \underset{4}{\zeta^*}Ad; \\ A_4 = & \underset{4}{\zeta^*}Md + 1. \end{cases}$$

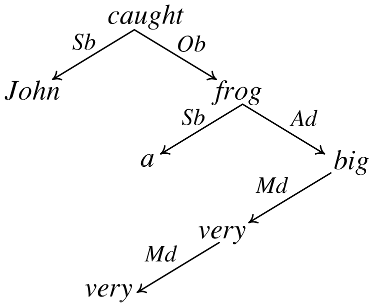
(I)



(II)



Example 21. A simple linguistic example. Consider the following grammar which captures simple sentences in English. We take the manifold W_{Eng} from Example 9. We consider the system of arrangements (I) from the previous Example 20. This defines a linearization Π_{Eng}. Joining both we obtain the grammar $(W_{\text{Eng}}, \Pi_{\text{Eng}})$ the language of which is a little fragment of English concerning simple sentences. For instance, the manifold W_{Eng} is able to license the syntagma:



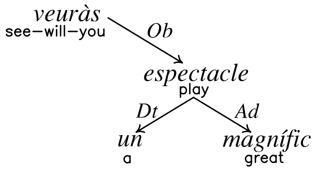
We have the following derivation:

$$
\begin{aligned}
S &\rightsquigarrow \zeta^{*}Sb + \text{caught} + \zeta^{*}Ob && (A_1)\\
&\rightsquigarrow (\zeta^{*}Dt + \zeta^{*}Ad + \text{John}) + \text{caught} + \zeta^{*}Ob && (A_2)\\
&\rightsquigarrow (0 + 0 + \text{John}) + \text{caught} + \zeta^{*}Ob && (A_3)\\
&\rightsquigarrow \text{John} + \text{caught} + (\zeta^{*}Dt + \zeta^{*}Ad + \text{frog}) && (A_2)\\
&\rightsquigarrow \text{John} + \text{caught} + \text{a} + (\zeta^{*}Md + \text{big}) + \text{frog} && (A_3)\\
&\rightsquigarrow \text{John} + \text{caught} + \text{a} + (\zeta^{*}Md + \text{very}) + \text{big} + \text{frog} && (A_3)\\
&\rightsquigarrow \text{John} + \text{caught} + \text{a} + (\zeta^{*}Md + \text{very}) + \text{very} + \text{big} + \text{frog} && (A_3)\\
&\rightsquigarrow \text{John} + \text{caught} + \text{a} + 0 + \text{very} + \text{very} + \text{big} + \text{frog}\\
&\rightsquigarrow \text{John} + \text{caught} + \text{a} + \text{very} + \text{very} + \text{big} + \text{frog.}
\end{aligned}
$$

We are going to see several examples in the following sections whose peculiarities help us to understand the nature of such grammars. For a lot of examples on formal languages we need just one arrangement. However natural languages with a great freeness of order need a large number of arrangements.

A linearization is in general not *deterministic* or *convergent* since linearizations can derive different word-orders from the same syntagma. We recall the examples in §7.1.2 about parentheticals or shifting adjectives in Romance languages.

Example 22. An example of non-convergence . As we said, systems of arrangements permit computations in parallel, and different word orders from a unique syntactic structure. We start with a unique syntagma:



We want to achieve the two sentences in Catalan:

(23) Veuràs un magnífic espectacle. / Veuràs un espectacle magnífic.
see-will-you a great play / see-will-you a play great
'You are going to see a great play'.

We take the system of arrangements (II) from the previous Example 20. Let us see the two possible derivations:

$$\begin{aligned}
S &\rightsquigarrow \zeta^* Sb + \text{veuràs} + \zeta^* Ob && (A_1)\\
&\rightsquigarrow (0) + \text{veuràs} + \zeta^* Ob && (A_2) \text{ or } (A_3)\\
&\rightsquigarrow \text{veuràs} + (\zeta^* Dt + \zeta^* Ad + \text{espectacle}) && (A_2)\\
&\rightsquigarrow \text{veuràs} + \text{un} + \text{magnífic} + \text{espectacle} && (A_4)
\end{aligned}$$

$$\begin{aligned}
S &\rightsquigarrow \zeta^* Sb + \text{veuràs} + \zeta^* Ob && (A_1)\\
&\rightsquigarrow (0) + \text{veuràs} + \zeta^* Ob && (A_2) \text{ or } (A_3)\\
&\rightsquigarrow \text{veuràs} + (\zeta^* Dt + \text{espectacle} + \zeta^* Ad) && (A_3)\\
&\rightsquigarrow \text{veuràs} + \text{un} + \text{espectacle} + \text{magnífic} && (A_4)
\end{aligned}$$

7.4 Decidability of Algebraic Dependency Grammars

The *universal recognition problem* is the problem of deciding for an algebraic dependency grammar $\mathscr{G} = (W, \Pi)$ and string $x \in \Sigma^*$ whether $x \in^? \mathscr{L}(\mathscr{G})$. In order to tackle this we can decompose the problem into two subproblems:

- First we parse the word x, which consists in finding all the syntagmata S such that $x \in \Pi(S)$. I.e. finding the set $\Pi^{-1}(\{x\})$.
- Second we check if there is some such syntagma such that $S \in W$.

Thus, if there is some $S \in \Pi^{-1}(\{x\})$ such that $S \in W$ then $x \in \mathscr{L}(\mathscr{G})$, otherwise $x \notin \mathscr{L}(\mathscr{G})$. So we have two subproblems: *parsing* and $S \in^? W$. We do not deal with the parsing problem, but we show that Π^{-1} is computable in finite time.

Theorem 7.8. *Let $\mathscr{G} = (W, \Pi)$ be an algebraic dependency grammar. If W is decidable, the universal problem $x \in^? \mathscr{L}(\mathscr{G})$ is decidable.*

Proof. We start from the equivalence $x \in \mathscr{L}(\mathscr{G}) \iff \Pi^{-1}(\{x\}) \cap W \neq \emptyset$. So first we calculate the set $\Pi^{-1}(\{x\})$ and then we verify whether there is a $S \in \Pi^{-1}(\{x\})$ such that $S \in W$. Recall that $\mathrm{Env}(S) = \mathrm{Spt}(S) \sqcup \mathrm{Ell}(S)$, by Lemma 2.8, and therefore $|\mathrm{Env}(S)| = |S| + |\mathrm{Ell}(S)|$. If $S \in \Pi^{-1}(x)$ then there is an individual linearization $\ell : S \longrightarrow x$ which means that $|S| = |x|$. So: $S \in \Pi^{-1}(\{x\}) \implies |\mathrm{Env}(S)| = |x| + |\mathrm{Ell}(S)|$. Since by definition the ellipticity is effectively bounded, $|\mathrm{Env}(S)| = |x| + |\mathrm{Ell}(S)| \leq |x| + \varepsilon(|S|) = |x| + \varepsilon(|x|)$. Then:

$$S \in \Pi^{-1}(\{x\}) \implies |\mathrm{Env}(S)| \leq |x| + \varepsilon(|x|).$$

If we notate the set $A_x = \{S \in \textbf{Synt} \mid$ there is a linearization $\ell : S \longrightarrow x$ such that $|\text{Env}(S)| \leq |x| + \varepsilon(|x|)\}$, this last statement can be paraphrased as $\Pi^{-1}(\{x\}) \subseteq A_x$. It is easy to check that A_x is a finite set, the cardinality of which is equivalent to the number of trees with at most $|x| + \varepsilon(|x|)$ vertices.

Now we prove that $\Pi^{-1}(\{x\})$ can be computed. Since the ellipticity is bounded by a computable function ε, we can construct the set A_x and for each $S \in A_x$ we consider the problem $S \in^? \Pi^{-1}(\{x\})$. In order to solve this we construct the set $\Pi(S)$, which can be built in finite time (we only have to apply the system of arrangements on S and write all the possible linearizations) and we saw before that $S(x)$ is finite (Proposition 7.6). If there is some $y \in \Pi(S)$ such that $y = x$ then $S \in \Pi^{-1}(x)$, otherwise $S \notin \Pi^{-1}(x)$.

Now we just have to check whether there is some $S \in \Pi^{-1}(\{x\})$ such that $S \in W$ which can be made in finite time because $\Pi^{-1}(\{x\})$ is finite and W decidable. □

7.5 The Bi-Hierarchies

A remarkable fact is that we are using the same building blocks for manifolds and linearizations, namely patterns. This opens a corridor between both modules of our grammars and suggests the next definition. Recall that we defined $\textbf{Man}(X)$ as the set of manifolds that have type X. Now we can do the same for linearizations. In this way we can classify a language according to a pair of types. Recall that these descriptions are given by a classificatory monoid. So depending on the classificatory monoid chosen we obtain different hierarchies of classes of languages, which we name bi-hierarchies.

Definition 7.9 (Bi-hierarchies)**.** Consider a fixed classificatory monoid $\{\mathbf{T}_1, \ldots, \mathbf{T}_n\}^*$ the elements of which we called types.

(i) (reminder) We say that a manifold *has type X* iff all the patterns of the simple manifolds involved in the definition of the manifold have type X; we define the set of *manifolds of type X* as:

$$\textbf{Man}(X) = \{W \subseteq \textbf{Synt} \mid W \text{ has type } X\}.$$

(ii) We say that an arrangement *has type Y* iff all the patterns involved in the arrangement have type Y. We say that a system of arrangements Π *has type Y* iff all the arrangements involved in the definition have type Y. We define the set of *linearizations of type Y* as:

$$\textbf{Lin}(Y) = \{\Pi : \textbf{Synt} \longrightarrow \wp(\Sigma^*) \mid \Pi \text{ has type } Y\}.$$

(iii) We define the class of *languages of type X over type Y* as:

$$X/_Y = \{\mathscr{L}(\mathscr{G}) \mid \mathscr{G} = (W, \Pi), W \in \textbf{Man}(X), \Pi \in \textbf{Lin}(Y)\}.$$

We call a *bi-hierarchy over* $\{\mathbf{T}_1, \ldots, \mathbf{T}_n\}^*$ the set of these classes of languages X/Y with types in the classificatory monoid and we notate it as $\mathcal{BH}(\mathbf{T}_1, \ldots, \mathbf{T_n})$.

Example 23. Some examples of bi-hierarchies are *the general bi-hierarchy* $\mathcal{BH}(\mathbf{k}, \mathbb{M})$, the *homogeneous bi-hierarchy* $\mathcal{BH}(\mathbf{k}, \mathbb{H})$, and the *pivoting bi-hierarchy*, $\mathcal{BH}(\mathbf{k}, \mathbb{P}, \mathbb{H})$. Let us list the first combinations in $\mathcal{BH}(\mathbf{k}, \mathbb{M})$ (we enumerate them by the sum of lengths $|X| + |Y|$):

$\lvert X\rvert + \lvert Y\rvert = 0$	$\mathbf{1}/\mathbf{1}$;
$\lvert X\rvert + \lvert Y\rvert = 1$	$\mathbf{1}/\mathbb{M}$, $\mathbf{1}/\mathbf{k}$, $\mathbb{M}/\mathbf{1}$, $\mathbf{k}/\mathbf{1}$;
$\lvert X\rvert + \lvert Y\rvert = 2$	$\mathbf{1}/\mathbb{M}^2$, $\mathbf{1}/\mathbb{M}\mathbf{k}$, $\mathbf{1}/\mathbf{k}\mathbb{M}$, $\mathbf{1}/\mathbf{k}^2$, $\mathbb{M}/\mathbb{M}$, $\mathbb{M}/\mathbf{k}$, $\mathbf{k}/\mathbb{M}$, $\mathbf{k}/\mathbf{k}$, $\mathbb{M}^2/\mathbf{1}$, $\mathbb{M}\mathbf{k}/\mathbf{1}$, $\mathbf{k}\mathbb{M}/\mathbf{1}$; $\mathbf{k}^2/\mathbf{1}$;
$\lvert X\rvert + \lvert Y\rvert = 3$	$\mathbf{1}/\mathbb{M}^3$, $\mathbf{1}/\mathbb{M}^2\mathbf{k}$, $\mathbf{1}/\mathbb{M}\mathbf{k}\mathbb{M}$, $\mathbf{1}/\mathbf{k}\mathbb{M}^2$, $\mathbf{1}/\mathbf{k}^3$, $\mathbf{1}/\mathbf{k}^2\mathbb{M}$, $\mathbf{1}/\mathbf{k}\mathbb{M}\mathbf{k}$, $\mathbf{1}/\mathbb{M}\mathbf{k}^2$, $\mathbb{M}/\mathbb{M}^2$, $\mathbb{M}/\mathbb{M}\mathbf{k}$, $\mathbb{M}/\mathbf{k}\mathbb{M}$, $\mathbb{M}/\mathbf{k}^2$, $\mathbf{k}/\mathbb{M}^2$, $\mathbf{k}/\mathbb{M}\mathbf{k}$, $\mathbf{k}/\mathbf{k}\mathbb{M}$, $\mathbf{k}/\mathbf{k}^2$, $\mathbb{M}^2/\mathbb{M}$, $\mathbb{M}^2/\mathbf{k}$, $\mathbf{k}^2/\mathbb{M}$, $\mathbf{k}^2/\mathbf{k}$, $\mathbf{k}\mathbb{M}/\mathbb{M}$, $\mathbf{k}\mathbb{M}/\mathbf{k}$, $\mathbb{M}\mathbf{k}/\mathbb{M}$, $\mathbb{M}\mathbf{k}/\mathbf{k}$, $\mathbb{M}^3/\mathbf{1}$, $\mathbb{M}^2\mathbf{k}/\mathbf{1}$, $\mathbb{M}\mathbf{k}\mathbb{M}/\mathbf{1}$, $\mathbf{k}\mathbb{M}^2/\mathbf{1}$, $\mathbf{k}^3/\mathbf{1}$, $\mathbf{k}^2\mathbb{M}/\mathbf{1}$, $\mathbf{k}\mathbb{M}\mathbf{k}/\mathbf{1}$, $\mathbb{M}\mathbf{k}^2/\mathbf{1}$;
$\vdots$	$\vdots$
$\lvert X\rvert + \lvert Y\rvert = n$	$(n+1)\cdot 2^n$ possible classes.

As we will prove, some classes collapse in one class, for example, the class of languages $X/\mathbf{1}$ is the same regardless of the type X, Corollary 8.11.

The bi-hierarchies are monotone in the following sense:

Lemma 7.10. *For any types X, Y the following holds:*

(i) $X \sqsubseteq X' \implies \mathbf{Man}(X) \subseteq \mathbf{Man}(X')$*;*

(ii) $Y \sqsubseteq Y' \implies \mathbf{Lin}(Y) \subseteq \mathbf{Lin}(Y')$*;*

(iii) $X \sqsubseteq X', Y \sqsubseteq Y' \implies {}^{X}/_{Y} \subseteq {}^{X'}/_{Y'}$.

Proof. (i) and (ii) are trivial. (iii) If $L \in X/Y$ then $L = \mathscr{L}(W, \Pi)$ for some $W \in \mathbf{Man}(X), \Pi \in \mathbf{Lin}(Y)$. Since $X \sqsubseteq X', Y \sqsubseteq Y'$ then $\mathbf{Man}(X) \subseteq \mathbf{Man}(X'), \mathbf{Lin}(Y) \subseteq \mathbf{Lin}(Y')$. So $L = \mathscr{L}(W, \Pi) \in X'/Y'$. □

Thus we could expect that the simpler the patterns, the simpler are the languages, as we will exemplify in subsequent chapters.

There is a simple relation between a classificatory monoid and its bi-hierarchy. Consider the direct product $\{\mathbf{T}_1, \ldots, \mathbf{T}_n\}^* \times \{\mathbf{T}_1, \ldots, \mathbf{T}_n\}^*$ with the component-wise order: $(X,Y) \sqsubseteq (X',Y') \iff X \sqsubseteq X',\ Y \sqsubseteq Y'$. Now consider $\mathcal{BH}(\mathbf{k}, \mathbb{M})$ with the subset order $\subseteq$. Then the mapping:

$$\begin{aligned} \{\mathbf{T}_1, \ldots, \mathbf{T}_n\}^* \times \{\mathbf{T}_1, \ldots, \mathbf{T}_n\}^* &\longrightarrow \mathcal{BH}(\mathbf{T}_1, \ldots, \mathbf{T}_n) \\ (X,Y) &\longmapsto {}^{X}\!/_{Y} \end{aligned}$$

is an epimorphism of orders; this is just another way of stating Lemma 7.10.

Remark 7.11. The previous epimorphism admits a linguistical interpretation. The pair $(X,Y) \in \{\mathbf{T}_1, \ldots, \mathbf{T}_n\}^* \times \{\mathbf{T}_1, \ldots, \mathbf{T}_n\}^*$ represents the description of the structural devices of the class of grammars with type X for the manifold and the type Y for the linearization, while the set X/Y represents the weak capacity. The epimorphism of orders, then, establishes the relation between both. So the pair (X,Y) is a new interpretation of "strong capacity" for classes of grammars.

So when we say that a certain construction of natural language has a type X for manifold and a type Y for linearization we are saying that there is a grammar generating the construction with structural devices (or a strong capacity) given by the pair of types (X,Y), and a weak capacity given by the class X/Y, for which it might be possible to find a weakly equivalent grammar with a simpler strong capacity: since the above morphism of orders is surjective but not injective, grammars with different strong capacity can have the same weak capacity.

7.6 First Examples

Consider the classical formal languages named *squares language*, *copy language*, *multiple abc language*, *respectively language* and *mirror language* defined by:

- $L_{\text{squa}} = \{a_1^2 \cdots a_n^2 \mid a_1, \ldots, a_n \in \Sigma, n \in \mathbb{N}_+\}$;
- $L_{\text{copy}} = \{x^2 \mid x \in \Sigma^*\}$;
- $L_{\text{mult}} = \{(abc)^n \mid n \in \mathbb{N}_+\}$;
- $L_{\text{resp}} = \{a^n b^n c^n \mid n \in \mathbb{N}_+\}$;
- $L_{\text{mirr}} = \{xx^R \mid x \in \Sigma^*\}$.

In §4.3.1 we highlighted the importance of these languages as idealizations of certain linguistic phenomena of natural language. We also defined several manifolds which captured their syntactic structure. In order to design a global linearization we can begin to imagine how to draw an individual linearization for some syntagmata of a given

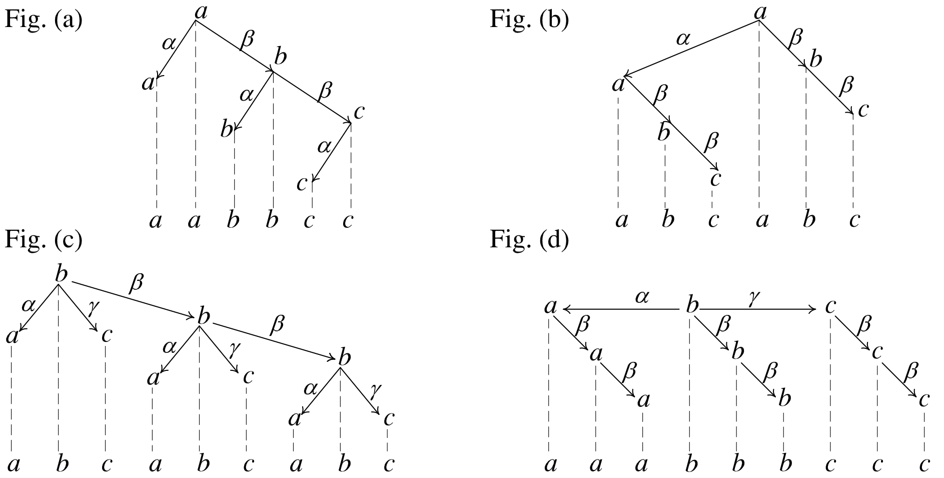


Figure 7.4: individual linearizations for (a) the squares language, (b) the copy language, (c) the multiple *abc* language, and (d) the respectively language.

manifold. Fig. 7.4 does this. The following examples conclude with a grammar for each language. In particular, on the one hand we have that:

$$L_{\text{squa}} \in {}^{\mathbf{k}^2\mathbb{H}}\!/_{\mathbb{H}\mathbf{k}}, \quad L_{\text{mult}} \in {}^{\mathbf{k}\mathbb{H}}\!/_{\mathbb{H}\mathbf{k}}, \quad L_{\text{mirr}} \in {}^{\mathbf{k}^2\mathbb{H}}\!/_{\mathbb{H}\mathbf{k}};$$

and on the other hand we have that:

$$L_{\text{copy}} \in {}^{\mathbb{H}\mathbf{k}^2}\!/_{\mathbb{H}\mathbf{k}}, \quad L_{\text{resp}} \in {}^{\mathbb{H}\mathbf{k}}\!/_{\mathbb{H}\mathbf{k}}, \quad L_{\text{mirr}} \in {}^{\mathbb{H}\mathbf{k}^2}\!/_{\mathbb{H}\mathbf{k}}.$$

Example 24. Squares language and copy language. We begin with the squares language. Consider the manifold $W_{\text{squa}} = \{Q_0, Q_a, Q_b, Q_{a,a}, Q_{a,b}, \ldots\}$ together with one arrangement:

$$A = \zeta^*\alpha + 1 + \zeta^*\beta.$$

We apply this arrangement several times, for example to $Q_{a,b,c}$ (the process is shown in Fig. (a)):

$$\begin{aligned}
&Q_{a,b,c} && \\
&\rightsquigarrow a + a + \zeta^*\beta && (A)\\
&\rightsquigarrow a + a + (b + b + \zeta^*\beta) && (A)\\
&\rightsquigarrow a + a + (b + b + (c + c)) && (A)\\
&\rightsquigarrow a + a + b + b + c + c. &&
\end{aligned}$$

Thus we obtain the language: $L_{\text{squa}} = \{a_1^2 \cdots a_n^2 \,|\, a_1, \dots, a_n \in \Sigma, n \in \mathbb{N}_+\}$. In other words, where Π_{squa} denotes the global linearization established by the above arrangement, we have $\mathscr{L}(W_{\text{squa}}, \Pi_{\text{squa}}) = L_{\text{squa}}$. All the patterns involved in the arrangement $\zeta^*\alpha, \zeta^*\beta$ are in $\mathbb{H}\mathbf{k}$ and $1 \in \mathbf{1} \subseteq \mathbb{H}\mathbf{k}$, so $\Pi_{\text{squa}} \in \mathbf{Lin}(\mathbb{H}\mathbf{k})$. Recall from the Example 7 that $W_{\text{squa}} \in \mathbf{Man}(\mathbf{k}^2\mathbb{H})$ and that it has effectively bounded ellipticity. Thus $L_{\text{squa}} \in \mathbf{k}^2\mathbb{H}/\mathbb{H}\mathbf{k}$.

Fig. (a)

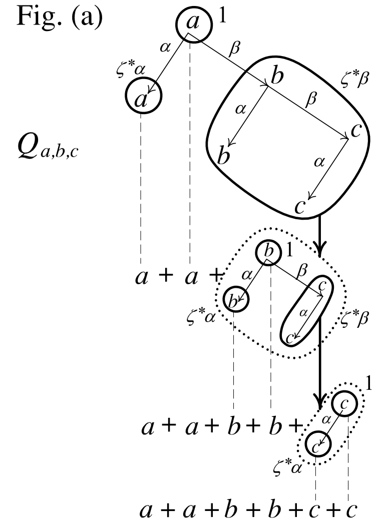
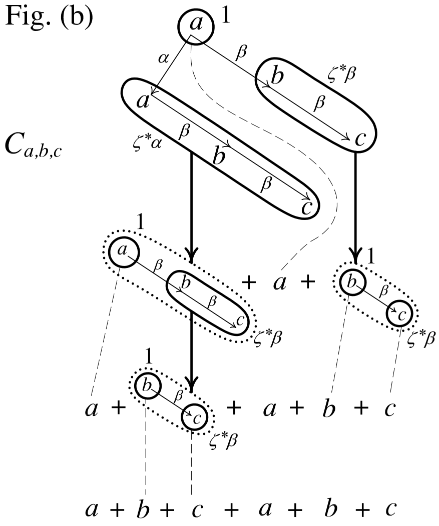


Fig. (b)



Now we are going to linearize the copy manifold by using the same arrangement as in the squares language, $A = \zeta^*\alpha + 1 + \zeta^*\beta$. Consider a syntagma $C_{a,b,c} \in W_{\text{copy}}$ (see Example 8). If we apply the arrangement several times, we have the derivation (see Fig. (b)):

$$\begin{aligned}
&C_{a,b,c} && \\
&\rightsquigarrow (\zeta^*\alpha + a + \zeta^*\beta) && (A)\\
&\rightsquigarrow (0 + a + \zeta^*\beta) + a + \zeta^*\beta && (A)\\
&\rightsquigarrow a + \zeta^*\beta + a + (0 + b + c) && (A)\\
&\rightsquigarrow a + (0 + b + c) + a + b + c && (A)\\
&\rightsquigarrow a + b + c + a + b + c. &&
\end{aligned}$$

The arrangement has type $\mathbb{H}\mathbf{k}$, and W_{copy} has type $\mathbb{H}\mathbf{k}^2$, so $L_{\text{copy}} \in \mathbb{H}\mathbf{k}^2/\mathbb{H}\mathbf{k}$.

Example 25. Multiple *abc* language, respectively language and mirror language. Let us review the classifications of these languages. For the multiple *abc* language we take the manifold W_{mult} which contains syntagmata denoted by $M_0, M_1, M_2 \dots$. We take the arrangement: $A = \zeta^*\alpha + \zeta^*\beta + 1 + \zeta^*\gamma$, and applying successively we will have for M_3:

$$\begin{array}{ll}
M_3 & \\
\rightsquigarrow (a + b + c + \zeta^*\gamma) & (A) \\
\rightsquigarrow a + b + c + (a + b + c + \zeta^*\gamma) & (A) \\
\rightsquigarrow a + b + c + a + b + c + (a + b + c + 0) & (A) \\
\rightsquigarrow a + b + c + a + b + c + a + b + c. &
\end{array}$$

The arrangement contains patterns in $\mathbb{H}\mathbf{k}$, and since W_{mult} uses patterns in $\mathbf{k}\mathbb{H}$ the language is in $\mathbf{k}\mathbb{H}/\mathbb{H}\mathbf{k}$. For the respectively language we take the same above arrangement which operates on the syntagmata $R_0, R_1, R_2, \ldots \in W_{\text{resp}} \in \mathbf{Man}(\mathbb{H}\mathbf{k})$. So the language is in $\mathbb{H}\mathbf{k}/\mathbb{H}\mathbf{k}$. An example of linearization follows:

$$\begin{array}{ll}
R_3 & \\
\rightsquigarrow (\zeta^*\alpha + \zeta^*\beta + c + \zeta^*\gamma) & (A) \\
\rightsquigarrow (a + \zeta^*\gamma) + \zeta^*\beta + c + \zeta^*\gamma & (A) \\
\rightsquigarrow a + (a + a) + \zeta^*\beta + c + \zeta^*\gamma & (A) \\
\rightsquigarrow a + a + a + (b + \zeta^*\gamma) + c + \zeta^*\gamma & (A) \\
\rightsquigarrow a + a + a + b + (b + b) + c + \zeta^*\gamma & (A) \\
\rightsquigarrow a + a + a + b + b + b + c + (c + c) & (A) \\
\rightsquigarrow a + a + a + b + b + b + c + c + c. &
\end{array}$$

Regarding the mirror language we discover an interesting property: it can be obtained equally either from the squares manifold or the copy manifold. If we take syntagmata from the squares manifold then we can use the arrangement: $A = \zeta^*\alpha + \zeta^*\beta + 1$. For example:

$$\begin{array}{ll}
Q_{a,b,c} & \\
\rightsquigarrow (a + \zeta^*\beta + a) & (A) \\
\rightsquigarrow a + (b + \zeta^*\beta + b) + a & (A) \\
\rightsquigarrow a + b + (c + c) + b + a & (A) \\
\rightsquigarrow a + b + c + c + b + a. &
\end{array}$$

Whereby the mirror language is in $\mathbf{k}^2\mathbb{H}/\mathbb{H}\mathbf{k}$. On the other hand if we take syntagmata from the copy manifold we can use the system of three arrangements:

$$\begin{cases}
A_1 = & \underset{2}{\zeta^*}\alpha + \underset{3}{\zeta^*}\beta + 1; \\
A_2 = & 1 + \underset{2}{\zeta^*}\beta; \\
A_3 = & \underset{3}{\zeta^*}\beta + 1.
\end{cases}$$

See for example the derivation:

$$\begin{aligned}
& C_{a,b,c} & \\
& \rightsquigarrow (\zeta^*\alpha + \zeta^*\beta + a) & (A_1) \\
& \rightsquigarrow (a + \zeta^*\beta) + \zeta^*\beta + a & (A_2) \\
& \rightsquigarrow a + (b + c) + \zeta^*\beta + a & (A_2) \\
& \rightsquigarrow a + b + c + (c + b) + a & (A_3) \\
& \rightsquigarrow a + b + c + c + b + a. &
\end{aligned}$$

This means that the mirror language is in $\mathbb{H}\mathbf{k}^2/\mathbb{H}\mathbf{k}$. As a consequence classes in the bi-hierarchy are not in general disjoint because:

$$L_{\text{mirror}} \in {}^{\mathbf{k}^2\mathbb{H}}\!/_{\mathbb{H}\mathbf{k}} \cap {}^{\mathbb{H}\mathbf{k}^2}\!/_{\mathbb{H}\mathbf{k}}.$$

7.7 Projective Arrangements

Definition 7.12. We say that an arrangement A is *projective* iff it is of the form:

$$A = \zeta^*\lambda_1 + \cdots + \zeta^*\lambda_k + 1 + \zeta^*\lambda_{k+1} + \cdots + \zeta^*\lambda_m,$$

such that the λ_i's are syntactic functions in ζ and $1 \leq k \leq m$. We also accept the case $m = 0$ with $A = 1$ as a projective arrangement.

Notice that projective arrangements use patterns in $\mathbb{H}\mathbf{k}$. Fig. 7.5 shows a partition of a syntagma into patterns and a projective arrangement. All the examples up to here were projective. By contrast we show now a pair of non-projective examples which will be developed in more detail in Chapter 11.

Example 26. A NON-PROJECTIVE ARRANGEMENT FOR DUTCH. Consider again the non-projective cross-serial Dutch subordinate clause from example 19. In order to linearize the corresponding syntagma (see Fig. below) we use the arrangement:

$$A_1 = Sb{\cdot}Ob_S^* + \zeta^*{\cdot}Ob{\cdot}Ob_S^* + Ob_S^*.$$

This is a non-projective arrangement because $Sb{\cdot}Ob_S^* \in \mathbf{k}\mathbb{M}$.

If we want to complete the linearization we must add more arrangements:

$$A_2 = 1 + Ob_S^*{\cdot}Ob_S, \quad \text{and} \quad A_3 = \zeta^*{\cdot}Dt + 1.$$

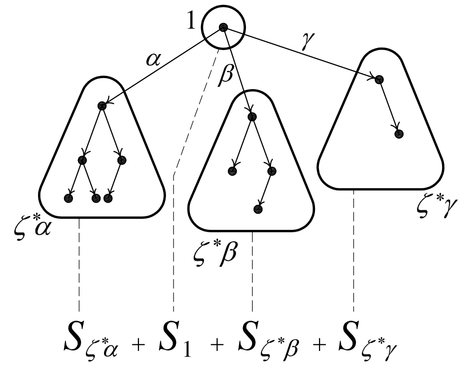
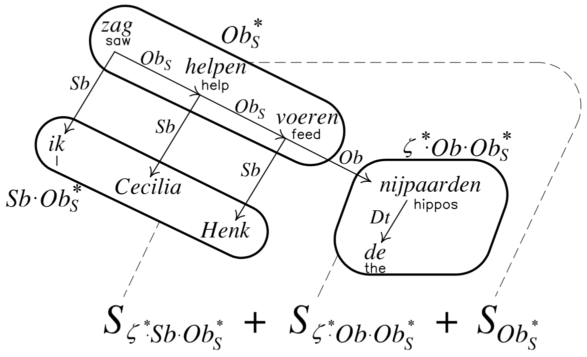


Figure 7.5: partition of a syntagma in patterns and a projective arrangement.

That is, the linearization continues projectively.

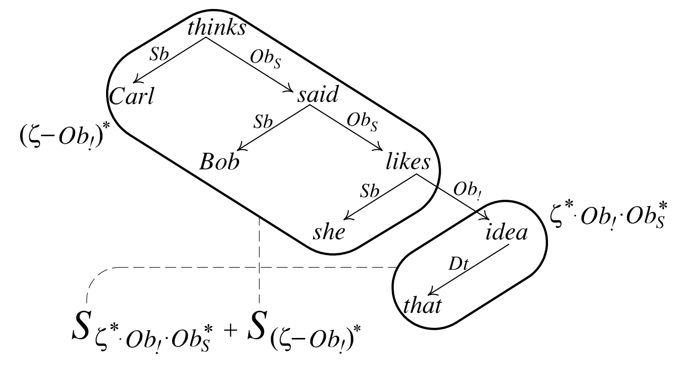


It must be said that in this case the subjects involved are single words. However, since such sentences can exhibit more complex subjects (e.g. with some modifiers) the arrangement must be a bit more general: $A_1 = \zeta^*{\cdot}Sb{\cdot}Ob^*_S + \zeta^*{\cdot}Ob{\cdot}Ob^*_S + Ob^*_S$. Thus $\zeta^*{\cdot}Sb{\cdot}Ob^*_S \in \mathbb{M}\mathbf{k}\mathbb{M}$.

Example 27. A NON-PROJECTIVE ARRANGEMENT FOR TOPICALIZATION IN ENGLISH. Consider now the sentence where the object of a subordinate clause is topicalizated: *That idea, Carl thinks Bob said she likes* (see Fig. below). The ordinary form, *Carl thinks Bob said she likes that idea*, can be obtained by projective arrangements. However the topicalization causes a discontinuity which can be solved by the non-projective arrangement:

$$A_1 = \zeta^* \cdot Ob_! \cdot Ob^*_S + (\zeta - \{Ob_!\})^*,$$

where we have marked the topicalized object with the syntactic function $Ob_!$. This is a non-projective arrangement since $\zeta^* \cdot Ob_! Ob^*_S \in \mathbb{M}\mathbf{k}\mathbb{M}$. Now we can linearize every new syntagma through projective arrangements.


thinks
Sb
Ob_S
Carl
said
$(\zeta - Ob_!)^*$
Sb
Ob_S
Bob
likes
Sb
$Ob_!$
she
idea
$\zeta^* \cdot Ob_! \cdot Ob_S^*$
Dt
that
$S_{\zeta^* \cdot Ob_! \cdot Ob_S^*} + S_{(\zeta - Ob_!)^*}$

Definition 7.13. We say that a system of arrangements is in *projective normal form* iff all the arrangements in it are projective arrangements.

Theorem 7.14. *An individual linearization $\ell : S \longrightarrow x$ is projective iff there is a system of arrangements Π in projective normal form, such that $x \in \Pi(S)$.*

Proof. We sketch the proof. We say that a subtree is *maximal* when it is defined by $\zeta^*\lambda$ with $|\lambda| = 1$. I.e. maximal subtrees are the greatest proper subtrees.

($\Rightarrow$) Let $\ell : S \longrightarrow x$ be a projection. We design a system of arrangements (i.e. a set of arrangements with the mapping ω) which linearizes S into x. We decompose a syntagma S into maximal subtrees and the root. Since ℓ is projective the maximal subtrees and the root becomes intervals. We take as initial arrangement:

$$A_1 = \zeta^*\lambda_1 + \cdots + \zeta^*\lambda_i + 1 + \zeta^*\lambda_{i+1} + \cdots + \zeta^*\lambda_m,$$

where the positions of every $\zeta^*\lambda_i$ correspond to the positions of the intervals of the maximal subtrees in x, and similarly the root. So the mapping ω invokes initially the arrangement A_1. Now we take all maximal subtrees and we deal with them again as full syntagmata with the same linearization ℓ restricted to them. Thus, for each one we design an arrangement $A'_1, \ldots, A'_m$ as before. Then each subtree $\zeta^*\lambda_i$ invokes a new arrangement A'_i. We continue until there are no more subtrees, which yields a tree-shaped diagram of invocations for ω.

($\Leftarrow$) Consider the initial arrangement of a linearization Π in projective normal form:

$$\zeta^*\lambda_1 + \cdots + \zeta^*\lambda_k + 1 + \zeta^*\lambda_{k+1} + \cdots + \zeta^*\lambda_m.$$

This takes the maximal subtrees $\zeta^*\lambda_i$ and puts them into an interval of the final string. All the material of these subtrees will remain inside this interval during the linearization (recall, Remark 7.7, that linearizations work "continuously"). So maximal subtrees are transformed into intervals. The second step of the computation of Π takes the maximal subtrees of the maximal subtrees and puts them into an interval. So these subtrees are also transformed into intervals. An inductive argument proves that every subtree is a maximal subtree of some subtree during the computation. So all the subtrees becomes intervals. □

7.8 A Normal Form for $\mathbf{Lin}(\mathbb{M}\mathbf{k})$ and $\mathbf{Lin}(\mathbb{H}\mathbf{k})$

A system of projective arrangements yields a lineariation in $\mathbf{Lin}(\mathbb{H}\mathbf{k}) \subseteq \mathbf{Lin}(\mathbb{M}\mathbf{k})$. By Theorem 7.14, the issue of projectivity is closely related to these types of patterns. We are going to prove that any linearization in $\mathbf{Lin}(\mathbb{M}\mathbf{k})$ can be expressed through projective arrangements. In particular we will have that $\mathbf{Lin}(\mathbb{H}\mathbf{k}) = \mathbf{Lin}(\mathbb{M}\mathbf{k})$.

Theorem 7.15. *For every linearization in $\Pi_{\mathcal{A}} \in \mathbf{Lin}(\mathbb{M}\mathbf{k})$ there is a linearization $\Pi_{\mathcal{B}}$ given by a system of arrangements, $\mathcal{B}$, in projective normal form such that $\Pi_{\mathcal{A}} = \Pi_{\mathcal{B}}$.*

Proof. For all the proof we recall lemma 3.2 which said that the prime factors of a 1-norm submonoid are prime factors of ζ. Given a system of arrangements, our goal is substituting some (maybe all) of the arrangements by projective arrangements and showing that the linearization induced is always the same. So we start with a system $\mathcal{A}_1 = (\{A_1, \ldots, A_s\}, \omega)$ of arrangements which use patterns in $\mathbb{M}\mathbf{k}$ and we consider the solution of the system $\Pi_{\mathcal{A}}$. We proceed in four steps. In the first three steps we replace the arrangements but not the mapping ω. In the last step we also replace this mapping.

(1) Every arrangement in any linearization in $\mathbf{Lin}(\mathbb{M}\mathbf{k})$ is necessarily a permutation of an arrangement in the form:

$$A = \sum_{i=1}^{n} \alpha_i + \sum_{j=1}^{m} H_j^* \beta_j + \sum_{k=1}^{p} G_k^*,$$

where $n, m, p \geq 0, n+m+p \geq 1$, and $|\alpha_i| = 1, |\beta_j| = 1$, and the sets $H_j, G_k \subseteq \zeta$ can be empty, for each subscript. Notice that this general form A contains projective arrangements under the form $n = 0$, $p = 1$ and $G_1 = \emptyset$, $(\emptyset^* = \{1\})$:

$$A = \sum_{j=1}^{m} \zeta^* \beta_j + 1,$$

or a permutation of this. First we prove that $p \leq 1$ must always hold, otherwise A is not an arrangement. We suppose for the sake of contradiction that $p > 1$. Then there are two different submonoids G_i^*, G_j^*. However the trivial monoid is always a submonoid of any monoid and then $G_i^* \cap G_j^* = \{1\} \neq \emptyset$, so the patterns are not disjoint and they do not constitute a partition, whereby A is not an arrangement. So $p \leq 1$ and any arrangement in $\mathcal{A}_1$ is of the form:

$$A = \sum_{i=1}^{n} \alpha_i + \sum_{j=1}^{m} H_j^* \beta_j + G^*, \text{ or of the form } A = \sum_{i=1}^{n} \alpha_i + \sum_{j=1}^{m} H_j^* \beta_j,$$

or a permutation of this.

Now we are going to see that we can always suppose that $p = 1$. Consider the arrangement with $p = 0$, $A = \sum_{i=1}^{n} \alpha_i + \sum_{j=1}^{m} H_j^* \beta_j$, and we suppose that it is applied to a syntagma S. Notice that:

$$1 \notin \Big(\bigcup_{i=1}^{n} \{\alpha_i\}\Big) \cup \Big(\bigcup_{j=1}^{m} H_j^* \beta_j\Big).$$

Since the arrangement is a partition of S then $1 \notin \mathrm{Spt}(S)$, i.e. $S(1) = 0$. However since $S(1) = 0$, if we add the trivial pattern in the arrangement it does not affect the result and it is a new arrangement: the patterns form a partition of S, S_1 exists, and $0 = |S_1| < |S|$ (provided that $S \neq 0^\bullet$). So we can suppose that the arrangements in $\mathcal{A}_1$ are always of the form (up to permutations):

$$A = \sum_{i=1}^{n} \alpha_i + \sum_{j=1}^{m} H_j^* \beta_j + G^*,$$

although G^* can be trivial, $1 = \emptyset^*$. In order to gain clarity we omit until the end of the proof the limits on the counters: $A = \sum_i \alpha_i + \sum_j H_j^* \beta_j + G^*$.

(2) Now we rewrite the constants. First we check that if we want to apply an arrangement A in $\mathcal{A}_1$ to a syntagma S, then all the constants α_i are leaves in S. Suppose for the sake of contradiction that α_i is not a leaf, or what is the same, that there is a $\psi \in \zeta^*$, $\psi \neq 1$ such that $S(\psi\alpha_i) \neq 0$. Patterns involved in an arrangement must form a partition of the support of S:

$$\mathrm{Spt}(S) \subseteq \Big(\bigcup_{i} \{\alpha_i\}\Big) \cup \Big(\bigcup_{j} H_j^* \beta_j\Big) \cup G^*.$$

But $\psi\alpha_i$ is not in any of these three sets. Let us check this:

(a) Since $\alpha_i, \psi \neq 1$, we have $\psi\alpha_i \notin \bigcup_i \{\alpha_i\}$.

(b) If $\psi\alpha_i \in H_j^* \beta_j$ for some j then there is an $x \in \zeta^*$ such that $\psi\alpha_i = x\beta_j$, but then $\alpha_i = \beta_j$ (recall again that $\alpha_i \neq 1$). Patterns form a partition and this means that they are disjoint. So $\psi\alpha_i \notin H_j^* \beta_j$.

(c) G is a norm-1 submonoid meaning that all the functions in G have norm at most 1. If we suppose that $\psi\alpha_i \in G^*$ all the prime factors of $\psi\alpha_i$ are also in G^*. In particular $\alpha_i \in G^*$, which is also a contradiction, with the fact that patterns of the arrangement form a partition.

In consequence if an arrangement A in $\mathscr{A}_1$ is applied to a syntagma S, $\psi\alpha_i \notin \mathrm{Spt}(S)$ or, what is the same, $S(\psi\alpha_i) = 0$, for any $\psi \in \zeta^*$, so α_i is a leaf. We can substitute the arrangement A by the arrangement (or an adequate permutation of):

$$A' = \sum_i \zeta^*\alpha_i + \sum_j H_j^*\beta_j + G^*.$$

In other words if A can be applied to S, then A' can also be applied to S and $A(S) = A'(S)$. This entails that A' is always an arrangement for S, that is, it forms a partition of S, which can be easily checked. We notate $\mathscr{A}_2$ the result of making this substitution in all the possible arrangements in $\mathscr{A}_1$.

(3) Now we want to rewrite all the factors $H_j^*\beta_j$. Let A be an arrangement in $\mathscr{A}_2$ and suppose that A is applied to a syntagma S. We pick a locus $\psi\beta_j \in (\zeta^*\backslash H_j^*)\beta_j$. Patterns in A must form a partition of S so:

$$\mathrm{Spt}(S) \subseteq \Big(\bigcup_i \zeta^*\alpha_i\Big) \cup \Big(\bigcup_j H_j^*\beta_j\Big) \cup G^*.$$

But, similarly to (2), $\psi\beta_i$ is not in any of these three sets:

(a) Clearly $\psi\beta_i \notin \bigcup_{i=1}^n \zeta^*\alpha_i$, otherwise $\beta_j = \alpha_i$ for some i and we know that $\beta_j \in H_j^*\beta_j$ and $\alpha_i \in \zeta^*\alpha_i$. So $\beta_j \in H_j^*\beta_j \cap \zeta^*\alpha_i \neq \emptyset$, which is a contradiction.

(b) Supose that $\psi\beta_j \in H_k^*\beta_k$. Then $\beta_k = \beta_j$. Since $\beta_j \in H_j^*\beta_j$ and $\beta_k \in H_k^*\beta_k$ then $H_j^*\beta_j \cap H_k^*\beta_k \neq \emptyset$. Since patterns must form a partition of S, the only possibility is $i = k$. But this is a contradiction because we picked $\psi\beta_j$ such that $\psi\beta_j \in (\zeta^*\backslash H_j^*)\beta_j$.

(c) Suppose, finally, that $\psi\beta_j \in G$. Similarly to the point (c) from the second point (2), since G^* is a 1-norm submonoid, prime factors of $\psi\beta_j$ are in G, so $\beta_j \in G$, which is a contradiction.

So necessarily for any syntagma to which the arrangement is applied we have that if $\psi\beta_j \in (\zeta^*\backslash H_j^*)\beta_j$, $S(\psi\beta_j) = 0$. But then we substitute the pattern $H_j^*\beta_j$ in the arrangement A, by the pattern $\zeta^*\beta_j$. We notate A' this new arrangement and we have $A(S) = A(S')$, because loci outside of $H_j^*\beta_j$ are null. Then we can rewrite every pattern as:

$$\sum_i \zeta^*\alpha_i + \sum_j \zeta^*\beta_j + G^*,$$

or an adequate permutation of this. Now simply grouping sums we write

$$\sum_i \zeta^* \gamma_i + G^*.$$

We notate $\mathcal{A}_3$ the result of making this substitution in all the possible arrangements in $\mathcal{A}_2$.

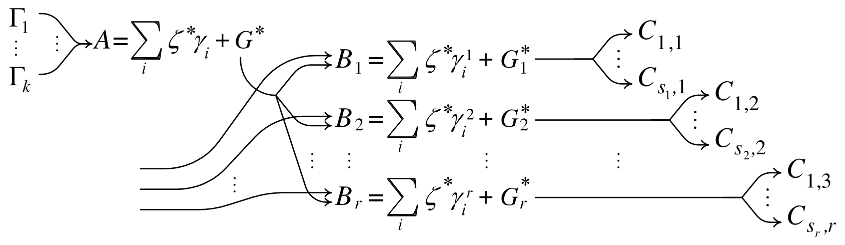
(4) It just remains to reduce the submonoid term G^*. Given an arrangement $A = \sum_i \zeta^* \gamma_i + G^*$ (or a permutation) in $\mathcal{A}_3$ we consider the following transformation of the system of arrangements defined in three subcases:

(a) If $G^* = \{1\}$, then the arrangement is projective and then we do not change anything.

(b) We suppose that $G^* \neq \{1\}$ and that we have a loop in the diagram of the mapping, i.e. the pattern G^* invokes the arrangement A itself, amongst other arrangements $B_1, \ldots, B_r$.

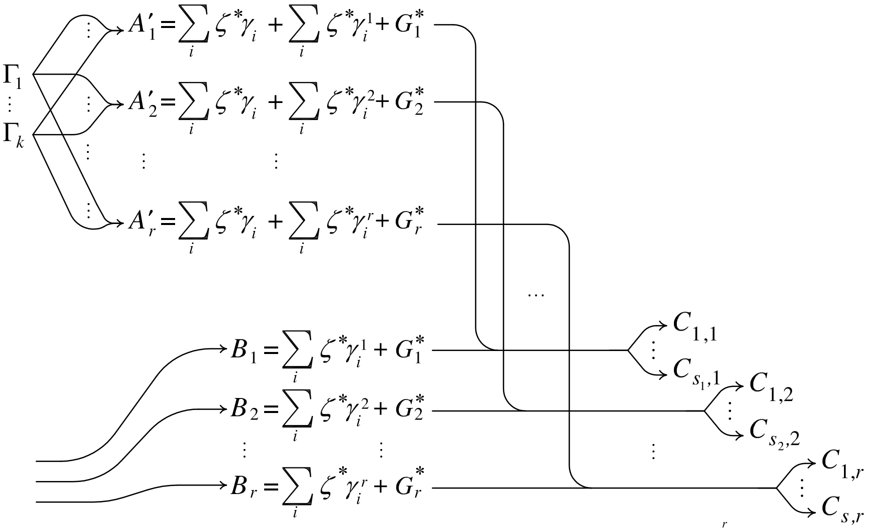
We can prove that the loop is not contributing to the linearization. Notice that if we apply the arrangement twice the result is the same. First recall that in general $S_{G^*} = SG^*$ (Proposition 6.10(iv)). Then $(S_{G^*})_{G^*} = (SG^*)_{G^*} = (SG^*)G^* = SG^* = S_{G^*}$. But then $|(S_{G^*})_{G^*}| \not< |S_{G*}|$. So necessarily S_{G^*} is atomic or null. So we can change G^* by the constant pattern 1. We can maintain the old invocations of G^*, although now these are unproductive, because the linearization always stops at this pattern 1.

(c) In the last case we suppose that $G^* \neq \{1\}$ and that G^* is invoking the arrangements $B_1, \ldots, B_r$, where $B_i \neq A, \forall i = 1, \ldots, r$, that is, there are no loops. Then we replace the configuration (with the corresponding permutations):



Γ_1
Γ_k
$A = \sum_i \zeta^* \gamma_i + G^*$
$B_1 = \sum_i \zeta^* \gamma_i^1 + G_1^*$
$B_2 = \sum_i \zeta^* \gamma_i^2 + G_2^*$
$B_r = \sum_i \zeta^* \gamma_i^r + G_r^*$
$C_{1,1}$
$C_{s_1,1}$
$C_{1,2}$
$C_{s_2,2}$
$C_{1,3}$
$C_{s_r,r}$

by the new configuration:


Γ_1
Γ_k
$A'_1 = \sum_i \zeta^*\gamma_i + \sum_i \zeta^*\gamma_i^1 + G_1^*$
$A'_2 = \sum_i \zeta^*\gamma_i + \sum_i \zeta^*\gamma_i^2 + G_2^*$
$A'_r = \sum_i \zeta^*\gamma_i + \sum_i \zeta^*\gamma_i^r + G_r^*$
$B_1 = \sum_i \zeta^*\gamma_i^1 + G_1^*$
$B_2 = \sum_i \zeta^*\gamma_i^2 + G_2^*$
$B_r = \sum_i \zeta^*\gamma_i^r + G_r^*$
$C_{1,1}$
$C_{s_1,1}$
$C_{1,2}$
$C_{s_2,2}$
$C_{1,r}$
$C_{s,r}$
r

A'_i is the result of applying B_i to A in the pattern $G*$ (importing and maintaining the convenient orders of the patterns). So the system continues defining the same linearization and in addition A'_i preserves the general form of $\mathcal{A}_3$. Notice that now we have more arrangements, but importantly, the pattern G^* has been removed from the system and in effect G_i^* satisfies that $|G_i| < |G|$. Let us see this. First we notice that we can assume that $G_i \subseteq G$. Since $(S_{G^*})_{G_i^*} = (SG^*)_{G_i^*} = (SG^*)G_i^* = S(G \cap G_i)^* = (S_{G^*})_{(G\cap G_i)^*}$, if $G_i \not\subseteq G$, then we could substitute previously all the G_i^* by $(G\cap G_i)^*$. Second we can assume that $G_i \subsetneq G$. If $G_i = G$ then $(S_{G^*})_{G_i^*} = (SG^*)_{G^*} = (SG^*)G^* = (SG^*)$, i.e. $|(S_{G^*})_{G_i^*}| = |(SG^*)G^*| = |(SG^*)|$, so this arrangement is only applicable when S is null or atomic; in this case we could substitute G_i by 1.

The cases (a), (b), (c) cover all the possibilities. We repeat the substitution (c) as many times as we can, throughout the system. If a loop appears, we apply the substitution (b) and come back to substitution (c). Since the size of the G_i's always decreases, at some moment they become empty, i.e. $\emptyset^* = 1$, and then all the arrangements are projective. We notate this final system $\mathcal{A}_4$.

To sum up, for any linearization $\Pi_{\mathcal{A}_1} \in \mathbf{Lin}(\mathbb{M}\mathbf{k})$ there are linearizations $\Pi_{\mathcal{A}_1} = \Pi_{\mathcal{A}_2} = \Pi_{\mathcal{A}_3} = \Pi_{\mathcal{A}_4}$, where the last is a linearization given by a system of arrangements in normal projective form. □

The previous proof shows indeed an algorithm to transform any linearization in $\mathbf{Lin}(\mathbb{M}\mathbf{k})$ into projective normal form. As a very simple corollary we obtain that individual linearizations in $\mathbf{Lin}(\mathbb{M}\mathbf{k})$ are projective.

Corollary 7.16. *Every global linearization $\Pi \in \mathbf{Lin}(\mathbb{M}\mathbf{k})$ is individually projective, i.e. for any $x \in \Pi(S)$ the local linearization $\ell : S \longrightarrow x$ is projective.*

Proof. Straightforward from Theorem 7.15. □

Corollary 7.17. **Lin**($\mathbb{H}$**k**) = **Lin**($\mathbb{M}$**k**) = **Lin**($\mathbb{G}$**k**).

Proof. The inclusion **Lin**($\mathbb{H}$**k**) $\subseteq$ **Lin**($\mathbb{M}$**k**) is trivial. The other inclusion is given by Theorem 7.15. Regarding **Lin**($\mathbb{M}$**k**) = **Lin**($\mathbb{G}$**k**) we notice that $\mathbb{M}$**k** $\cap$ { patterns of arity 1} = $\mathbb{G}$**k** $\cap$ { patterns of arity 1} □

Part III

APPLICATIONS

8

Lowest Positions and Properties of the Bi-Hierarchies

This chapter and the following one are devoted to studying the bi-hierarchies in itself and especially $\mathcal{BH}(\mathbf{k}, \mathbb{M})$ and $\mathcal{BH}(\mathbf{k}, \mathbb{H})$. The goal is to obtain a sketch of the character of these bi-hierarchies and their relation to the landscape of formal languages.

A remarkable property of the bi-hierarchies is so-called *bi-symmetry* which states that if we reverse the types of the manifold and the linearization simultaneously we obtain the same class of languages. In relation to this we also prove that the bi-hierarchies are symmetric, meaning that the Hasse diagram of inclusions is geometrically symmetric.

We go on to study the general bi-hierarchy. By examining the lowest positions we observe that several classes collapses to simpler cases. We prove in addition some closure properties of algebraic dependency languages.

8.1 Structure of this Chapter and the Following One

This chapter and the following one are devoted to studying general properties of the bi-hierarchies and studying some of the lowest positions in them, especially on the general and the homogeneous bi-hierarchies. The general bi-hierarchy $\mathcal{BH}(\mathbf{k}, \mathbb{M})$ contains every algebraic dependency language which means that for each $L \in \mathsf{AD}$ there is a class $X/Y \in \mathcal{BH}(\mathbf{k}, \mathbb{M})$ such that $L \in X/Y$. However given its generality this bi-hierarchy is hard to study. Notice that for every class $X/Y \in \mathcal{BH}(\mathbf{k}, \mathbb{H})$ we have $X/Y \subseteq \mu(X)/\mu(Y)$ where μ is the isomorphism of free monoids $\mu : \{\mathbf{k}, \mathbb{H}\}^* \longrightarrow \{\mathbf{k}, \mathbb{M}\}^*$, given by $\mu(\mathbf{k}) = \mathbf{k}$, $\mu(\mathbb{H}) = \mu(\mathbb{M})$. For example, $\mathbf{k}^2\mathbb{H}/\mathbb{H}\mathbf{k} \subseteq \mathbf{k}^2\mathbb{M}/\mathbb{M}\mathbf{k}$. So the homogeneous bi-hierarchy is more restricted and easier to study.

At the beginning of this chapter we study how reversing the types affects the class. Later we study the general bi-hierarchy. We examine the lowest positions and we observe that several classes collapses to simpler cases. We prove some closure properties.

The classes reviewed in this chapter seem not to be linguistically relevant since we just explore the lowest positions.[1] Fortunately, the homogeneous bi-hierarchy encapsulate more interesting classes, such us context-free languages, semi-linear languages and others. In addition some results from the general bi-hierarchy can be translated to it. This will be dealt in the following chapter.

In the sequel the form of the types inform us in which bi-hierarchy we are. So, for example, when a result uses general monoids $\mathbb{M}$, this means that what is claimed is valid for any monoid.

The known classes which we will treat in relation with the bi-hierarchies in this and the following chapter will be introduced in due course, but the diagram depicted in Fig. 8.1 summarizes their inter-inclusions.

8.2 Symmetry and Bi-Symmetry in the Bi-Hierarchies

The theorem which we are going to prove greatly simplifies the bi-hierarchies; it identifies one half of a bi-hierarchy with the other half. We have called it bi-symmetry because it uses a double symmetry in the two modules of an algebraic dependency language: manifolds and linearizations. For the moment let us notice that languages have more than one description in the bi-hierarchy; see the following example:

Example 28. Alternative classification of the copy and squares languages. We have shown that $L_{\text{copy}} \in \mathbb{H}\mathbf{k}^2/\mathbb{H}\mathbf{k}$ through the manifold W_{copy} and the linearization given by the arrangement $\zeta^*\alpha + 1 + \zeta^*\beta$. However, we can use the manifold W_{squa} with type $\mathbf{k}^2\mathbb{H}$ and the arrangement $A = \alpha\zeta^* + 1 + \beta\zeta^*$ of type $\mathbb{H}\mathbf{k}$ to obtain the same language. See the following derivation and the figure at the end:

$$\begin{aligned}
&Q_{a,b,c} \\
&\rightsquigarrow (\alpha\zeta^* + a + \beta\zeta^*) && (A)\\
&\rightsquigarrow (a + \beta\zeta^*) + a + \beta\zeta^* && (A)\\
&\rightsquigarrow a + (b + c) + a + \beta\zeta^* && (A)\\
&\rightsquigarrow a + b + c + a + (b + c) \\
&\rightsquigarrow a + b + c + a + b + c
\end{aligned}$$

Notice that, as we said at the beginning (see §5.2), the manifold W_{squa} is symmetric to the manifold W_{copy}. The arrangements $\zeta^*\alpha + 1 + \zeta^*\beta$ and $\alpha\zeta^* + 1 + \beta\zeta^*$ that we have just used are almost the same, with the difference that the involved patterns are reversed. This means that the copy language is also in the class $\mathbf{k}^2\mathbb{H}/\mathbf{k}\mathbb{H}$.

[1] And higher positions appear to be very complicated to characterize at the moment.

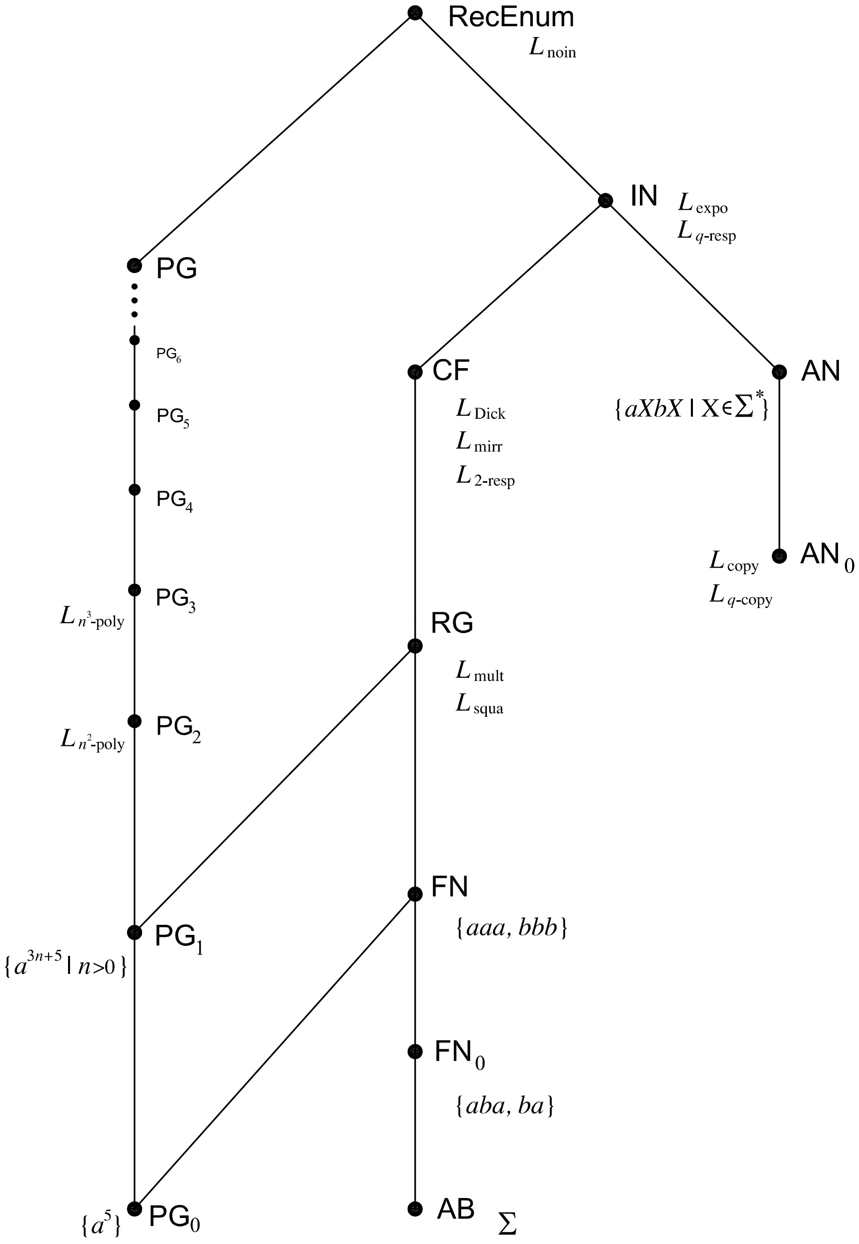
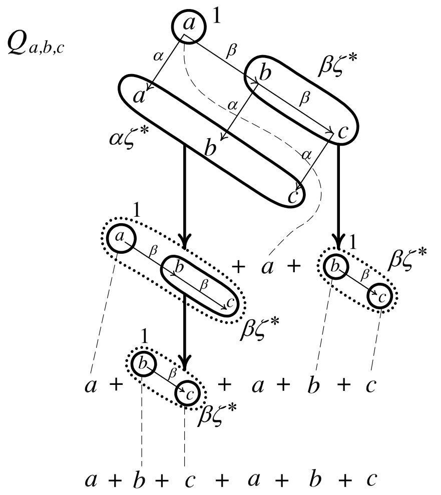


Figure 8.1: Hasse diagram of inclusions of classes of languages examined in this and the next chapter: AB alphabet lngs., FN finite lngs, FN_0 finite stressed lngs., RG regular lngs., CF context-free lngs., AN Angluin lngs., AN_0 constant-free Angluin lngs., PG_p p-polynomial growth unary lngs., IN indexed lngs., RecEnum Recursive Enumerable lngs.

Similarly we can define arrangements using patterns in $\mathbb{H}\mathbf{k}$ and obtain the squares languages from the copy manifold, whereby the squares language is also in the class $\mathbb{H}\mathbf{k}^2/\mathbb{H}\mathbf{k}$.



This example suggests that when we take the symmetric manifold and the symmetric linearization we obtain the same language. Remarkably the following theorem shows that this situation is totally general. We make a previous definition.

Definition 8.1. Given the system of arrangements:

$$\begin{cases} A_1 = \underset{\omega(1,1)}{\Gamma_{1,1}} + \cdots + \underset{\omega(m_1,1)}{\Gamma_{m_1,1}}, \\ \vdots \\ A_n = \underset{\omega(1,n)}{\Gamma_{1,n}} + \cdots + \underset{\omega(m_n,n)}{\Gamma_{m_n,n}}; \end{cases} \quad \text{we define:} \quad \begin{cases} A_1^{\mathfrak{R}} = \underset{\omega(1,1)}{\Gamma^R_{1,1}} + \cdots + \underset{\omega(m_1,1)}{\Gamma^R_{m_1,1}}, \\ \vdots \\ A_n^{\mathfrak{R}} = \underset{\omega(1,n)}{\Gamma^R_{1,n}} + \cdots + \underset{\omega(m_n,n)}{\Gamma^R_{m_n,n}}, \end{cases}$$

which we call the *symmetric system.*

Theorem 8.2. *(Bi-Symmetry) Given a classificatory monoid, for any types X, Y we have:*

$$ {}^{X}\!/_{Y} = {}^{X^{\mathfrak{R}}}\!/_{Y^{\mathfrak{R}}}.$$

Proof. Let $\mathscr{G} = (W, \Pi)$ be a grammar. We define the *symmetric grammar* as $\mathscr{G}^{\mathfrak{R}} = (W^{\mathfrak{R}}, \Pi^{\mathfrak{R}})$ where $W^{\mathfrak{R}}$ is the symmetric manifold (which by Lemma 5.8 has effectively bounded ellipticity) and where $\Pi^{\mathfrak{R}}$ is the solution of the symmetric system above defined. Clearly if $\mathscr{L}(\mathscr{G}) \in X/Y$ then $\mathscr{L}(\mathscr{G}^{\mathfrak{R}}) \in X^{\mathfrak{R}}/Y^{\mathfrak{R}}$. Our goal is to show that $\mathscr{L}(\mathscr{G}) = \mathscr{L}(\mathscr{G}^{\mathfrak{R}})$.

We prove that for any $S \in W$, $\Pi(S) = \Pi^{\mathfrak{R}}(S^{\mathfrak{R}})$, by induction on the size s of S. In fact we prove that for all the auxiliary mappings $\Pi_1, \ldots, \Pi_n$ of Π, and $\Pi_1^{\mathfrak{R}}, \ldots, \Pi_n^{\mathfrak{R}}$ of $\Pi^{\mathfrak{R}}$, we have $\Pi_j(S) = \Pi^{\mathfrak{R}}(S^{\mathfrak{R}})$ ($j = 1, \ldots, n$ in the sequel). For $s = 0$ and $s = 1$ we have atomic cases and we have trivially that $\Pi_j^{\mathfrak{R}}((a^\bullet)^{\mathfrak{R}}) = \Pi_j^{\mathfrak{R}}(a^\bullet) = a = \Pi_j(a^\bullet)$. We suppose that the statement is true for any syntagma with size $< s$ and let S be a syntagma $|S| = s$. Then we have the equalities:

$$\begin{aligned}
\Pi_j^{\mathfrak{R}}(S) &= \Pi^{\mathfrak{R}}_{\omega(1,j)}(S^{\mathfrak{R}}_{\Gamma^R_{1,j}}) + \cdots + \Pi^{\mathfrak{R}}_{\omega(m_j,j)}(S^{\mathfrak{R}}_{\Gamma^R_{m_j,j}}) \\
&= \Pi^{\mathfrak{R}}_{\omega(1,j)}((S_{\Gamma_{1,j}})^{\mathfrak{R}}) + \cdots + \Pi^{\mathfrak{R}}_{\omega(m_j,j)}((S_{\Gamma_{m_j,j}})^{\mathfrak{R}}) \\
&= \Pi_{\omega(1,j)}(S_{\Gamma_{1,j}}) + \cdots + \Pi_{\omega(m_j,j)}(S_{\Gamma_{m_j,j}}) = \Pi_j(S).
\end{aligned}$$

In the second line we have used that $S^{\mathfrak{R}}_{\Gamma^R} = (S_\Gamma)^{\mathfrak{R}}$, by Lemma 6.11. In the third line we have used the hypothesis of induction, since $|S_{\Gamma_{i,j}}| < s$. In consequence $\Pi^{\mathfrak{R}}(S^{\mathfrak{R}}) = \Pi(S)$.

Thus $\mathscr{L}(\mathscr{G}) = \bigcup_{S \in W} \Pi(S) = \bigcup_{S^{\mathfrak{R}} \in W^{\mathfrak{R}}} \Pi^{\mathfrak{R}}(S^{\mathfrak{R}}) = \bigcup_{S' \in W} \Pi^{\mathfrak{R}}(S') = \mathscr{L}(\mathscr{G}^{\mathfrak{R}})$, because the symmetry operator $(\cdot)^{\mathfrak{R}} : \mathbf{Synt} \longrightarrow \mathbf{Synt}$ is bijective.

We observed that $\mathscr{L}(\mathscr{G}) \in X/Y \implies \mathscr{L}(\mathscr{G}^{\mathfrak{R}}) \in X^{\mathfrak{R}}/Y^{\mathfrak{R}}$. But since the languages coincide, $\mathscr{L}(\mathscr{G}) = \mathscr{L}(\mathscr{G}^{\mathfrak{R}})$, we have $X/Y \subseteq X^{\mathfrak{R}}/Y^{\mathfrak{R}}$. Finally since $\mathfrak{R}$ is involutive we can also use the same result and write $X^{\mathfrak{R}}/Y^{\mathfrak{R}} \subseteq (X^{\mathfrak{R}})^{\mathfrak{R}}/(Y^{\mathfrak{R}})^{\mathfrak{R}} = X/Y$. Joining both: $X/Y = X^{\mathfrak{R}}/Y^{\mathfrak{R}}$. □

We say that two classes X/Y and $X' = Y'$ are *symmetric* iff $X^{\mathfrak{R}}/Y = X'/Y'$ or equivalently $X/Y^{\mathfrak{R}} = X'/Y'$, since, by the bi-symmetry theorem:

$$X^{\mathfrak{R}}\big/_{Y} = (X^{\mathfrak{R}})^{\mathfrak{R}}\big/_{Y^{\mathfrak{R}}} = X\big/_{Y^{\mathfrak{R}}} = X'\big/_{Y'}.$$

We say that a class is *self-symmetric* iff $X^{\mathfrak{R}}/Y = X/Y = X/Y^{\mathfrak{R}}$. The simplest classes in a bi-hierarchy are self-symmetric, for instance: $\mathbb{M}\mathbf{k}/\mathbf{k} = \mathbf{k}\mathbb{M}/\mathbf{k}$. An easy property is the following:

Proposition 8.3. *Given a class $X/Y \in \mathcal{BH}(\mathbf{T}_1, \ldots, \mathbf{T}_n)$, if $|X| + |Y| \leq 3$ then X/Y is self-symmetric.*

Proof. The proof just needs the fact that if $|X| + |Y| \leq 3$ then either $|X| \leq 1$ or $|Y| \leq 1$ and then either $X^{\mathfrak{R}} = X$ or $Y = Y^{\mathfrak{R}}$. □

There are, of course, non-self-symmetric classes. For example the classes which we have been using for classifying. In fact now we can revisit them and rewrite:

$$L_{\text{squa}} \in \mathbf{k}^2\mathbb{H}\big/_{\mathbb{H}\mathbf{k}} = \mathbb{H}\mathbf{k}^2\big/_{\mathbf{k}\mathbb{H}}, \quad L_{\text{mult}} \in \mathbf{k}\mathbb{H}\big/_{\mathbb{H}\mathbf{k}} = \mathbb{H}\mathbf{k}\big/_{\mathbf{k}\mathbb{H}}, \quad L_{\text{mirr}} \in \mathbf{k}^2\mathbb{H}\big/_{\mathbb{H}\mathbf{k}} = \mathbb{H}\mathbf{k}^2\big/_{\mathbf{k}\mathbb{H}};$$

$$L_{\text{copy}} \in {}^{\mathbb{H}\mathbf{k}^2}\!\big/_{\mathbb{H}\mathbf{k}} = {}^{\mathbf{k}^2\mathbb{H}}\!\big/_{\mathbf{k}\mathbb{H}}, \quad L_{\text{resp}} \in {}^{\mathbb{H}\mathbf{k}}\!\big/_{\mathbb{H}\mathbf{k}} = {}^{\mathbf{k}\mathbb{H}}\!\big/_{\mathbf{k}\mathbb{H}}, \quad L_{\text{mirr}} \in {}^{\mathbb{H}\mathbf{k}^2}\!\big/_{\mathbb{H}\mathbf{k}} = {}^{\mathbf{k}^2\mathbb{H}}\!\big/_{\mathbf{k}\mathbb{H}}.$$

We have the circumstance that both classes are symmetric to each other, but not self-symmetric. What about the mirror language? We know that $L_{\text{mirr}} \in \mathbf{k}^2\mathbb{H}/\mathbb{H}\mathbf{k} \cap \mathbb{H}\mathbf{k}^2/\mathbb{H}\mathbf{k} = (\mathbf{k}^2\mathbb{H} \cap \mathbb{H}\mathbf{k}^2)/\mathbb{H}\mathbf{k}$. Although this intersection does not inhabit the bi-hierarchy, it is interesting to consider it. Indeed, we have that:

$${}^{(\mathbf{k}^2\mathbb{H} \cap \mathbb{H}\mathbf{k}^2)^{\mathfrak{R}}}\!\big/_{\mathbb{H}\mathbf{k}} = {}^{(\mathbf{k}^2\mathbb{H})^{\mathfrak{R}} \cap (\mathbb{H}\mathbf{k}^2)^{\mathfrak{R}}}\!\big/_{\mathbb{H}\mathbf{k}} = {}^{\mathbb{H}\mathbf{k}^2 \cap \mathbf{k}^2\mathbb{H}}\!\big/_{\mathbb{H}\mathbf{k}} = {}^{\mathbf{k}^2\mathbb{H} \cap \mathbb{H}\mathbf{k}^2}\!\big/_{\mathbb{H}\mathbf{k}}.$$

So this class is self-symmetric.

8.3 Anti-Languages

Remark 8.4. The notation for the classes of the bi-hierarchy has an arithmetical motivation. If we notated $-X = X^{\mathfrak{R}}$, the reversed type X then bi-symmetry would be expressed as

$$\frac{-X}{-Y} = \frac{X}{Y} \quad \text{and} \quad \frac{-X}{Y} = \frac{X}{-Y}.$$

This raises the question whether sense can be made of:

$$-\frac{X}{Y}\,?$$

In this relation, this section translates the reversing operator to languages as well as types in order to obtain this third arithmetical rule and others. This leads as to the concept of *anti-languages* of a given class of languages. We will preserve during this section the notation for the reversed type as $-X$.

Let us recall that we saw in the proof of the Bi-Symmetry Theorem 8.2 that $\mathscr{L}(W,\Pi) = \mathscr{L}(W^{\mathfrak{R}},\Pi^{\mathfrak{R}})$, which is equivalent to saying $\mathscr{L}(W^{\mathfrak{R}},\Pi) = \mathscr{L}(W,\Pi^{\mathfrak{R}})$.

Definition 8.5. Let L, L' be algebraic dependency languages. We say that L and L' are *symmetric*, notated $L \perp L'$, iff there are manifolds W, W' and linearizations Π, Π' such that $L = \mathscr{L}(W,\Pi)$, $L' = \mathscr{L}'(W',\Pi')$ and such that:

$$W^{\mathfrak{R}} = W' \quad \text{and} \quad \Pi = \Pi',$$

or equivalently in virtue of the above comment:

$$W = W' \quad \text{and} \quad \Pi^{\mathfrak{R}} = \Pi'.$$

These properties mean that symmetry resembles the *orthogonality* relation in vectorial spaces. For a vector there are many orthogonal vectors. Here the situation is the same; let us see an example.

Example 29. Symmetric languages. From the examples from previous chapters we have the following symmetric languages: $L_{\text{copy}} \perp L_{\text{squa}}$ and $L_{\text{mult}} \perp L_{\text{resp}}$.

A language does not have a unique symmetric language. Let us examine the following case. First we consider another classical language called the *Dyck language* $L_{\text{Dyck}} \subseteq (\Sigma \sqcup \overline{\Sigma})^*$ where $\Sigma = \{\ (, [, \{, [\![\ \}$ is the alphabet of left parenthesis and $\overline{\Sigma} = \{\ ),], \},]\!]\ \}$ the alphabet of right parenthesis. L_{Dyck} consists in the strings of *well balanced parentheses*; see (Berstel, 1979). This language is representative of context-freeness in the sense stated in the Chomsky-Schützenberger Theorem, see (Chomsky and Schützenberger, 1963; Berstel, 1979; Autebert et al., 1997).

Now we consider a pair of languages very similar to L_{squa} and L_{copy}. We define the operator $\overline{(\cdot)} : \Sigma \longrightarrow \overline{\Sigma}$ by $\overline{(} =)$, $\overline{[} =]$, $\overline{\{} =\}$ and $\overline{[\![} =]\!]$. With this we define:

$$\widetilde{L_{\text{squa}}} = \{a_1\overline{a}_1 \cdots a_n\overline{a}_n \mid a_1 \ldots a_n \in \Sigma, n \in \mathbb{N}_+\},$$
$$\widetilde{L_{\text{copy}}} = \{a_1 \cdots a_n\overline{a}_1 \cdots \overline{a}_n \mid a_1 \cdots a_n \in \Sigma, n \in \mathbb{N}_+\}.$$

These languages are, respectively, context-free (indeed regular) and non-context-free. Of course all the three languages are algebraic dependency languages.

For the Dyck language we can construct a manifold for which the following Fig. (a) shows a syntagma. The function α opens a parenthesis, γ closes the parenthesis, β puts material inside the parenthesis while δ puts material at the right. So we take the arrangement $1 + \zeta^*\alpha + \zeta^*\beta + \zeta^*\gamma + \zeta^*\delta$. Once this syntagma is linearized it yields the string $[\,(\,)\,]\,\{\,[\![\,]\!]\,\}$. Fig. (b) depicts the symmetric syntagma which, when it is linearized with the same arrangement, yields the string $[\,(\,[\![\,\{\,]\,)\,]\!]\,\} \in \widetilde{L_{\text{copy}}}$. This shows that $L_{\text{Dyck}} \perp \widetilde{L_{\text{copy}}}$.

Fig. (a)

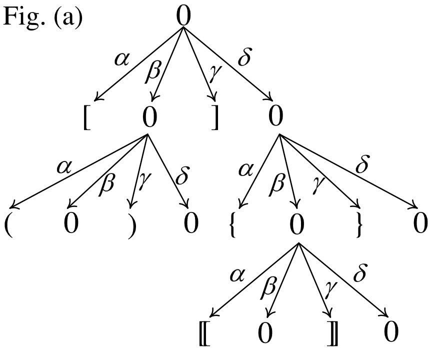
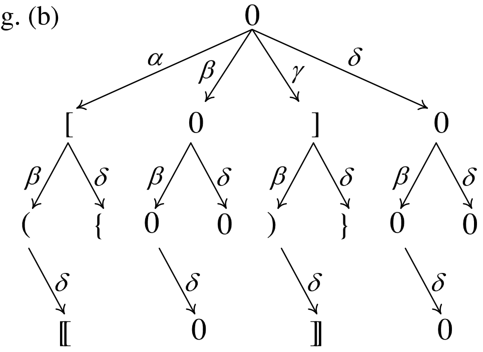


Fig. (b)



Now we consider the following Figures. Fig. (c) depicts a syntagma for a certain manifold with which the arrangement $\zeta^*\alpha + 1 + \zeta^*\beta$ yields the language $\widetilde{L_{\text{squa}}}$. In particular the syntagma from the figure yields the string $[\,]\,(\,)\,[\![\,]\!]\,\{\,\}$. Fig. (d) shows the symmetric syntagma with which the same arrangement yields the string $[\,(\,[\![\,\{\,]\,)\,]\!]\,\}\ \in \widetilde{L_{\text{copy}}}$

Fig. (c)

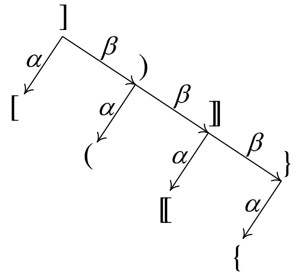
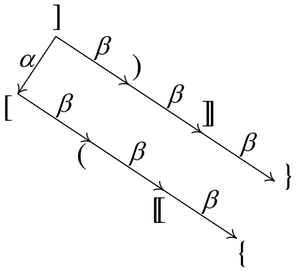


Fig. (d)



In sum, $\widetilde{L_{\text{copy}}} \perp L_{\text{Dyck}}$ and $\widetilde{L_{\text{copy}}} \perp \widetilde{L_{\text{squa}}}$, whereby a language has more than one symmetric language.

Definition 8.6. Let $\mathcal{L}$ be a class of algebraic dependency languages. We call *anti-languages of* $\mathcal{L}$, and we notate $-\mathcal{L}$ the class of languages that are symmetric to L for some language $L \in \mathcal{L}$. More formally:

$$-\mathcal{L} = \bigcup_{\text{alphabets } \Sigma} \{L \subseteq \Sigma^* \mid \exists L' \in \mathcal{L}, L' \subseteq \Sigma^* \text{ such that } L \perp L'\}.$$

Clearly $-\mathcal{L}$ is a class of languages.

Proposition 8.7. *Some properties of the "anti" operator $-\mathcal{L}$ are:*

(i) $\mathcal{L} \subseteq -(-\mathcal{L})$*;*

(ii) $\mathcal{L} \subseteq \mathcal{L}' \implies -\mathcal{L} \subseteq -\mathcal{L}'$*;*

(iii) $-(\mathcal{L} \cup \mathcal{L}') \supseteq -\mathcal{L} \cup -\mathcal{L}'$*;*

(iv) $-(\mathcal{L} \cap \mathcal{L}') \subseteq -\mathcal{L} \cap -\mathcal{L}'$*.*

Proof. (i) If $\mathscr{L}(W, \Pi) \in \mathcal{L}$ then $\mathscr{L}(W^{\mathfrak{R}}, \Pi) \in -\mathcal{L}$. Thus $\mathscr{L}(W, \Pi) = \mathscr{L}((W^{\mathfrak{R}})^{\mathfrak{R}}, \Pi) \in -(-\mathcal{L})$. i.e. $\mathcal{L} \subseteq -(-\mathcal{L})$. (ii) Let $\mathcal{L} \subseteq \mathcal{L}'$. We take $T \in -\mathcal{L}$; i.e. there is an $L \in \mathcal{L}$ such that $L \perp T$. However this means that $T \in -\mathcal{L}'$ since there is an $L \in \mathcal{L} \subseteq \mathcal{L}'$ such that $L \perp T$. (iii) $\mathcal{L} \subseteq \mathcal{L} \cup \mathcal{L}'$ and $\mathcal{L}' \subseteq \mathcal{L} \cup \mathcal{L}'$. By (ii) $-\mathcal{L} \subseteq -(\mathcal{L} \cup \mathcal{L}')$ and $-\mathcal{L}' \subseteq -(\mathcal{L} \cup \mathcal{L}')$. Then $-\mathcal{L} \cup -\mathcal{L}' \subseteq -(\mathcal{L} \cup \mathcal{L}')$. (iv) $\mathcal{L} \cap \mathcal{L}' \subseteq \mathcal{L}$ and $\mathcal{L} \cap \mathcal{L}' \subseteq \mathcal{L}'$; by (ii) $-(\mathcal{L} \cap \mathcal{L}') \subseteq -\mathcal{L}$ and $-(\mathcal{L} \cap \mathcal{L}') \subseteq -\mathcal{L}'$. Then $-(\mathcal{L} \cap \mathcal{L}') \subseteq -\mathcal{L} \cap -\mathcal{L}'$. □

When the class $\mathcal{L}$ coincides fully with a class in the bi-hierarchy the inclusion in point (i) in the last proposition becomes an equality:

Theorem 8.8 (Arithmetical properties of the bi-hierarchies)**.** *Given a classificatory monoid, for any types X, Y we have the equalities in its bi-hierarchy:*

$$\begin{aligned} {}^{X}\!/_{Y} &= {}^{-X}\!/_{-Y}, \\ -\left({}^{X}\!/_{Y}\right) &= {}^{-X}\!/_{Y} = {}^{X}\!/_{-Y}, \\ -\left(-\left({}^{X}\!/_{Y}\right)\right) &= {}^{X}\!/_{Y}. \end{aligned}$$

Proof. The first equality $X/Y = -X/-Y$ is the Bi-Symmetry Theorem 8.2 rewritten with the minus symbol instead of the reversed type. For the second equality $-(X/Y) = (-X)/Y$ we consider the equalities:

$$\begin{aligned} {}^{-X}\!/_{Y} &= \{\mathscr{L}(W', \Pi) \mid W' \in \mathbf{Man}(-X), \Pi \in \mathbf{Lin}(Y)\} \\ &= \{\mathscr{L}(W', \Pi) \mid (W')^{\mathfrak{R}} \in \mathbf{Man}(X), \Pi \in \mathbf{Lin}(Y)\} \\ &= \{\mathscr{L}(W^{\mathfrak{R}}, \Pi) \mid W \in \mathbf{Man}(X), \Pi \in \mathbf{Lin}(Y)\} \\ &= \{\mathscr{L}(W^{\mathfrak{R}}, \Pi) \mid \mathscr{L}(W, \Pi) \in X/Y\} \\ &= \{L' \mid \exists L' \in X/Y,\ L \perp L'\} \\ &= -\left({}^{X}\!/_{Y}\right). \end{aligned}$$

In the second line we have used that $W' \in \mathbf{Man}(-X) \iff (W')^{\mathfrak{R}} \in \mathbf{Man}(X)$; in the third line we have made the change of variable $W' = W^{\mathfrak{R}}$.

In order to see $-(X/Y) = X/(-Y)$ we use the previous equality and bi-symmetry: $-(X/Y) = (-X)/Y = (-(-X))/(-Y) = X/(-Y)$.

Finally for the last equality $-(-(X/Y)) = X/Y$ we use the second equality twice: $-(-(X/Y)) = -((-X)/Y) = (-(-X))/Y = X/Y$. □

Corollary 8.9. *For any classificatory monoid, the following mapping is an automorphism of orders:*

$$\begin{aligned} \mathcal{BH}(\mathbf{T}_1, \ldots, \mathbf{T}_n) &\longrightarrow \mathcal{BH}(\mathbf{T}_1, \ldots, \mathbf{T}_n) \\ {}^{X}\!/_{Y} &\longmapsto -\left({}^{X}\!/_{Y}\right) \end{aligned}$$

Proof. We saw in Proposition 8.7(i) that $-\mathcal{L}$ preserves inclusions. Since it is involutive (last equality in Theorem 8.8), this mapping is injective and surjective. □

This tells us that the Hasse diagram of a bi-hierarchy is symmetric in the geometrical sense.

8.4 The General Bi-Hierarchy $\mathcal{BH}(\mathbf{k}, \mathbb{M})$

8.4.1 The Classes $X/\mathbf{1}$

The most elementary class inhabiting the bi-hierarchy is the class of *alphabet languages* AB, i.e. languages equal to the alphabet $L = \Sigma$.

Lemma 8.10. $\mathsf{AB} \subseteq {}^{X}\!/_{Y}$ *for any types* X, Y.

Proof. We prove that $\mathsf{AB} \subseteq \mathbf{1}/\mathbf{1}$, whereby $\mathsf{AB} \subseteq X/Y$ for any types X, Y, since $\mathbf{1}/\mathbf{1} \subseteq X/Y$ for any types X, Y. Given $L = \Sigma \in \mathsf{AB}$ we build a manifold with vocabulary Σ and the set of syntactic functions $\zeta = \emptyset$; this means that the patterns can only be the trivial patterns $\{(1, \dots, 1)\}$. We already saw how to obtain atomic syntagmata as a manifold:

$$\mathbf{Atom} = \mathbf{Synt}\begin{pmatrix} \not\approx 0 \\ 1 \end{pmatrix} = \{a^\bullet \mid a \in \Sigma\}.$$

If we apply the trivial arrangement $A = 1$ we obtain the language L. The pattern involved in the manifold and the pattern of the arrangement are in $\mathbf{1}/\mathbf{1}$. □

Lemma 8.11. ${}^{X}\!/_{\mathbf{1}} \subseteq \mathsf{AB}$ *for any type* X.

Proof. $\mathbf{Lin}(\mathbf{1})$ are linearizations given by trivial arrangements $A = 1$. So the manifold of a grammar can only contain atomic syntagmata. Therefore the language of the grammar is just the alphabet Σ and then $X/\mathbf{1} \subseteq \mathsf{AB}$. □

Theorem 8.12. $\mathsf{AB} = {}^{\mathbf{1}}\!/_{\mathbf{1}}$.

Proof. By the above lemmas, $\mathbf{1}/\mathbf{1} \subseteq \mathsf{AB} \subseteq \mathbf{1}/\mathbf{1}$. □

8.4.2 The classes $X/\mathbb{M}^q$

Proposition 8.13. ${}^{X}\!/_{\mathbb{M}^q} = \mathsf{AB}$ *for any* $q \geq 0$ *and for any type* X.

Proof. The direction $(\supseteq)$ is trivial by Lemma 8.10. $(\subseteq)$ Let $L = \mathscr{L}(W, \Pi)$ be a language in $X/\mathbb{M}^q$. Suppose that $\zeta \neq \emptyset$, otherwise the unique possible arrangements are $A = 1$ and we are done.

We pick an $S \in W$ and consider the first arrangement applied to S to obtain a linearization, say $A = \Gamma_1 + \cdots + \Gamma_n$. Since $\Gamma_1, \dots, \Gamma_n \in \mathbb{M}^q$ then $1 \in \Gamma_i$ for any $i = 1, \dots, n$ which implies that $1 \in \Gamma_1 \cap \cdots \cap \Gamma_n$. First notice that the patterns in an arrangement must be disjoint. Thus the only possibility to consider is $n = 1$, which makes the arrangement $A = \Gamma$. Now notice that the patterns in an arrangement must cover the support of the syntagma, so $\mathrm{Spt}(S) \subseteq \Gamma$. However this means that $S_\Gamma = S_{\zeta^*} = S$ and therefore $|S_\Gamma| \nless |S|$. This means that S is always atomic or null, and thus the system of arrangements just effects the root. The arrangement A can be substituted by the arrangement $A' = 1$ and no more arrangements are needed. This system of arrangements can only linearize atomic syntagmata, so $L \in \mathsf{AB}$. □

8.4.3 The classes $\mathbf{k}^p/X$

Proposition 8.14. $^{\mathbf{k}^p}\!\!/_X = \mathsf{AB}$ *for any* $p \geq 0$ *and for any type* X.

Proof. ($\supseteq$) is trivial. ($\subseteq$). We saw in Example 11 that almost no manifold in $\mathbf{Man}(\mathbf{k}^p)$ has effectively bounded ellipticity. The unique exception is by considering that $\zeta = \emptyset$ and then the manifold consists in atomic syntagmata. So the unique language definable in this class is in AB. □

8.4.4 The Classes $X/\mathbf{k}^p$

Let FN stand for the class of finite languages.

Theorem 8.15. $\mathsf{FN} = {}^{\mathbf{k}^2\mathbb{M}}\!\!/_{\mathbf{k}}$.

Proof. Let us see the inclusion $\mathsf{FN} \subseteq \mathbf{k}^2\mathbb{M}/\mathbf{k}$. Let there be a string $x = c_0 \cdots c_t$, with $t \geq 0$. First we build a manifold with only one syntagma of depth 1 which when it is linearized yields the string x. Consider $n \geq t$ syntactic functions $\zeta = \{\lambda_1, \ldots, \lambda_n\}$ and let W_x^n be the manifold:

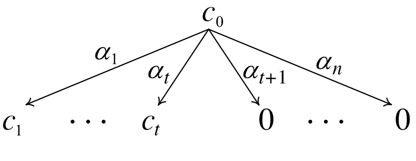
$$W_x^n = \mathbf{Synt}\begin{pmatrix} \approx c_0 & \approx c_1 & \cdots & \approx c_t & \approx 0 & \cdots & \approx 0 \\ 1 & \lambda_1 & \cdots & \lambda_t & \lambda_{t+1} & \cdots & \lambda_n \end{pmatrix}.$$

Consider the manifold of non-ellipticity **Nell** and finally consider the manifold with null loci of length greater than 1:

$$V = \bigcap_{j=1}^{n} \mathbf{Synt}\begin{pmatrix} \approx 0 & \cdots & \approx 0 \\ \lambda_1\lambda_j & \cdots & \lambda_n\lambda_j \end{pmatrix}.$$

Nell $\cap\, V$ grants that the syntagma has depth 1, while W_x^n fills out all the letters. So $W_x^n \cap \mathbf{Nell} \cap V$ has a unique syntagmata with all the letters of x, and maybe some zeros. We notate this syntagma S_x which looks as follows:

$S_x =$



We construct an arrangement which uses only patterns in $\mathbf{k}$: $A(S) = 1 + \lambda_1 + \cdots + \lambda_n$. We have that $A(S_x) = x$. Now we consider a finite language L, with $n = \max_{x \in L} |x|$ and we let there be the intersection of manifolds:

$$W_L = \mathbf{Nell} \cap V \cap \bigcup_{x \in L} W_x^n,$$

which contains as many syntagmata as strings in L. If we use the same arrangements above we obtain the language L. We just have to check that W_L has type $\mathbf{k}^2\mathbb{M}$. This is so because **Nell** has type $\mathbf{k}\mathbb{M}$, V has type $\mathbf{k}^2$ and W_x^n has type $\mathbf{k}$. So the full manifold has type $\mathbf{k}^2\mathbb{M}$. Since the arrangement has type $\mathbf{k}$, we have that $\mathsf{FN} \subseteq \mathbf{k}^2\mathbb{M}/\mathbf{k}$.

On the other hand if a language is in $X/\mathbf{k}$ then its strings are linearized by constant patterns. Any other non-null locus of greater depth could not be captured by any arrangement. So all the syntagmata from a manifold have depth ≤ 1. In consequence the manifold is finite which implies that the language is finite. Since this is valid for any X, we have $\mathbf{k}^2\mathbb{M}/\mathbf{k} \subseteq \mathsf{FN}$. □

We consider a subclass of finite languages. We say that a finite language L is *finite stressed* iff there are two alphabets Σ and $\underline{\Sigma}$ with $\underline{\Sigma} \cap \Sigma = \emptyset$ such that every string in $L \subseteq (\Sigma \cup \underline{\Sigma})^*$ contains exactly one letter in $\underline{\Sigma}$. We notate FN_0 the class of these languages which forms a strict subclass of finite languages $\mathsf{FN}_0 \subset \mathsf{FN}$ as proves the language $\{a^2\} \notin \mathsf{FN}_0$.

We call such languages "stressed" because they can formalize a decomposition of words in syllables with one stressed syllable. For example we take the alphabets $\underline{\Sigma} = \{\underline{\text{tu}}, \underline{\text{crash}}, \underline{\text{clo}}, \underline{\text{car}}, \underline{\text{ti}}\}$ and $\Sigma = \{\text{tu}, \text{sed}, \text{pet}, \text{com}, \text{pe}, \text{tion}\}$. The following language is in FN_0:

$$L = \{\text{tu}\cdot\underline{\text{tu}}, \underline{\text{crash}}, \underline{\text{clo}}\cdot\text{sed}, \underline{\text{car}}\cdot\text{pet}, \text{com}\cdot\text{pe}\cdot\underline{\text{ti}}\cdot\text{tion}\}$$

In addition notice that if we take $\underline{\Sigma} \neq \emptyset, \Sigma = \emptyset$ this, together with the condition that every string in a finite language $L \subseteq (\Sigma \cup \underline{\Sigma})^*$ contains just one letter in $\underline{\Sigma}$, yields an alphabet language $L \subseteq \underline{\Sigma} \in \mathsf{AB}$. Thus $\mathsf{AB} \subset \mathsf{FN}_0$. The strictness is trivial.

Theorem 8.16. $\mathsf{FN}_0 \subseteq {}^{\mathbf{k}\mathbb{M}}\!/_{\mathbf{k}}$.

Proof. Let $L \in \mathsf{FN}_0$ be a finite stressed language. Every string in $z \in L$ contains just one letter $\underline{b} \in \underline{\Sigma}$ which splits the string z in three parts, $z = x \cdot \underline{b} \cdot y$, with $x, y \in \Sigma^*$. We call x *the first part of* z and y the *second part of* z. We take $\zeta = \{\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_m\}$, where $n = \max\{|x| \mid x \text{ is the first part of } x \text{ for some } z \in L\}$ and $m = \max\{|y| \mid y \text{ is the second part of } y \text{ for some } z \in L\}$.

For every string $z \in L$ with $z = a_1 \cdots a_{n'} \cdot \underline{b} \cdot c_1 \cdots c_{m'}$ where $a_1 \cdots a_p \in \Sigma^*$ is the first part ($n' \leq n$), $\underline{b} \in \underline{\Sigma}$ and $c_1 \cdots c_{m'} \in \Sigma^*$ is the second part ($m' \leq m$), we take the manifold:

$$W_z = \mathbf{Synt}\begin{pmatrix} \approx a_1 \cdots \approx a_{n'} & \approx 0 \ \cdots \approx 0 & \approx \underline{b} & \approx c_1 \cdots \approx c_{m'} & \approx 0 \cdots \approx 0 \\ \alpha_1 \ \cdots \ \alpha_{n'} & \alpha_{n'+1} \cdots \alpha_n & 1 & \beta_1 \ \cdots \ \beta_{m'} & \beta_{m'+1} \cdots \beta_m \end{pmatrix}.$$

Notice that the following manifold:

$$W_z \cap \mathbf{Nell} \cap \bigcap_{\lambda \in \zeta} \mathbf{Synt}\begin{pmatrix} (x \in \Sigma) \longrightarrow (y \approx 0) \\ (1, \lambda)(\zeta)_2^* \end{pmatrix},$$

contains a unique syntagma which we can linearize with the arrangement $\sum_{i=1}^{n} \alpha_i + 1 + \sum_{j=1}^{m} \beta_j$ in order to obtain the string:

$$z = a_1 + \cdots + a_{n'} + 0 + \cdots + 0 + b + c_1 + \cdots + c_{m'} + 0 + \cdots + 0,$$

or in multiplicative notation: $z = a_1 \cdots a_{n'} 1 \cdots 1 b c_1 \cdots c_{m'} 1 \cdots 1 = a_1 \cdots a_{n'} b c_1 \cdots c_{m'}$. Clearly this arrangement describe a linearization in $\mathbf{Lin}(\mathbf{k})$. In order to describe the language L we only have to consider the join of manifolds:

$$\bigcup_{z \in L} \left(W_z \cap \mathbf{Nell} \cap \bigcap_{\lambda \in \zeta} \mathbf{Synt} \begin{pmatrix} (x \in \Sigma) \longrightarrow (y \approx 0) \\ (1, \lambda)(\zeta)^*_2 \end{pmatrix} \right) \in \mathbf{Man}(\mathbf{k}\mathbb{M}).$$

□

Lemma 8.17. ${}^{X}\!/_{\mathbf{k}^p} \subseteq \mathsf{FN}$ *for any* $p \geq 0$ *and for any type* X.

Proof. Any arrangement in $\mathbf{Lin}(\mathbf{k}^p)$ has the form $A = \varphi_1 + \cdots + \varphi_s$ with $\varphi_i \in \mathbf{k}^p$. So the manifold of a grammar can only contain syntagmata with depth $\leq p$. Since the set of syntactic functions is finite we have $|S| \leq \sum_{k=1}^{p} |\zeta|^k$. Since all the syntagmata are bounded, the manifold must be finite. So any language in $X/\mathbf{k}^p$ is finite. □

Corollary 8.18. *For any type* X *and for any* $p \geq 1$, *if* $\mathbf{k}^2$, $\mathbb{M} \sqsubseteq X$ *then* ${}^{X}\!/_{\mathbf{k}^p} = \mathsf{FN}$.

Proof. We know that $X/\mathbf{k} \subseteq \mathsf{FN}$. Notice that if $\mathbf{k}^2 \sqsubseteq X$ then we can define the manifolds V and W_x in the proof of Theorem 8.15. If $\mathbb{M} \sqsubseteq X$, since $\mathbf{k}^2 \sqsubseteq X$ then either $\mathbf{k}\mathbb{M} \sqsubseteq X$ or $\mathbb{M}\mathbf{k} \sqsubseteq X$. In the first case we can define the manifold **Nell** and we are done. In the second case we observe that $X/\mathbf{k}$ is self-symmetric, so we can reverse X to obtain $\mathbf{k}\mathbb{M} \sqsubseteq X^{\mathfrak{R}}$. □

Corollary 8.19. ${}^{\mathbf{k}\mathbb{M}^p}\!/_{\mathbf{k}^q} = {}^{\mathbb{M}^p\mathbf{k}}\!/_{\mathbf{k}^q}$ *for any* $p, q \geq 0$.

Proof. Straightforward by bi-symmetry. □

These results reduce the classes $X/\mathbf{k}^p$ to the following cases:

$${}^{\mathbf{k}^2\mathbb{M}}\!/_{\mathbf{k}} = \mathsf{FN}, \quad {}^{\mathbf{1}}\!/_{\mathbf{1}} = \mathsf{AB}, \quad {}^{\mathbb{M}^p}\!/_{\mathbf{k}^q}, {}^{\mathbf{k}\mathbb{M}^p}\!/_{\mathbf{k}^q}, {}^{\mathbb{M}^p\mathbf{k}\mathbb{M}^q}\!/_{\mathbf{k}^r} \subseteq \mathsf{FN}.$$

8.5 Closure Properties of $\mathcal{BH}(\mathbf{k}, \mathbb{M})$

In the following we consider all the manifolds with effectively bounded ellipticity. Manifolds built from these inherits the bounds.

Suppose $\zeta \subseteq \xi$, we write $\zeta' = \xi \setminus \zeta$. Consider the mapping $\eta : \mathbf{Synt}_{\Sigma,\zeta} \longrightarrow \mathbf{Synt}_{\Sigma,\zeta'}$ given by:

$$\eta(S)(x) = \begin{cases} S(x) & \text{if } x \in \zeta^*; \\ 0 & \text{otherwise.} \end{cases}$$

If $W = \mathbf{Synt}_{\Sigma,\zeta}\binom{B}{\Gamma} \subseteq \mathbf{Synt}_{\Sigma,\zeta}$ is a manifold, W can be embedded by η and it is still a manifold which can be calculated as:

$$\eta(W) = \mathbf{Synt}_{\Sigma,\xi}\begin{pmatrix} B \\ \Gamma \end{pmatrix} \cap \bigcap_{\alpha' \in \zeta'} \mathbf{Synt}_{\Sigma,\xi}\begin{pmatrix} \approx 0 \\ \xi^* \alpha' \zeta^* \end{pmatrix}.$$

The second manifold is necessary in order to make null all the loci in $\xi^* \setminus \zeta^*$. So if $\Gamma \in X$ and $\mathbb{M}\mathbf{k}\mathbb{M} \sqsubseteq X$ then $\eta(W) \in X$. In the case that W is non-elliptic we can simplify the expression of the manifold:

$$\eta(W) = \mathbf{Synt}_{\Sigma,\xi}\begin{pmatrix} B \\ \Gamma \end{pmatrix} \cap \bigcap_{\alpha' \in \zeta'} \mathbf{Synt}_{\Sigma,\xi}\begin{pmatrix} \approx 0 \\ \alpha' \zeta^* \end{pmatrix} \cap \mathbf{Nell}_{\Sigma,\xi}.$$

Now if $\Gamma \in X$ and $\mathbf{k}\mathbb{M} \sqsubseteq X$ then $\eta(W) \in X$.

Theorem 8.20. *If X is a type such that $\mathbb{M}\mathbf{k}\mathbb{M} \sqsubseteq X$, then the class X/Y is closed under unions, i.e.:*

$$L_1, L_2 \in {}^{X}\!/_{Y} \implies L_1 \cup L_2 \in {}^{X}\!/_{Y}.$$

Proof. Consider the languages $L_1 = \mathscr{L}(W_1, \Pi_1), L_2 = \mathscr{L}(W_2, \Pi_2)$ such that $L_1, L_2 \in X/Y$. Without lost of generality we can suppose that W_1 uses the set of syntactic functions ζ_1 and W_2 uses the set ζ_2 and that $\zeta_1 \cap \zeta_2 = \emptyset$ (otherwise we rename the syntactic functions). Consider the two embeddings: $\eta_1 : \mathbf{Synt}_{\Sigma,\zeta_1} \longrightarrow \mathbf{Synt}_{\Sigma,\zeta_1 \sqcup \zeta_2}$ and $\eta_2 : \mathbf{Synt}_{\Sigma,\zeta_2} \longrightarrow \mathbf{Synt}_{\Sigma,\zeta_1 \sqcup \zeta_2}$. Then we consider the manifold:

$$W_1 \smile W_2 = \eta_1(W_1) \cup \eta_2(W_2)$$

as in Fig. (a), and the system of arrangements $\Pi_1 \smile \Pi_2$ given by the mapping $\omega_1 \smile \omega_2$ as in Fig. (b).

Fig. (a)

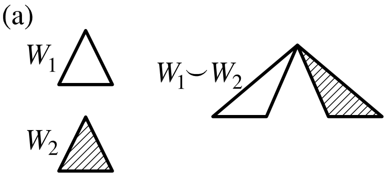
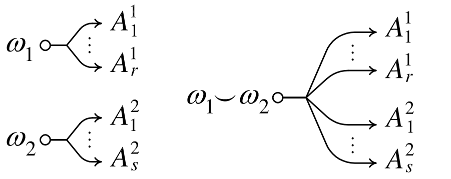


Fig. (b)



Then $\mathscr{L}(W_1 \smile W_2, \Pi_1 \smile \Pi_2) = L_1 \cup L_2$. On the one hand since $W_1, W_2 \in \mathbf{Man}(X)$ and $\mathbb{M}\mathbf{k}\mathbb{M} \sqsubseteq X$ by assumption, and taking account the comment made before, then $W_1 \smile W_2 = \eta(W_1) \cup \eta(W_2) \in \mathbf{Man}(X)$. On the other hand clearly if $\Pi_1, \Pi_2 \in \mathbf{Lin}(Y)$ then $\Pi_1 \smile \Pi_2 \in \mathbf{Lin}(Y)$. □

Definition 8.21. We say that an algebraic dependency language is a *non-elliptic language* if there is a non-elliptic manifold and a system of arrangements defining it.

Corollary 8.22. *Let X be a type such that $\mathbb{M}\mathbf{k} \sqsubseteq X$. If L_1, L_2 are non-elliptic languages in X/Y then $L_1 \cup L_2 \in {}^{X}\!/_{Y}$.*

Proof. The proof is the same as in the last theorem. However at the end we use the fact that manifolds are non-elliptic and the comment made before Theorem 8.20. □

Theorem 8.23. *Let X, Y be types satisfying that $\mathbb{M}\mathbf{k}\mathbb{M} \sqsubseteq X$ and $\mathbb{M}\mathbf{k} \sqsubseteq Y$. We have:*

$$L_1, \ldots, L_n \in {}^{X}\!/_{Y} \implies L_1 \cdots L_n \in {}^{X\mathbf{k}}\!/_{Y}.$$

Proof. Consider the languages $L_i = \mathscr{L}(W_i, \Pi_i)$, such that $W_i \in \mathbf{Man}(X)$ and $\Pi_i \in \mathbf{Lin}(Y)$ for any $i = 1, \ldots, n$. We show the proof for the case $n = 2$; the case $n > 2$ generalizes easily. Consider the manifolds $W_i = \mathbf{Synt}_{\Sigma_i,\zeta_i}\binom{B_i}{\Gamma_i}$ with syntactic functions ζ_i and vocabulary Σ_i. Without lost of generality we can suppose that ζ_1, ζ_2 are disjoint. We take $\zeta = \zeta_1 \sqcup \zeta_2 \sqcup \{\beta_1, \beta_2\}$ (where β_1, β_2 are new functions), $\Sigma = \Sigma_1 \cup \cdots \cup \Sigma_n$ and construct the manifold:

$$\begin{aligned} W = \mathbf{Synt}_{\Sigma,\zeta}\begin{pmatrix} B_1 \\ \Gamma_1(\beta_1)_{a_1} \end{pmatrix} \cap \bigcap_{\alpha_1 \in \zeta} \mathbf{Synt}_{\Sigma,\zeta}\begin{pmatrix} \approx 0 \\ \zeta^*\alpha_1\zeta_1^*\beta_1 \end{pmatrix} \cap \\ \mathbf{Synt}_{\Sigma,\zeta}\begin{pmatrix} B_2 \\ \Gamma_2(\beta_2)_{a_2} \end{pmatrix} \cap \bigcap_{\alpha_2 \in \zeta} \mathbf{Synt}_{\Sigma,\zeta}\begin{pmatrix} \approx 0 \\ \zeta^*\alpha_2\zeta_2^*\beta_2 \end{pmatrix} \cap \\ \bigcap_{\alpha \in \zeta_1 \sqcup \zeta_2} \mathbf{Synt}_{\Sigma,\zeta}\begin{pmatrix} \approx 0 \\ \zeta^*\alpha \end{pmatrix} \cap \mathbf{Synt}_{\Sigma,\zeta}\begin{pmatrix} \approx 0 \\ 1 \end{pmatrix}, \end{aligned}$$

where a_i is the arity of Γ_i, $i = 1, 2$. We have $\Gamma_i\alpha_i \in X\mathbf{k}$, since $\Gamma_i \in X$. And since $\mathbb{M}\mathbf{k}\mathbb{M} \sqsubseteq X$, $\mathbb{M}\mathbf{k}\mathbb{M}\mathbf{k} \sqsubseteq X\mathbf{k}$ and then $W \in \mathbf{Man}(X\mathbf{k})$. Now we consider the set of all the arrangements headed by a new arrangement $1 + \zeta_1^*\beta_1 + \zeta_2^*\beta_2$; see the new system of arrangements in Fig.(b):

Fig. (a)

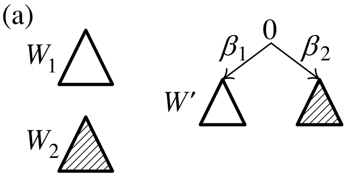
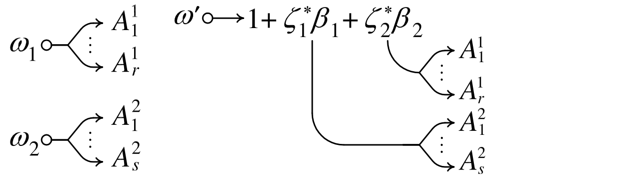


Fig. (b)



Since we have assumed that $\mathbb{M}\mathbf{k} \sqsubseteq Y$ we have that this gives a new linearization $\Pi \in \mathbf{Lin}(Y)$. Finally $L_1 \cdot L_2 = \mathscr{L}(W, \Pi) \in X\mathbf{k}/Y$. □

Theorem 8.24. *Let X, Y be types satisfying that $\mathbb{M}\mathbf{k}^2 \sqsubseteq X$ and $\mathbb{M}\mathbf{k} \sqsubseteq Y$. If L is a language such that $0 \in L$, we have:*

$$L \in {}^{X}\!/_{Y} \implies L^* \in {}^{X\mathbb{M}}\!/_{Y}.$$

Proof. Consider the language $L = \mathscr{L}(W,\Pi)$, such that $W = \mathbf{Synt}_{\Sigma,\zeta}\binom{B}{\Gamma} \in \mathbf{Man}(X)$ and $\Pi \in \mathbf{Lin}(Y)$. We take the set $\zeta' = \zeta \sqcup \{\lambda\}$ and we construct the following manifold:

$$W' = \mathbf{Synt}_{\Sigma,\zeta'}\begin{pmatrix} B \\ \Gamma(\lambda)^*_a \end{pmatrix} \cap \bigcap_{\alpha\in\zeta} \mathbf{Synt}_{\Sigma,\zeta'}\begin{pmatrix} \approx 0 \\ \zeta'^*\lambda\alpha\lambda^* \end{pmatrix},$$

where a is the arity of Γ. This manifold contains syntagmata as in Fig.(a). The condition of $0 \in L$ is required in order to avoid that the chain of subsyntagmata (shaded triangles in the figure) grows indefinitely. Notice that the first manifold in the definition of W' chains syntagmata of W through the function λ. If we supposed that $0^\bullet \notin W$ then chaining could not stop, which would make $W' = \emptyset$. However since $0 \in L$, we have $0^\bullet \in W$. Regarding the types, if $\mathbb{M}\mathbf{k}^2 \sqsubseteq X$ then $\mathbb{M}\mathbf{k}^2\mathbb{M} \sqsubseteq X\mathbb{M}$ and then $W \in \mathbf{Man}(X\mathbb{M})$.

Fig. (a)

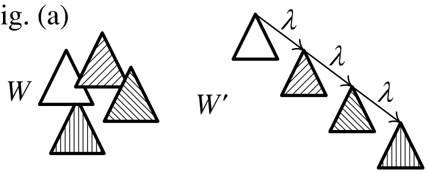
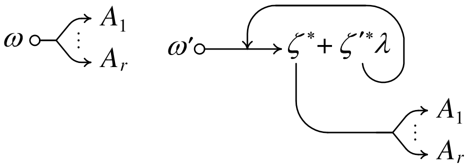


Fig. (b)



Now we take all the arrangements in Π and a new arrangement $\zeta^* + \zeta'^*\lambda$ with the mapping ω' given in Fig. (b). This yields a linearization $\Pi' \in \mathbf{Lin}(Y)$ since we have assumed that $\mathbb{M}\mathbf{k} \sqsubseteq Y$. So $\mathscr{L}(W',\Pi') = L^* \in X\mathbb{M}/Y$. □

The condition $0 \in L$ can be removed from the last theorem.

Corollary 8.25. *Let X, Y be types satisfying that $\mathbb{M}\mathbf{k}^2 \sqsubseteq X$ and $\mathbb{M}\mathbf{k} \sqsubseteq Y$. We have:*

$$L \in {}^{X}\!/_{Y} \implies L^* \in {}^{X\mathbb{M}}\!/_{Y}.$$

Proof. Consider the language $L = \mathscr{L}(W,\Pi)$, such that $W = \mathbf{Synt}_{\Sigma,\zeta}\binom{B}{\Gamma} \in \mathbf{Man}(X)$ and $\Pi \in \mathbf{Lin}(Y)$. Then we have $L \cup \{0\} = \mathscr{L}(W \cup \{0^\bullet\},\Pi)$. The manifold $W \cup \{0^\bullet\}$ can be calculated as $W \cup \{0^\bullet\} = W \cup \mathbf{Synt}\binom{\approx 0}{\zeta^*}$. Since $\mathbb{M}\mathbf{k}^2 \sqsubseteq X$, in particular $\mathbb{M} \sqsubseteq X$ whereby $W \cup \{0^\bullet\} \in \mathbf{Man}(X)$ and then $L \cup \{0\} \in X/Y$. We apply the last Theorem 8.24 and we have that $(L \cup \{0\})^* \in X/Y$. Now simply notice that $(L \cup \{0\})^* = L^*$. □

Theorem 8.26. *Let X, Y be any types:*

$$\textit{if}\, L \in {}^{X}\!/_{Y} \implies L^R \in {}^{X}\!/_{Y}.$$

Proof. Let L be the language $L = \mathscr{L}(W,\Pi)$. Given an arrangement $A = \sum_{i=1}^n \Gamma_i$ we can construct the *reversed arrangement* $A^R = \sum_{i=1}^n \Gamma_{n-i}$.[2] Then we can construct the

[2] We must not confuse the reversed arrangement A^R with the symmetric arrangements of the symmetric system $A^{\Re} = \sum_{i=1}^n \Gamma_i^R$ (see Definition 8.1 and Theorem 8.2): here we reverse the order of the patterns, while in the symmetric arrangement we reversed patterns, but not the order.

reversed linearization Π^R defined by the system of reversed arrangements with the same mapping ω. We have to prove that:

$$\big(\mathscr{L}(W,\Pi)\big)^R = \mathscr{L}(W,\Pi^R).$$

We see the direction ($\subseteq$), that is, if a succession of arrangements is applied to a syntagma S to obtain the string x, applying to S the same succession of arrangements but reversed yields the string x^R. This can be shown by induction on the size of syntagmata. When the syntagmata are null or atomic the statement is trivial. Suppose now that the statement is true for any syntagma of size n and let S be a syntagma with size $n+1$ and such that S yields the string x after the application of some arrangements. Let $A(S) = \sum_{i=1}^{n} S_{\Gamma_i}$ be the first arrangement to be applied.

Let x_i be the string resulting from linearizing every S_{Γ_i}. Since $|S_{\Gamma_i}| < |S|$ we can use the hypothesis of induction. When we apply to each subsyntagma the remaining arrangements in the reversed form we obtain the strings x_i^R. So if we have the linearization (using non-reversed arrangements):

$$S \rightsquigarrow S_{\Gamma_1} + \cdots + S_{\Gamma_n} \rightsquigarrow x_1 + \cdots + x_n = x,$$

then we have the linearization (using reversed arragements):

$$S \rightsquigarrow S_{\Gamma_n} + \cdots + S_{\Gamma_1} \rightsquigarrow x_n^R + \cdots + x_1^R = x^R.$$

This proves that $\big(\mathscr{L}(W,\Pi)\big)^R \subseteq \mathscr{L}(W,\Pi^R)$. In order to see the other direction ($\supseteq$) first we use the result just proved to obtain: $\big(\mathscr{L}(W,\Pi^R)\big)^R \subseteq \mathscr{L}(W,(\Pi^R)^R) = \mathscr{L}(W,\Pi)$. Now we notice that in general if $L \subseteq L'$ then $L^R \subseteq L'^R$. Hence $\big((\mathscr{L}(W,\Pi^R))^R\big)^R \subseteq \big(\mathscr{L}(W,\Pi)\big)^R$, or what is the same $\mathscr{L}(W,\Pi^R) \subseteq \big(\mathscr{L}(W,\Pi)\big)^R$.

Finally it is obvious that if $\Pi \in \mathbf{Lin}(Y)$ then $\Pi^R \in \mathbf{Lin}(Y)$. So if $L \in X/Y$ then $L^R \in X/Y$. □

If a homomorphism of monoids $f : \Sigma^* \longrightarrow \Sigma'^*$ is length-preserving, i.e. $|f(x)| = |x|$, then f is determined by the restriction $f : \Sigma \longrightarrow \Sigma'$. In particular f is 0-free (that is $f^{-1}(0) = \{0\})$.[3]

Theorem 8.27. *For any types X, Y and for any length-preserving homomorphism f we have:*

$$L \in {}^{X}\!/\!_{Y} \implies f^{-1}(L) \in {}^{X}\!/\!_{Y}.$$

Proof. Let $f : \Sigma^* \longrightarrow \Sigma'^*$ be a homomorphism. The syntagma given by the composition $f \circ S$ makes sense. Consider $L \in X/Y$, $L = \mathscr{L}(W,\Pi)$, $W \in \mathbf{Man}(X)$, $\Pi \in \mathbf{Lin}(Y)$.

Let x be a string of L and suppose that x is obtained from the syntagma S_x and certain succession of arrangements in Π.

[3] 0-free homomorphisms are also called ϵ-free or λ-free homomorphisms, (Hopcroft et al., 2001).

If f preserves the length, the restriction $f : \Sigma \longrightarrow \Sigma'$ is well-defined and then $f(\Sigma) \subseteq \Sigma'$. We are going to see that each string $y \in f^{-1}(x)$ can be obtained from some syntagma S_y such that $f \circ S_y = S_x$ by the same succession of arrangements. First we notice that if $x \in \Pi(S)$ then $f(x) \in \Pi(f \circ S)$ (this can be proved by induction on the size of S as on other occasions). Since f preserves the length, if we apply f to S, it does not remove any node and then S and $f \circ S$ have the same shape up to the letters. So during the linearization the same arrangements are invoked equally for S and for $f \circ S$. The only difference lies in the atomic syntagmata: in the case of $f \circ S$ these become $(f(a))^\bullet$. All this means that the language $f^{-1}(L)$ can be described as:

$$f^{-1}(L) = \mathscr{L}\big(\{S \in \mathbf{Synt} \mid f \circ S \in W\}, \Pi\big).$$

However this expression is sound only if the set of syntagmata is indeed a manifold. We notate $f_a(x_1, \ldots, x_a) = (f(x_1), \ldots, f(x_a))$. The mapping $B \circ f_a$, where a is the arity of B is always well-defined. We write $W = \mathbf{Synt}\left(\begin{smallmatrix} B \\ \Gamma \end{smallmatrix}\right)$ and $W_f = \mathbf{Synt}\left(\begin{smallmatrix} B \circ f_a \\ \Gamma \end{smallmatrix}\right)$. Notice that we have the equivalences:

$$\begin{aligned}
f \circ W &\iff f \circ S \in \mathbf{Synt}\begin{pmatrix} B \\ \Gamma \end{pmatrix} \\
&\iff \forall (x_1, \ldots, x_a) \in \Gamma,\ B(f \circ S(x_1), \ldots, f \circ S(x_a)) = 1 \\
&\iff \forall (x_1, \ldots, x_a) \in \Gamma,\ (B \circ f_a)(S(x_1), \ldots, S(x_a)) = 1 \\
&\iff S \in \mathbf{Synt}\begin{pmatrix} B \circ f_a \\ \Gamma \end{pmatrix} \\
&\iff S \in W_f.
\end{aligned}$$

This means that the set $\{S \in \mathbf{Synt} \mid f \circ S \in W\} = W_f$. So, if $W \in \mathbf{Man}(X)$ then $W_f \in \mathbf{Man}(X)$ and then $f^{-1}(L) \in X/Y$. □

The following corollary sums up all the results viewed in this section.

Corollary 8.28. *The class of algebraic dependency languages is closed under unions, products, Kleene star, reversal, and length-preserving inverse homomorphism. That is if $L, L' \in$ AD then*

$$L \cup L',\ \ L{\cdot}L',\ \ L^*,\ \ L^R,\ \ f^{-1}(L)\ \ \in \mathsf{AD},$$

where f is an homomorphism preserving the length.

Proof. We apply according to each case Theorem 8.20, Theorem 8.23, Corollary 8.25, Theorem 8.26, Theorem 8.27. We notice that if one of the languages L is in a class X/Y which does not satisfy some of the conditions on X and Y of the previous theorems, then these types can be enriched as needed; for example if $L \in X/Y$ then $L \in XZ/YT$ for some adequate Z, T, and the theorem can be applied. □

9

Lowest Positions and Properties of the Bi-Hierarchies (Cont.)

This chapter continues examining the structure of the bi-hierarchies. We import some results from the general bi-hierarchy $\mathcal{BH}(\mathbf{k}, \mathbb{M})$ to the homogeneous bi-hierarchy $\mathcal{BH}(\mathbf{k}, \mathbb{H})$. This more restricted bi-hierarchy permits us to characterize context-free languages as the class $\mathbf{k}^2\mathbb{H}/\mathbb{H}\mathbf{k}$. We also see that Angluin languages, which can exhibit cross-serial dependencies, are in the symmetric class $\mathbb{H}\mathbf{k}^2/\mathbb{H}\mathbf{k}$.

Semi-linearity is a property on the growth of languages which is supposed to hold for natural languages. Restrictiveness of the homogeneous bi-hierarchy allows identification of several fully semi-linear classes. We also classify some non-semi-linear languages inside and outside of the homogeneous bi-hierarchy.

We consider the operator *anti-* to form classes of languages, already introduced in the previous chapter, which we apply here to the context-free languages. We show basic properties of this new class of formal languages which we call *anti-context-free languages*, such as cross-serial dependency and semi-linearity.

In order to summarize the main results we depict the lowest positions of the homogeneous bi-hierarchy. By contrast to the classical hierarchies which are basically linear, this bi-hierarchy suggests a branching disposition of languages.

9.1 The Homogeneous Bi-Hierarchy $\mathcal{BH}(\mathbf{k}, \mathbb{H})$

9.1.1 Results Translatable from the General Bi-Hierarchy, and Some Differences

The first important result of the homogeneous bi-hierarchy is that any language L in any class $L \in X/Y \in \mathcal{BH}(\mathbf{k}, \mathbb{H})$ is decidable. This is consequence of Theorem 7.8 and Theorem C.12.

Some results from the general bi-hierarchy can be translated directly to the homogeneous bi-hierarchy. This is so because the monoids of the patterns of those proofs were homogeneous. So those proofs can be reused for analogous results in $\mathcal{BH}(\mathbf{k}, \mathbb{H})$. More in particular, for any $p \geq 0$ and for any types X, Y:

- ${}^{X}\!/_{\mathbf{1}} = {}^{\mathbf{k}^p}\!/_{Y} = {}^{X}\!/_{\mathbb{H}} = \mathsf{AB}$;
- ${}^{\mathbf{k}^2\mathbb{H}}\!/_{\mathbf{k}} = \mathsf{FN}$;
- $\mathsf{FN}_0 \subseteq {}^{\mathbf{k}\mathbb{H}}\!/_{\mathbf{k}}$;

In addition we have that $\mathbb{H}/X = \mathsf{AB}$ for any type X: we can use the characterization of manifolds in $\mathbf{Man}(\mathbb{H})$ as local sets, and then we see that if $\zeta \neq \emptyset$ then such manifolds do not have effectively bounded ellipticity, whereby the unique language definable is an alphabet language.

An important difference between the homogeneous bi-hierarchy with respect to the general bi-hierarchy is that types $\mathbb{H}^p$ with $p > 1$ do not contribute at all:

Proposition 9.1. *For any types X, Y, Z, T and $p, q \geq 1$:*

$${}^{X\mathbb{H}^pY}\!/_{Z\mathbb{H}^qT} = {}^{X\mathbb{H}Y}\!/_{Z\mathbb{H}T}$$

Proof. The inclusion ($\supseteq$) is trivial. For the other inclusion we recall that a pattern has type $\mathbb{H}$ iff it is of the form $(\zeta)^*_a$. However notice that this pattern is idempotent: $(\zeta)^*_a \cdot (\zeta)^*_a = (\zeta)^*_a$. So, in general, if a pattern has type $X\mathbb{H}^pY$ with $p \geq 1$ then it also has type $X\mathbb{H}Y$. □

This means that classes of finite languages are reduced to the cases:

$$\mathbf{1}/\mathbf{1} = \mathsf{AB},\ \mathbf{k}^2\mathbb{H}/\mathbf{k} = \mathsf{FN},\ {}^{\mathbf{k}\mathbb{H}}\!/_{\mathbf{k}^q}, {}^{\mathbb{H}\mathbf{k}\mathbb{H}}\!/_{\mathbf{k}^q} \subseteq \mathsf{FN}.$$

We can take further advantage of the restrictiviness of the basic type $\mathbb{H}$.

Proposition 9.2. *For any type X, ${}^{X}\!/_{\mathbb{H}\mathbf{k}\mathbb{H}} \subseteq {}^{X}\!/_{\mathbf{k}\mathbb{H}\mathbf{k}}$.*

Proof. Consider a pattern in an arrangement in a linearization in $\mathbf{Lin}(\mathbb{H}\mathbf{k}\mathbb{H})$ of the form $\zeta^*\alpha\zeta^* \in \mathbb{H}\mathbf{k}\mathbb{H}$. This arrangement is never proper because it is ambiguous. So S_Γ is not defined and the arrangement cannot be applied in any case. So the patterns must be of the form α, $\zeta^*\alpha$, or $\alpha\zeta^*$ which are all in $\mathbf{k}\mathbb{H}\mathbf{k}$. □

More in general:

Theorem 9.3. *for any $X/Y \in \mathcal{BH}(\mathbf{k}, \mathbb{H})$ there are $p, q \geq 0$ such that ${}^{X}\!/_{Y} \subseteq {}^{X}\!/_{\mathbf{k}^p\mathbb{H}\mathbf{k}^q}$.*

Proof. By generalizing the previous arguments. A pattern of the form $\varphi_1\zeta^*\varphi_2 \cdots \zeta^*\varphi_n$ with at least two ζ^*'s is never proper. The unique possibilities are $\varphi, \zeta^*\psi, \varphi\zeta^*, \varphi\zeta^*\psi \in \mathbf{k}^p\mathbb{H}\mathbf{k}^q$, for some $p, q \geq 0$. □

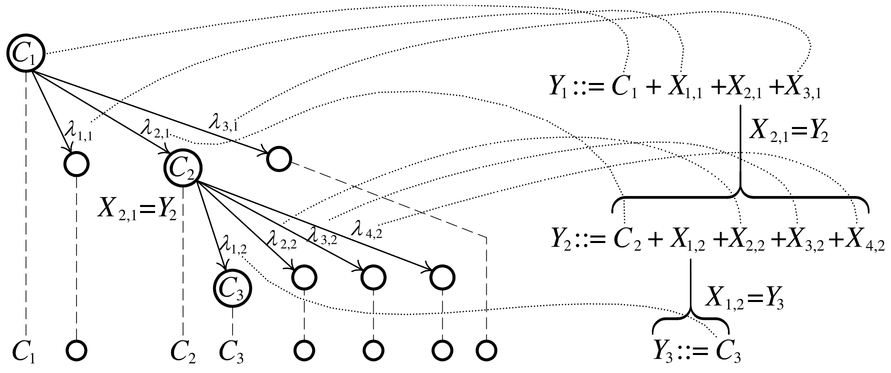


Figure 9.1: An example of simulation of a context-free grammar through syntagmata.

9.1.2 Context-Free Languages

Context-free languages lie in a specific place in the bi-hierarchy: $\mathsf{CF} = \mathbf{k}^2\mathbb{H}/\mathbb{H}\mathbf{k}$. We see first the inclusion ($\subseteq$). The proof rests in simulating derivations of Greibach normal form of a context-free grammar through syntagmata. The Fig. 9.1 shows an example.

Theorem 9.4. $\mathsf{CF} \subseteq {}^{\mathbf{k}^2\mathbb{H}}\!/\!_{\mathbb{H}\mathbf{k}}$.

Proof. Let G be a context-free grammar generating the non-empty language. For each such context-free grammar we can take an equivalent grammar (i.e. generating the same language) in Greibach normal form, see (Hopcroft and Ullman, 1979) and (Blum and Koch, 1999). In these grammars each rewriting rule begins with a terminal symbol which is followed by some variables. More precisely all the rules are of the form $Y ::= C + X_1 + \cdots + X_n$ where $C \in \Sigma$ is a terminal symbol and $Y, X_1, \ldots, X_n$ are variables. If we want the language have the empty string, 0, we have to allow the rule *Start* $::= 0$ where *Start* is the start symbol. Here we ignore this, however the proof can be modified easily to incorporate this rule.

We fix a finite set of letters or terminal symbols and a finite set of variable symbols. Consider the Greinbach normal form context-free grammar G with n rewriting rules:

$$\begin{cases} Y_1 ::= C_1 + X_{1,1} + \cdots + X_{m_1,1}, \\ \;\vdots \qquad \vdots \\ Y_n ::= C_n + X_{1,n} + \cdots + X_{m_n,n}; \end{cases}$$

where Y_i, $X_{i,j}$ are variables and $C_j \in \Sigma$ are terminal symbols for all subscripts i, j. We suppose a start symbol *Start*, so some of the Y_i can be equal to *Start*. We denote L the language generated by this grammar.

Our goal is to build an algebraic dependency grammar (W, Π) such that $L = \mathscr{L}(W, \Pi)$. In order to construct these we take as vocabulary $\Sigma = \{C_1, \ldots, C_n\}$ (the same set of terminals symbols), and as the set of syntactic functions we take $\zeta = \{\lambda_{i,j} \,|\, i = 1, \ldots m_j,\ j = 1, \ldots, n\}$. Notice that we can have more syntactic functions than symbols in the rewriting rules, because symbols can be repeated but not syntactic functions. This will make our construction easier.

The spirit of the proof consists in simulating each rewriting rule by a depth 1 triangle of a syntagma, so that the whole syntagma simulates a derivation in the CFG; see the example in Fig. 9.1. Thus since the derivational history of a string in the CFG can be decomposed by the rewriting rules, the related syntagma can be described by concatenation of triangles.

We need also an auxiliary function:

$$\delta : \{(i, j) \,|\, i = 1, \ldots, m_j, j = 1, \ldots, n\} \cup \{(0, 0)\} \longrightarrow \wp(\{1, \ldots, n\}),$$

which works similarly to the mapping ω of a system of arrangements, although we are going to perform the concatenation of triangles rather than linearizations. This mapping, δ, is defined by the variables in the context-free grammar G as:

$$\delta(0, 0) = \{k \in \{1, \ldots, n\} \,|\, Y_k = Start\},$$
$$\delta(i, j) = \{k \in \{1, \ldots, n\} \,|\, Y_k = X_{i,j}\},$$

for all the subscripts i, j. In the following, when the counters i, j appear they are to be read as $i = 1, \ldots, m_j, j = 1, \ldots, n$.

We establish three manifolds:

- *Non-Ellipticity Manifold.* This is the manifold already used before in order to preclude elliptic syntagmata: **Nell** .

- *Starting Derivation Manifold.* We have to define a valuation to describe the top of each syntagma, which we associate to the start of the derivation in the context-free grammar. Suppose that Y_k is the start symbol $Y_k = Start$. This manifold states the conditions in order to begin with the variable Y_k:

  $$\mathbf{Synt}\begin{pmatrix} \approx C_k \\ 1 \end{pmatrix} \cap \bigcap_{i,j,\ j \neq k} \mathbf{Synt}\begin{pmatrix} \approx 0 \\ \lambda_{i,j} \end{pmatrix}.$$

  The first operand says that the root must be the letter C_k while the second says that only the variables $\lambda_{i,k}$ can be non-null and continue the derivation.

  However several variables Y_k can coincide with the start symbol *Start*, and then we have to consider any of those possibilities, whereby we define:

  $$W_{\text{start}} = \bigcup_{k \in \delta(0,0)} \left(\mathbf{Synt}\begin{pmatrix} \approx C_k \\ 1 \end{pmatrix} \cap \bigcap_{i,j,\ j \neq k} \mathbf{Synt}\begin{pmatrix} \approx 0 \\ \lambda_{i,j} \end{pmatrix} \right).$$

- *Concatenating Triangles Manifold.* Suppose that the variable $X_{i,j}$ is equal to the variable Y_k in the CFG. Then two things occur. First, the function $\lambda_{i,j}$ takes the letter C_k. Second the function $\lambda_{i,j}$ can be followed by the functions $\lambda_{1,k}, \ldots, \lambda_{m_k,k}$ (and they can be null or non-null). However, and most importantly, $\lambda_{i,j}$ cannot be followed by any other function but those. That is $\lambda_{i',j'} \cdot \lambda_{i,j}$ must be null for all subscripts i', j' with $j' \neq k$. In addition this condition must be considered throughout the whole syntagma. The following manifold captures exactly this:

$$\mathbf{Synt}\begin{pmatrix} \approx C_k \\ \lambda_{i,j} \cdot \zeta^* \end{pmatrix} \cap \bigcap_{i',j',\ j' \neq k} \mathbf{Synt}\begin{pmatrix} \approx 0 \\ \lambda_{i',j'} \cdot \lambda_{i,j} \cdot \zeta^* \end{pmatrix}.$$

However as in the above point, we have to consider all the possible Y_k such that $X_{i,j} = Y_k$. So we define:

$$W_{\text{conc}} = \bigcup_{k \in \delta(i,j)} \left(\mathbf{Synt}\begin{pmatrix} \approx C_k \\ \lambda_{i,j} \cdot \zeta^* \end{pmatrix} \cap \bigcap_{i',j',\ j' \neq k} \mathbf{Synt}\begin{pmatrix} \approx 0 \\ \lambda_{i',j'} \cdot \lambda_{i,j} \cdot \zeta^* \end{pmatrix} \right).$$

Notice that for terminal symbols we have $\delta(i, j) = \emptyset$ and then the intersection operator $\bigcap_{i',j',\ j' \neq k}$ must be read $\bigcap_{i',j'}$ since there is no k to consider.

Now we intersect the three manifolds: $W = \mathbf{Nell} \cap W_{\text{start}} \cap W_{\text{conc}}$. We can calculate: $\mathbf{Nell} \in \mathbf{Man}(\mathbf{k}\mathbb{H})$, $W_{\text{start}} \in \mathbf{Man}(\mathbf{k})$ and $W_{\text{conc}} \in \mathbf{Man}(\mathbf{k}^2\mathbb{H})$. So W has type $\mathbf{k}^2\mathbb{H}$.

There remains the linearization. One could think that we need to use the function δ in order to define a system of arrangements. Nevertheless since we have used many more syntactic functions than variables, we have to linearize directly. So we take a unique arrangement which yields a linearization $\Pi \in \mathbf{Lin}(\mathbb{H}\mathbf{k})$:

$$A = 1 + \sum_{j=1}^{n} \sum_{i=1}^{m_j} \zeta^* \lambda_{i,j}.$$

The second sum gives the order of the subsyntagmata according to each rewriting rule, while the first just groups all the possibilities. Let us see this in a little more detail. Let S be a syntagma which simulates a derivation such that Y_k =*Start*. Then the only possible non-null loci in the top of the tree are the root and its children $\lambda_{1,k}, \ldots, \lambda_{m_k,k}$. So when we apply A to S it yields:

$$A(S) = S_1 + 0^\bullet + \cdots + 0^\bullet + \sum_{i=1}^{m_k} S_{\zeta^* \lambda_{i,k}} + 0^\bullet + \cdots + 0^\bullet = S_1 + \sum_{i=1}^{m_k} S_{\zeta^* \lambda_{i,k}},$$

which is exactly the order related to the k-rewriting rule of the context-free grammar. This situation repeats itself for every subtree, so when we continue the linearization, the arrangement only produces the order of the corresponding arrangement. So the arrangement A suffices to linearize any syntagma in the manifold.

To summarize, for each string $x \in L$ there is a derivation in the context-free grammar G that leads to x. This derivation can be simulated by a syntagma in W which can be linearized projectively into x by Π. This yields the inclusion $L \subseteq \mathscr{L}(W,\Pi)$. Conversely, every string $x \in \mathscr{L}(W,\Pi)$ is linearized from at least one syntagma in W, which can be decomposed in triangles and which represents the derivation of x in the context-free grammar G, and this means that $\mathscr{L}(W,\Pi) \subseteq L$. Thus $\Pi(W) = L$.

For every context-free language we can construct a manifold in $\mathbf{Man}(\mathbf{k}^2\mathbb{H})$ and a linearization in $\mathbf{Lin}(\mathbb{H}\mathbf{k})$ proportioning the same language, thus we have that $\mathsf{CF} \subseteq \mathbf{k}^2\mathbb{H}/\mathbb{H}\mathbf{k}$. □

We see the other direction and we prove at the same time that for $p \geq 2$ the constants in the class $\mathbf{k}^p\mathbb{H}/\mathbb{H}\mathbf{k}$ do not confer more expressibility.

Theorem 9.5. ${}^{\mathbf{k}^p\mathbb{H}}\!/_{\mathbb{H}\mathbf{k}} \subseteq \mathsf{CF}$ *for all* $p \geq 0$.

Proof. Let (W,Π) be an algebraic dependency grammar with $W \in \mathbf{Man}(\mathbf{k}^p\mathbb{H})$ and $\Pi \in \mathbf{Lin}(\mathbb{H}\mathbf{k})$. Recall that we characterized manifolds of type $\mathbf{k}^p\mathbb{H}$, Theorem 4.10. These manifolds are $\cup\cap$-combinations of simple manifolds, and these are either strictly strongly p-local sets of syntagmata and p-top languages. We can consider without lost of generality that W is given in disjunctive normal form:

$$W = \bigcup_j \bigcap_i W_{i,j},$$

where $W_{i,j}$ are simple manifolds and i, j range over finite sets of counters. By reordering the $\cap$-terms we can rewrite:

$$W = \bigcup_j \left(\Big(\bigcap_i \mathrm{Top}_p[\mathcal{U}_{i,j}] \Big) \cap \Big(\bigcap_{i'} \mathrm{Tri}_p[\mathcal{V}_{i',j}] \Big) \right),$$

for some sample sets $\mathcal{U}_{i,j}$, $\mathcal{V}_{i,j}$. We saw in the proof of Theorem 4.8 that p-top sets are $\cup\cap$-closed, and then:

$$\mathrm{Top}_p[\mathcal{U}] \cap \mathrm{Top}_p[\mathcal{U}'] = \mathrm{Top}_p[\mathcal{U} \cap \mathcal{U}'],$$
$$\mathrm{Top}_p[\mathcal{U}] \cup \mathrm{Top}_p[\mathcal{U}'] = \mathrm{Top}_p[\mathcal{U} \cup \mathcal{U}'].$$

Regarding the strictly strongly p-local sets, they are closed under intersections:

$$\mathrm{Tri}_p[\mathcal{V}] \cap \mathrm{Tri}_p[\mathcal{V}'] = \mathrm{Tri}_p[\mathcal{V} \cap \mathcal{V}'],$$

but not under unions. Then we can rewrite W as:

$$W = \bigcup_j \left(\mathrm{Top}_p[\bigcap_i \mathcal{U}_{i,j}] \cap \mathrm{Top}_p[\bigcap_{i'} \mathcal{V}_{i',j}] \right) = \bigcup_j \left(\mathrm{Top}_p[\mathcal{U}'_j] \cap \mathrm{Top}_p[\mathcal{V}'_j] \right),$$

where $\mathcal{U}'_j = \bigcap_i \mathcal{U}_{i,j}$ and $\mathcal{V}'_j = \bigcap_{i'} \mathcal{V}_{i',j}$. Now we observe that given two algebraic grammars with the same linearization (V, Π), (V', Π) we have: $\mathscr{L}(V \cup V', \Pi) = \mathscr{L}(V, \Pi) \cup \mathscr{L}(V', \Pi)$. Then:

$$\mathscr{L}(W, \Pi) = \mathscr{L}\Big(\bigcup_j (\mathrm{Top}[\mathcal{U}'_j] \cap \mathrm{Top}_p[\mathcal{V}'_j]), \Pi\Big) = \bigcup_j \mathscr{L}\Big(\mathrm{Top}_p[\mathcal{U}'_j] \cap \mathrm{Top}_p[\mathcal{V}'_j], \Pi\Big).$$

Hence, if we prove that $\mathscr{L}\Big(\mathrm{Top}_p[\mathcal{U}'_j] \cap \mathrm{Top}_p[\mathcal{V}'_j], \Pi\Big) \in \mathsf{CF}$ with $\Pi \in \mathbf{Lin}(\mathbb{H}\mathbf{k})$ we will be done, since context-free languages are closed under unions.

Thus we are going to prove that languages of the form $\mathscr{L}(\mathrm{Top}_p[\mathcal{U}] \cap \mathrm{Top}_p[\mathcal{V}], \Pi)$ are context-free. First we will prove the case $p = 1$, second the case $p = 2$, and then we will generalize for any $p > 2$. Consider then the language $\mathscr{L}(W, \Pi)$, with $W = \mathrm{Top}_1[\mathcal{U}] \cap \mathrm{Top}_1[\mathcal{V}]$ and $\Pi \in \mathbf{Lin}(\mathbb{H}\mathbf{k})$ given by the system of arrangements $\mathcal{A}$. We suppose that the language is not empty (otherwise we are done); in consequence W is not empty.

We construct a context-free grammar with Σ_+ as the set of terminal symbols and the set of variable symbols of the form $X_{T,A}$, i.e. we parametrize the variables ranging $T \in \mathcal{U}$ and $A \in \mathcal{A}$. *Start* stands for the start symbol.

Recall from previous chapters that every linearization in $\mathbf{Lin}(\mathbb{H}\mathbf{k})$ admits a projective normal form where each arrangement consists in a certain permutation of the arrangement:

$$1 + \sum_{\alpha \in \zeta} \zeta^* \alpha.$$

We observe also that when we apply an arrangement, every pattern $\zeta^*\alpha$ in it invokes a new arrangement. Given an arrangement A, A_α stands for an arrangement invoked by the pattern $\zeta^*\alpha$. We need the following relation: we say that the triangle $T' \in \mathcal{V}$ is 1-*concatenable at the locus α with the triangle $T \in \mathcal{V}$* iff $T(\alpha) = T'(1)$.

Now consider the context-free grammar with the following rewriting rules:

- A first set of rewriting rules of the sort:

  $$\mathit{Start} ::= X_{T,A},$$

  for every $T \in \mathcal{U}$ and for every initial arrangement A.

- A second set of rewriting rules consisting in:

  $$X_{T,A} ::= \rho_A \left(T(1) + \sum_{\alpha \in \zeta} X_{T_\alpha, A_\alpha} \right),$$

  for every arrangement $A \in \mathcal{A}$, for every triangle $T \in \mathcal{V}$, and for every possible sum $\sum_{\alpha \in \zeta} X_{T_\alpha, A_\alpha}$, where $T_\alpha \in \mathcal{V}$ is a triangle such that it is 1-concatenable with

T at the locus α and A_α is an arrangement invoked by $\zeta^*\alpha$. Finally, ρ_A is the permutation of the arrangement A, i.e.:[1]

$$A = \rho_A \left(1 + \sum_{\alpha\in\zeta} \zeta^*\alpha\right).$$

- A set of empty rules:

$$X_{0^\bullet,A} ::= 0,$$

for every arrangement A.

Notice that the CFG makes sense. On the one hand, if $W \neq \emptyset$ then $\mathcal{U} \subseteq \mathcal{V}$. So the second set of rules connects to the starting rules. On the other hand, since $0^\bullet \in \mathcal{V}$, which is always true when $W \neq \emptyset$, the right part of second set of rules must contain variables of the form $X_{0^\bullet,A}$. Then these rules connect to the third set of rewriting rules permitting that the computation ends.

This context-free grammar generates the language $\mathscr{L}(W, \Pi)$. The key to understand how it works consists in the fact that syntagmata in W can be decomposed into overlapping triangles of depth 1, i.e. a node and its immediate descendants. Since the linearization is in normal form, each step of a derivation is described by a node (1 in the arrangement) and its immediate descendants (the $\alpha's$ of the patterns $\zeta^*\alpha$ in the arrangement). The above context-free grammar synchronizes the concatenation of triangles and the arrangements.

Now we consider the case $p = 2$, $\mathscr{L}(W, \Pi)$, with $W = \mathrm{Top}_2[\mathcal{U}] \cap \mathrm{Top}_2[\mathcal{V}]$ and $\Pi \in \mathbf{Lin}(\mathbb{H}\mathbf{k})$. We have to visualize each syntagma decomposed in overlapping triangles of depth 2. However the linearization, which we suppose in projective normal form, is in $\mathbf{Lin}(\mathbb{H}\mathbf{k})$ which works with subtrees at depth 1. We can solve this by composing arrangements. Consider the arrangements $A_0, A_1, \ldots, A_m$, such that each A_i, $1 \leq i \leq m$ is invoked by some pattern of A_0. Then we can apply the arrangements in each invoked pattern. The result is a new arrangement of type $\mathbb{H}\mathbf{k}^2$ consisting in a permutation of:

$$1 + \sum_{\alpha\in\zeta} \alpha + \sum_{\alpha\in\zeta}\sum_{\beta\in\zeta} \zeta^*\alpha\beta.$$

We call these 2-composed arrangements. As before, given a 2-composed arrangement A and the locus $\alpha\beta \in \zeta \cdot \zeta$ we can refer a new arrangement invoked by the pattern $\zeta^*\alpha\beta$ as $A_{\alpha\beta}$. We generalize the concatenation of triangles. We say that the triangle $T'' \in \mathcal{V}$ is 2-*concatenable at the locus* $\alpha\beta$ *with the triangle* $T \in \mathcal{V}$ iff there is another triangle $T' \in \mathcal{V}$ such that $T''(1) = T'(\beta)$ and $T'(1) = T(\alpha)$. Then we define the context-free grammar:

[1] More formally, given a permutation (a bijective mapping) $\rho : \{1, \ldots, n\} \longrightarrow \{1, \ldots, n\}$ we define the *permutation* $\rho(x)$ *of a string* $x = x_1 + \cdots + x_n$, where $x_1, \ldots, x_n$ are prime factors, as $\rho(x) = x_{\rho(1)} + \cdots + x_{\rho(n)}$. Then $\rho(x)$ does not depend on the prime factors of x but in the place of these factors. For example, given the permutation $\rho = \binom{1\,2\,3}{3\,1\,2}$, we have $\rho(a+b+c) = b+c+a$ and $\rho(p+q+r) = q+r+p$.

- A first set of rules of the sort:

  $$\mathit{Start} ::= X_{T,A},$$

  for every $T \in \mathcal{U}$ and for every initial 2-composed arrangement A.

- A second set consisting in:

  $$X_{T,A} ::= \rho_A \left(T(1) + \sum_{\alpha\in\zeta} T(\alpha) + \sum_{\alpha\in\zeta} \sum_{\beta\in\zeta} X_{T_{\alpha\beta},A_{\alpha\beta}} \right),$$

  for every 2-composed arrangement A, for every triangle $T \in \mathcal{V}$, and for every possible double sum $\sum_{\alpha\in\zeta} \sum_{\beta\in\zeta} X_{T_{\alpha\beta},A_{\alpha\beta}}$ where $T_{\alpha\beta} \in \mathcal{V}$ is a triangle such that it is 2-concatenable with T at the locus $\alpha\beta$ and $A_{\alpha\beta}$ is an arrangement invoked by $\zeta^*\alpha\beta$. Finally, ρ_A is the permutation of the 2-composed arrangement A, i.e.:

  $$A = \rho_A \left(1 + \sum_{\alpha\in\zeta} \alpha + \sum_{\alpha\in\zeta} \sum_{\beta\in\zeta} \zeta^*\alpha\beta \right).$$

- A set of empty rules:

  $$X_{0^\bullet,A} ::= 0,$$

  for every 2-composed arrangement A.

This grammar generates the language $\mathscr{L}(W,\Pi)$. The case $p > 2$, although the notation turns out more complicated, consists in considering triangles of depth p, p-concatenation of triangles and p-composed arrangements. □

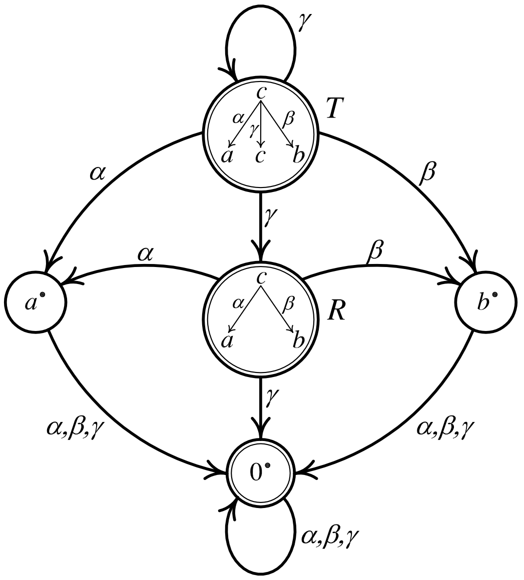
Corollary 9.6. *We have that:*

(i) $\mathsf{CF} = {}^{\mathbf{k}^2\mathbb{H}}\!/\!_{\mathbb{H}\mathbf{k}} = {}^{\mathbf{k}^p\mathbb{H}}\!/\!_{\mathbb{H}\mathbf{k}}$ *for all* $p \geq 2$.

(ii) ${}^{\mathbf{k}\mathbb{H}}\!/\!_{\mathbb{H}\mathbf{k}} \subsetneq {}^{\mathbf{k}^2\mathbb{H}}\!/\!_{\mathbb{H}\mathbf{k}}$.

Proof. (i) According the two preceding theorems: $\mathbf{k}^p\mathbb{H}/\mathbb{H}\mathbf{k} \subseteq \mathsf{CF} \subseteq \mathbf{k}^2\mathbb{H}/\mathbb{H}\mathbf{k} \subseteq \mathbf{k}^p\mathbb{H}/\mathbb{H}\mathbf{k}$, for all $p \geq 2$. For (ii) we have that $\{a^2\} \in \mathbf{k}^2\mathbb{H}/\mathbb{H}\mathbf{k} = \mathsf{CF}$ (trivially) but $\{a^2\} \notin \mathbf{k}\mathbb{H}/\mathbb{H}\mathbf{k}$. This last is obtained by considering all the cases in the set of inner triangles and the top triangles which defines the manifolds with type $\mathbf{k}^p\mathbb{H}$ with $p = 0, 1$ (Theorem 4.10). □

Example 30. An illustration of Theorem 9.5. Consider the language $L_{\mathrm{mult}} = \mathscr{L}(W_{\mathrm{mult}}, \Pi_{\mathrm{mult}}) \in \mathbf{k}\mathbb{H}/\mathbb{H}\mathbf{k}$. We saw in Example 10 that $W_{\mathrm{mult}} = \mathrm{Top}_1[\mathcal{U}] \cap \mathrm{Tri}_1[\mathcal{V}]$ where $\mathcal{U} = \{0^\bullet, T, R\}$, $\mathcal{V} = \{0^\bullet, a^\bullet, b^\bullet, T, R\}$ and where T, R are the syntagmata in the figure below which captures also the relation of 1-concatenability. Vertices in that graph are given by $\mathcal{V}$, vertices in $\mathcal{U}$ are doubly circled, and an edge $S \xrightarrow{\varphi} S'$ means that a triangle S is 1-concatenable to the S' at the locus φ.



Recall that Π_{mult} is given by the unique arrangement $A = \zeta^*\alpha + \zeta^*\beta + 1 + \zeta^*\gamma$. Then we construct the corresponding context-free grammar:

$$\begin{cases} \mathit{Start} ::= X_{T,A}, \\ \mathit{Start} ::= X_{R,A}, \\ \mathit{Start} ::= 0, \\ X_{T,A} ::= X_{a^\bullet,A} + X_{b^\bullet,A} + c + X_{T,A}, \\ X_{T,A} ::= X_{a^\bullet,A} + X_{b^\bullet,A} + c + X_{R,A}, \\ X_{R,A} ::= X_{a^\bullet,A} + X_{b^\bullet,A} + c + X_{0^\bullet,A}, \\ X_{a^\bullet,A} ::= X_{0^\bullet,A} + X_{0^\bullet,A} + a + X_{0^\bullet,A}, \\ X_{b^\bullet,A} ::= X_{0^\bullet,A} + X_{0^\bullet,A} + b + X_{0^\bullet,A}, \\ X_{0^\bullet,A} ::= 0; \end{cases} \cong \begin{cases} \mathit{Start} ::= X, Y, 0, \\ X ::= a + b + c + X, \\ X ::= a + b + c + Y, \\ Y ::= a + b + c + Y. \end{cases}$$

And these context-free grammars generate the language $(abc)^* = L_{\mathrm{mult}}$.

9.1.3 Angluin Languages

Angluin Pattern Grammars are a very intuitive kind of grammars, probably the most simple way to show a non-context-free language, which were introduced by Angluin (1980*a*,*b*) in the context of machine learning.

Let us recall a comment on the terminology. Angluin denominated *pattern languages* the languages that we are going to present, but to avoid interference of terms we call them *Angluin Languages* and we call *Angluin-pattern* a pattern in the sense of Angluin. This is a coincidence: there is no relevant relation between Angluin patterns and Monoidal patterns.

An *Angluin pattern* is a string in the alphabet $\Sigma_0 \sqcup \Sigma_{var}$. Elements in Σ_0 are called *Angluin constants* and elements in Σ_{var} are called *variables*. A *substitution* is a homomorphism $f : (\Sigma_0 \sqcup \Sigma_{var})^* \longrightarrow \Sigma_0^*$ such that when we restrict it to Σ_0 it is the identity mapping. An Angluin language $L_P \subseteq \Sigma_0^*$ is given by a *Angluin pattern* $P \in (\Sigma_0 \sqcup \Sigma_{var})^*$ as the set:

$$L_P = \{f(P) \in \Sigma_0^* \mid f : (\Sigma_0 \sqcup \Sigma_{var})^* \longrightarrow \Sigma_0^*,\ f \text{ substitution}\}.$$

For example if we consider the Angluin pattern $P = abXXcY \in (\{a, b\}^* \sqcup \Sigma_{var})^*$ this generates the language $L_P = ab \cdot L_{\text{copy}} \cdot c \cdot \{a, b\}^*$. We notate AN the class of Angluin Languages. This class is not comparable with context-free languages; however:

$$\mathsf{AB} \subset \mathsf{AN} \subset \mathsf{IN},$$

where IN is the class of indexed languages.[2]

We say that an Angluin language is *free of Angluin constants* iff the Angluin pattern does not have any constant. We notate AN_0 the class of these languages. Clearly Angluin languages free of Angluin constants is a proper subclass of Angluin languages: $\mathsf{AN}_0 \subset \mathsf{AN}$. For example the copy language is free of Angluin constants, $L_{\text{copy}} \in \mathsf{AN}_0$.

Theorem 9.7. $\mathsf{AN} \subset {}^{\mathbb{H}\mathbf{k}^2}\!/\!_{\mathbb{H}\mathbf{k}}$.

Proof. First we see the inclusion. Consider an Angluin Pattern $P \in (\Sigma_0 \sqcup \Sigma_{var})^*$. Factorize P in prime factors in $\Sigma_0 \sqcup \Sigma_{var}$, $P = p_1 \cdots p_m$. We take $\zeta = \{\alpha_1, \ldots, \alpha_m\}$ and we decompose ζ as $\zeta_0 \sqcup \zeta_{var}$, such that $\zeta_0 = \{\alpha_i \mid p_i \in \Sigma_0\}$ and $\zeta_{var} = \{\alpha_i \mid p_i \in \Sigma_{var}\}$. For example the pattern $abXXcY$ gives the decomposition $\zeta_0 = \{\alpha_1, \alpha_2, \alpha_5\}$, $\zeta_{var} = \{\alpha_3, \alpha_4, \alpha_6\}$.

Let W_1 be the manifold:

$$W_1 = \bigcap_{\alpha\in\zeta} \bigcap_{\beta\in\zeta_0} \mathbf{Synt}\begin{pmatrix} \approx 0 \\ \zeta^*\alpha\beta \end{pmatrix} \in \mathbf{Man}(\mathbb{H}\mathbf{k}^2),$$

[2] See Example 35 later for some comments about indexed languages.

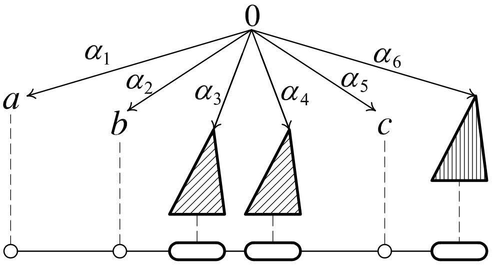
which prunes below the functions for the Angluin constants in the tree. Let W_2 be the manifold:

$$W_2 = \bigcap_{\alpha_i,\alpha_j \in \zeta_{var},\, p_i = p_j} \mathbf{Synt}\begin{pmatrix} \approx \\ (\zeta)_2^*(\alpha, \beta) \end{pmatrix} \in \mathbf{Man}(\mathbb{H}\mathbf{k}),$$

which makes equal subtrees that are governed by functions which represent identical variables. Finally we define:

$$W_3 = \bigcap_{\alpha_i \in \zeta_0} \mathbf{Synt}\begin{pmatrix} \approx p_i \\ \alpha_i \end{pmatrix} \in \mathbf{Man}(\mathbf{k}), \quad \text{and} \quad W_4 = \mathbf{Synt}\begin{pmatrix} \approx 0 \\ 1 \end{pmatrix} \in \mathbf{Man}(\mathbf{1}).$$

W_3 fills out the Angluin constants in the tree and W_4 fills out a 0 at the root. Then we consider $W = W_1 \cap W_2 \cap W_3 \cap W_4 \in \mathbf{Man}(\mathbb{H}\mathbf{k}^2)$. The following figure shows an example of syntagma in the manifold W for the Angluin pattern $abXXcY$.



Now we simply take the arrangement $A = 1 + \sum_{i=1}^{m} \zeta^* \alpha_i$ (see again the example in the figure) which yields a linearization in $\mathbf{Lin}(\mathbb{H}\mathbf{k})$. So $\mathsf{AN} \subseteq \mathbb{H}\mathbf{k}^2/\mathbb{H}\mathbf{k}$.

In order to see that the inclusion is strict we take a finite language, for example $\{a, b\} \notin \mathsf{AN}$. Since $\mathsf{FN} \subset \mathbb{H}\mathbf{k}^2/\mathbb{H}\mathbf{k}$, $\mathsf{AN} \subsetneq \mathbb{H}\mathbf{k}^2/\mathbb{H}\mathbf{k}$. □

Angluin Languages free of Angluin constants inhabit a lower position in the bi-hierarchy.

Corollary 9.8. $\mathsf{AN}_0 \subset {}^{\mathbb{H}\mathbf{k}}/_{\mathbb{H}\mathbf{k}}$.

Proof. We reuse the notation from the last proof. Since $\zeta_0 = \emptyset$ we only need to take the manifold $W = W_2 \cap W_3 \cap W_4 \in \mathbf{Man}(\mathbb{H}\mathbf{k})$. For the strictness of the inclusion we take $\{ab, cb\} \in \mathsf{FN}_0 \subseteq \mathbb{H}\mathbf{k}/\mathbb{H}\mathbf{k}$; but $\{ab, cb\} \notin \mathsf{AN}$ and therefore $\{ab, cb\} \notin \mathsf{AN}_0$. □

9.1.4 The q-copy and q-respectively Languages

We have seen that some non-context-free languages which contain cross-serial dependencies inhabit the class $\mathbb{H}\mathbf{k}^2/\mathbb{H}\mathbf{k}$. We explore a little more the weak capacity of this class, subsuming some generalizations. We note that the q-copy language, $L_{q\text{-copy}} = \{x^q \mid x \in \Sigma^*\}$, is an Angluin language for any $q \geq 0$, so $L_{q\text{-copy}} \in \mathbb{H}\mathbf{k}^2/\mathbb{H}\mathbf{k}$.

We see another possible grammar for the q-copy language the base trees of which serve to construct grammars for a couple more examples:[3]

Example 31. Generalized copy, generalized counting and crossed languages. Consider the languages: $L_{q\text{-copy}} = \{x^q \mid x \in \Sigma^*\}$, $L_{q\text{-count}} = \{a_1^n a_2^n \cdots a_q^n \mid a_1, \ldots, a_q \in \Sigma, n \in \mathbb{N}_+\}$ and $L_{\text{cros}} = \{a^n b^m c^n d^m \mid n, m \in \mathbb{N}_+\}$, named the *q-copies language*, *q-counting language* and *crossed language*. Even though these languages are more general than the original versions, they inhabit still the same class:

$$L_{q\text{-copy}},\ L_{q\text{-resp}},\ L_{\text{cros}} \in {}^{\mathbb{H}\mathbf{k}^2}\!\!\diagup_{\mathbb{H}\mathbf{k}}.$$

To show this, we are going to suppose $q = 4$, but the general case generalizes easily. We suppose four syntactic functions $\zeta = \{\alpha, \beta, \gamma, \sigma\}$. For the three languages we are going to use a manifold which trims the tree to obtain syntagmata as in Fig. (a), (b) and (c):

$$W_{\text{trim}} = \bigcap_{\varphi \in \{\alpha,\beta,\gamma\}} \bigcap_{\psi \in \{\alpha,\beta,\gamma,\sigma\}} \mathbf{Synt}\begin{pmatrix} \approx 0 \\ \zeta^* \varphi \psi \end{pmatrix}.$$

In the three cases, given that all of them have the same kind of tree shape, we use the same linearization given by the arrangement: $A = \zeta^*\alpha + \zeta^*\beta + \zeta^*\gamma + 1 + \zeta^*\sigma$.

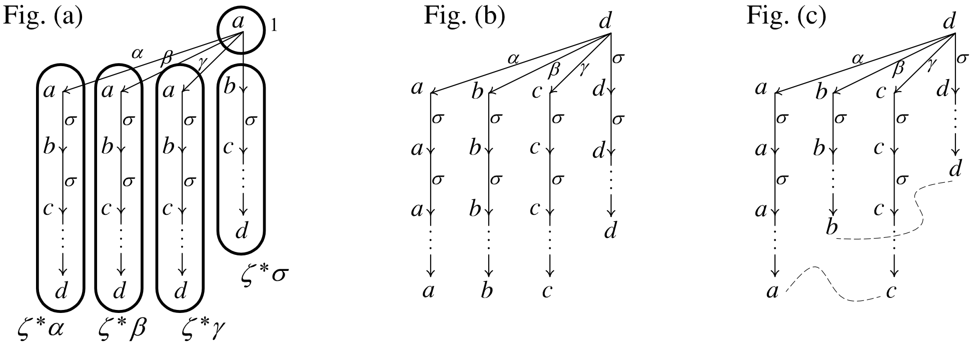


Fig. (a) Fig. (b) Fig. (c)

Let us see the first language:

$$W_{4\text{-copy}} = W_{\text{trim}} \cap \mathbf{Synt}\begin{pmatrix} \approx_4 \\ (\zeta)_2^*(1, \alpha, \beta, \gamma) \end{pmatrix},$$

where $\approx_4 (x, y, z, t) = 1 \iff x = y = z = t$. The first part gives the syntagmata tree-shape as in Fig. (a), while the last manifold forces all the branches to be equal.

To obtain the second language, Fig. (b), we have to change the second manifold:

$$W_{4\text{-count}} = W_{\text{Trim}} \cap \mathbf{Synt}\begin{pmatrix} a_+ & b_+ & c_+ & d_+ \\ \zeta^*\alpha & \zeta^*\beta & \zeta^*\gamma & \zeta^* \end{pmatrix} \cap \mathbf{Synt}\begin{pmatrix} B_4 \\ (\zeta)_4^* \cdot (1, \alpha, \beta, \gamma) \end{pmatrix},$$

[3] The q-counting languages $L_{q\text{-count}} = \{a_1^n \cdots a_q^n \mid n \in \mathbb{N}_+, a_1, \ldots, a_q \in \Sigma\}$ are very similar to the q-respectively languages, $L_{q\text{-resp}} = \{a_1^n \cdots a_q^n \mid n \in \mathbb{N}_+\}$ with $a_1, \ldots, a_q \in \Sigma$ fixed. This last also inhabits $\mathbb{H}\mathbf{k}^2/\mathbb{H}\mathbf{k}$.

where the valuations a_+ means $(a_+)(x) = 1 \iff x \in a_+ = \{0, a\}$, and the valuation B_4 means:

$$B_4(x, y, z, t) = 1 \iff \text{(if one of the } x, y, z, t \text{ is 0 then the others are also 0).}$$

This ensures that all the branches have the same length.

For the crossed language we just change the valuation B_4 to B defined as $B(x, y, z, t) = B_2(x, z) \wedge B_2(y, t)$. Now instead of linking simultaneously the depth of all branches, we link the first to the third one and the second to the fourth one, see Fig. (c).

9.2 Semi-linearity

9.2.1 Semi-linearity in the Bi-hierarchies

We set $\Sigma = \{a_1, \ldots, a_k\}$. The *Parikh mapping* is the mapping $p : \Sigma^* \longrightarrow \mathbb{N}_+^k$ defined by $p(x) = (|x|_{a_1}, \ldots, |x|_{a_k})$. A set in $\mathbb{N}_+^k$ is *semi-linear* iff it is a finite union of sets of the form $\{p_1 A_1 + \cdots + p_n A_n + B \mid p_1, \ldots, p_n \in \mathbb{N}_+\}$ for some $n \geq 0$ and $A_1, \ldots, A_n, B \in \mathbb{N}_+^k$. A language $L \subseteq \Sigma^*$ is *semi-linear* iff $p(L)$ is a semi-linear set.

Theorem 9.9 (Parikh Theorem)**.** *Each context-free language is semi-linear.*

Proof. (Parikh, 1966). □

Semi-linearity is a property of a languages, but in the case of the algebraic dependency languages this property is translated to the manifold. We define the Parikh mapping for syntagmata as $P : \mathbf{Synt}_{\Sigma,\zeta} \longrightarrow \mathbb{N}_+^n$, by $P(S) = (|S^{-1}(a_1)|, \ldots, |S^{-1}(a_n)|)$, where $\Sigma = \{a_1, \ldots, a_n\}$. We say that a manifold is semi-linear if its Parikh image is a semi-linear set.

Suppose an algebraic dependency language $L = \mathscr{L}(W, \Pi)$, and let x be the result of linearizing $S \in W$, i.e. $x \in \Pi(S)$. We have that the Parikh mappings for the string and for the syntagma coincide, $p(x) = P(S)$, and then:

$$L \text{ is semi-linear} \iff W \text{ is semi-linear.}$$

With this in mind the following theorems are easy to prove. We say that a class of languages is semi-linear iff each language in the class is.

Theorem 9.10. *Consider a class X/Y in any bi-hierarchy:*

$$X\!/\!Y \textit{ is semi-linear} \iff -\left(X\!/\!Y\right) \textit{ is semi-linear.}$$

Proof. $\mathscr{L}(W,\Pi) \in -X/Y \iff$ there is a language $\mathscr{L}(W',\Pi) \in X/Y$ with $W' = W^{\Re}$. However it is clear that $P(S) = P(S^{\Re})$, since the symmetric syntagma contains exactly the same letters though in other loci. Then $P(W) = P(W')$, hence the equivalence. □

Turning back to the homogeneous monoid:

Lemma 9.11. *For any type $Y \in \{\mathbf{k}, \mathbb{H}\}^*$ the classes ${}^{\mathbf{k}^p\mathbb{H}}\!/_Y$ and ${}^{\mathbb{H}\mathbf{k}^p}\!/_Y$ are semi-linear for any $p \geq 0$.*

Proof. Instead of proving that manifolds in $\mathbf{Man}(\mathbf{k}^p\mathbb{H})$ are semi-linear we proceed indirectly. We consider the case ${}^{\mathbf{k}^p\mathbb{H}}\!/_Y$, the symmetric case follows by the previous theorem. Languages in $\mathbf{k}^p\mathbb{H}/\mathbf{k}^q$ or $\mathbf{k}^p\mathbb{H}/\mathbb{H}^q$ are finite and thus trivially semi-linear. Now we consider the case $\mathbb{H}\mathbf{k} \sqsubseteq Y$. Suppose $\mathscr{L}(W,\Pi) \in \mathbf{k}^p\mathbb{H}/Y$. Since $\mathbb{H}\mathbf{k} \sqsubseteq Y$, we take any linearization $\Pi' \in \mathbf{Lin}(\mathbb{H}\mathbf{k})$ in projective normal form, which can linearize any manifold (i.e. $\mathrm{dom}(\Pi') = \mathbf{Synt}$ provided that Π' is in projective normal form). Then we consider the language $\mathscr{L}(W,\Pi') \in \mathbf{k}^p\mathbb{H}/\mathbb{H}\mathbf{k} \subseteq \mathsf{CF}$ which is semi-linear by the Parikh Theorem. Thus the manifold W is semi-linear, which implies that $\mathscr{L}(W,\Pi)$ is semi-linear. Now consider the remaining cases: $Y \neq \mathbf{k}^q$, $Y \neq \mathbb{H}^q$, $\mathbb{H}\mathbf{k} \not\sqsubseteq Y$ and we observe that the only possibility is $Y = \mathbf{k}\mathbb{H}$. However $\mathbf{k}^p\mathbb{H}/\mathbf{k}\mathbb{H} = -(\mathbf{k}^p\mathbb{H}/\mathbb{H}\mathbf{k})$ and by the previous theorem since $\mathbf{k}^p\mathbb{H}/\mathbb{H}\mathbf{k}$ is semi-linear, $\mathbf{k}^p\mathbb{H}/\mathbf{k}\mathbb{H}$ is too. □

Theorem 9.12. *Manifolds in $\mathbf{k}^p\mathbb{H}$ and $\mathbb{H}\mathbf{k}^p$ are semi-linear for any $p \geq 0$. So in general for any type Y in any classificatory monoid the classes ${}^{\mathbf{k}^p\mathbb{H}}\!/_Y$ and ${}^{\mathbb{H}\mathbf{k}^p}\!/_Y$ are semi-linear.*

Proof. Trivial, by the last theorem. □

9.2.2 Exponential Growth Languages

This subsection and the following one will consider languages that do not satisfy the constant growth property, nor the semi-linearity constraint. First we examine the exponential growth language which is still in the homogeneous bi-hierarchy.

The language $\{a^{2^n} \mid n \in \mathbb{N}_+\}$ is known to be an *indexed language*, but non-context-free. Indexed languages, IN, were introduced by A. Aho as a generalization of context-free languages, (Aho, 1968).[4] We have seen some examples of indexed languages: $\mathsf{AN} \subset \mathsf{IN}$, or $\mathsf{CF} \subset \mathsf{IN}$.

An indexed grammar (IG) (we adapt notation and follow comments from (Kallmeyer, 2010)), looks like a context-free grammar except that the non-terminals are equipped with stacks of indices, i.e., besides the non-terminals symbols N and the terminals Σ, we have an alphabet I of indices. In a derived sentential form x, non-terminals can be equipped with stacks of indices, i.e., $x \in (N \cdot I^* \cup \Sigma)^*$.

[4] See also (Hayashi, 1973). For a comparison with other formalisms with an equivalent weak capacity see (Vijay-Shanker and Weir, 1994).

The productions in an IG have the form (i) $A \longrightarrow \alpha$ or (ii) $A \longrightarrow Bf$ or (iii) $Af \longrightarrow \alpha$ with $A, B \in N$, $f \in I$, $\alpha \in (N \cup \Sigma)$. The first kind of production works like context-free productions while copying the stack of A to all non-terminals in α. The second kind of production adds a symbol to the stack of A while replacing A with B. The third kind of production deletes a symbol f from the stack of A and then works like the first kind of production.

In order to obtain $\{a^{2^n} \mid n \in \mathbb{N}_+\}$ we consider the indexed grammar with $N = \{S, A, B\}$, $I = \{f, g\}$, $\Sigma = \{a\}$ and the productions: $S \longrightarrow a$, $S \longrightarrow Ag$, $A \longrightarrow Af$, $A \longrightarrow B$, $Bf \longrightarrow BB$, $Bg \longrightarrow aa$. We have the derivation:

$$\begin{aligned}
S &\Rightarrow Ag && (S \rightarrow Ag)\\
&\Rightarrow Afg && (A \rightarrow Af)\\
&\overset{*}{\Rightarrow} Afffg && \\
&\Rightarrow Bfffg && (A \rightarrow B)\\
&\Rightarrow BffgBffg && (Bf \rightarrow BB)\\
&\overset{*}{\Rightarrow} BfgBfgBfgBfg && \\
&\overset{*}{\Rightarrow} BgBgBgBgBgBgBgBg && \\
&\overset{*}{\Rightarrow} aaaaaaaaaaaaaaaa = a^{2^4} && (Bg \rightarrow aa)
\end{aligned}$$

The following example situates in the homogeneous bi-hierarchy the language defined as $L_{\mathrm{expo}} = \{a^{2^n - 1} \mid n \in \mathbb{N}_+\}$. The construction can be modified slightly to obtain the extra a by adding an extra function hanging from the root which just contains this extra a. Thus one obtains the language $\{a^{2^n} \mid n \in \mathbb{N}_+\}$.

Example 32. Exponential growth languages. We have that:

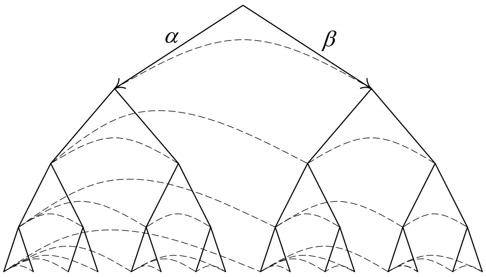
$$L_{\mathrm{expo}} \in {}^{\mathbb{H}\mathbf{k}\mathbb{H}}\!/_{\mathbb{H}\mathbf{k}}.$$

Unary languages have an advantage: since all the letters in the tree are the same we do not have to worry about the linearization; for example we can always use a projective linearization in **Lin**($\mathbb{H}$**k**).

We take $\Sigma = \{a\}$ and $\zeta = \{\alpha, \beta\}$. We are going to construct a manifold containing full binary trees. Since the size of full binary trees of depth d is $2^d - 1$ when we linearize them we obtain the required language. The key consists in forcing all the loci at the same level to be equal. We take the manifold:

$$W_{\mathrm{expo}} = \mathbf{Nell} \cap \mathbf{Synt}\left((\zeta)_2^* \overset{\approx}{(\alpha, \beta)} (\zeta)_2^* \right).$$

This rule can be read as: for any syntagma S and for all $x, y \in \zeta^*$, $S(x\alpha y) = S(x\beta y)$. So for example the sibling loci α and β at the first level take always same value. In general siblings αy and βy take always the same value. Consider at the second level the grandchildren $\alpha\alpha$ $\beta\alpha$ and their cousins $\alpha\beta$ and $\beta\beta$. The locus $\alpha\alpha$ and its cousin $\beta\alpha$ take the same value. Since siblings take the same value, all the grandchildren at the second level take the same value, and so forth for the next generations. In general if a member of a generation is null all its distant relatives at the same level will take the same value, 0 or a. The figure below shows a set of links which suffices to make equal all the loci at a same level.



9.2.3 Polynomial Growth Languages

We are going to see a family of languages which seem to have an infinitely increasing complexity in the general bi-hierarchy. First we see a pair of examples for the quadratic and cubic growth unary languages, and then we generalize for any unary language with polynomial growth.

In the following chapter we examine (supposed) natural constructions which are not semi-linear. Chinese numbers and genitive stack in Old Georgian exhibit quadratic growths. What is significant is that in both cases the classes inhabit the general bi-hierarchy and importantly two $\mathbb{M}$'s in the numerator of the class seem to be needed. The following examples suggest that the degree of the polynomial growth coincides with the numbers of $\mathbb{M}$'s in the numerator of the class.

Example 33. Quadratic growth language. Consider the language $L_{n^2\text{-poly}} = \{a^{n^2} \mid n \in \mathbb{N}_+\}$. We see that

$$L_{n^2\text{-poly}} \in {}^{(\mathbf{k}\mathbb{M})^2}\!/_{\mathbb{M}\mathbf{k}}.$$

Recall that we do not have to worry about linearizations for unary languages; we can always use a projective linearization in $\mathbf{Lin}(\mathbb{M}\mathbf{k})$. So we see the manifold. The strategy consists in constructing a square-shaped tree through binary trees, $\zeta = \{\alpha, \beta\}$. First we construct a framework which simulates the grid of natural number pairs, $\mathbb{N}_+ \times \mathbb{N}_+$. This is made by the manifold:

$$W_{2\text{-grid}} = \mathbf{Synt}\begin{pmatrix} \approx 0 \\ \beta\alpha\zeta^* \end{pmatrix} \cap \mathbf{Nell}.$$

Of course this is not a grid, but it behaves similarly, see Fig. (a):

Fig. (a)

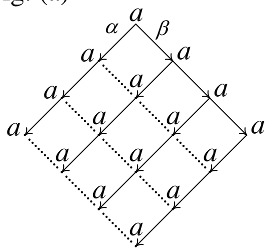


Fig. (b)

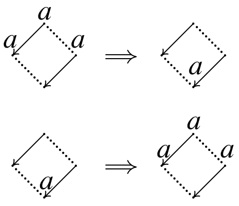
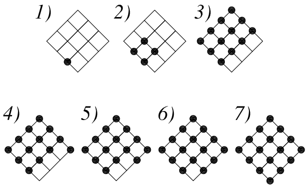


Fig. (c)



Now the key is imposing a constraint on a cell, and then transporting it around the grid:

$$S(1) = a, S(\alpha) = a, S(\beta) = a \iff S(\alpha\beta) = a.$$

See Fig. (b). Since we want the condition for every $\alpha^n\beta^m$, we write:

$$S(\alpha^n\beta^m) = a, S(\alpha^{n+1}\beta^m) = a, S(\alpha^n\beta^{m+1}) = a \iff S(\alpha^{n+1}\beta^{m+1}) = a.$$

This can be captured by the valuation $B(x, y, x, t) = (x \approx a) \wedge (y \approx a) \wedge (z \approx a) \leftrightarrow (t \approx a)$ and the pattern: $\Gamma = (1, \alpha, 1, \alpha)\cdot(\alpha, \alpha, \alpha, \alpha)^*\cdot(1, 1, \beta, \beta)\cdot(\beta, \beta, \beta, \beta)^*$. With this we define:

$$W_{2\text{-cell}} = \mathbf{Synt}\begin{pmatrix} B \\ \Gamma \end{pmatrix}, \qquad W_{2\text{-edges}} = \mathbf{Synt}\begin{pmatrix} \approx \\ (\alpha, \beta)^* \end{pmatrix}.$$

The second manifold ensures that the two sides of the square are equal (notice that the pattern $(\alpha, \beta)^*$ in this manifold is not homogeneous). And now we have the full manifold:

$$W_{2\text{-poly}} = W_{2\text{-grid}} \cap W_{2\text{-cell}} \cap W_{2\text{-edges}}.$$

All this works as follows (see Fig. (c)). Suppose for example that $S(\alpha^3\beta^2) = a$ which is marked with a bullet in the figure (step 1); then by the condition $W_{2\text{-cell}}$, we can mark the three superior loci (step 2), $S(\alpha^3\beta), S(\alpha^2\beta), S(\alpha^2\beta^2) = a$. This can be repeated, up to the root, until to get a rectangle (step 3). Now we apply the condition on sides, $W_{2\text{-edges}}$, and then we perform step 4. Steps 5, 6, and 7 mark from top to bottom using again the cell conditions. So in general given some non-null locus we always complete a square and $W_{2\text{-poly}}$ just contains squares. To sum up, $W_{2\text{-grid}}$ has type $\mathbf{k}^2\mathbb{M}$, $W_{2\text{-cell}}$ has type $(\mathbf{k}\mathbb{M})^2$, $W_{2\text{-edges}}$ has type $\mathbb{M}$. Therefore $L_{n^2\text{-poly}} \in (\mathbf{k}\mathbb{M})^2/\mathbb{M}\mathbf{k}$.

Example 34. Cubic growth language. Consider the language $L_{n^3\text{-poly}} = \{a^{n^3} \mid n \in \mathbb{N}_+\}$. We see that:

$$L_{n^3\text{-poly}} \in {}^{(\mathbf{k}\mathbb{M})^3}\!\big/_{\mathbb{M}\mathbf{k}}.$$

The planes are similar as in the quadratic case. We take $\zeta = \{\alpha, \beta, \gamma\}$ and we define a cubic grid (see Fig. (a)):

$$W_{3\text{-grid}} = \mathbf{Synt}\begin{pmatrix} \approx 0 & \approx 0 & \approx 0 \\ \gamma\beta\zeta^* & \gamma\alpha\zeta^* & \beta\alpha\zeta^* \end{pmatrix} \cap \mathbf{Nell}.$$

These conditions say that γ cannot appear before β; γ cannot appear before α, and β cannot appear before α. If a string does not have the order $\alpha^n\beta^m\gamma^p$ then by one of the conditions in the manifold it is considered null.

Fig. (a)

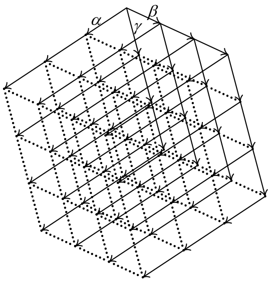
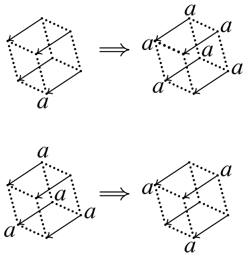


Fig. (b)



Regarding the cell conditions, we have to highlight that now we cannot multiply the left and the right side to transport the conditions. Instead we can use the factorization of Γ in the quadratic case, and generalize it. We need two cell conditions such that given any non-null locus, we will be able to fill out a cube. They are shown in Fig. (b). We see the first cell condition:

$$S(\alpha\beta\gamma) = a \Longrightarrow S(1), S(\alpha), S(\beta), S(\gamma), S(\alpha\beta), S(\beta\gamma), S(\alpha\gamma) = a.$$

Now we need to transport this condition around the grid, so in general the condition is:

$$\begin{aligned} S(\alpha^{n+1}\beta^{m+1}\gamma^{p+1}) = a \Longrightarrow & S(\alpha^n\beta^m\gamma^p) = a,\ S(\alpha^{n+1}\beta^m\gamma^p) = a,\ S(\alpha^n\beta^{m+1}\gamma^p) = a, \\ & S(\alpha^n\beta^m\gamma^{p+1}) = a,\ S(\alpha^{n+1}\beta^{m+1}\gamma^p) = a,\ S(\alpha^n\beta^{m+1}\gamma^{p+1}) = a, \\ & S(\alpha^{n+1}\beta^m\gamma^{p+1}) = a. \end{aligned}$$

We need the valuation $B(x_1, \ldots, x_8) = (x_1 \approx a) \rightarrow (x_2 \approx a \wedge \cdots \wedge x_8 \approx a)$ and the pattern

$$\begin{aligned} &\{(\alpha^{n+1}\beta^{m+1}\gamma^{p+1}, \alpha^n\beta^m\gamma^p, \alpha^{n+1}\beta^m\gamma^p, \alpha^n\beta^{m+1}\gamma^p, \alpha^n\beta^m\gamma^{p+1}, \\ &\alpha^{n+1}\beta^{m+1}\gamma^p, \alpha^n\beta^{m+1}\gamma^{p+1}, \alpha^{n+1}\beta^m\gamma^{p+1}) \mid n, m, p \in \mathbb{N}_+\}. \end{aligned}$$

This can be rewritten as

$$(\alpha,1,\alpha,1,1,\alpha,1,\alpha)\cdot(\alpha)_8^*\cdot(\beta,1,1,\beta,1,\beta,\beta,1)\cdot(\beta)_8^*\cdot(\gamma,1,1,1,\gamma,1,\gamma,\gamma)\cdot(\gamma)_8^* \in (\mathbf{k}\mathbb{M})^3,$$

where recall that we use the notation $(x)_8$ to represent a vector with 8 equal components x. The other cell condition can be rewritten similarly, and it is also in $(\mathbf{k}\mathbb{M})^3$. With this we can define $W_{\text{3-Cell}}$. Finally we need to make the three sides of the cube equal:

$$W_{\text{3-edges}} = \textbf{Synt}\begin{pmatrix} \approx & \approx & \approx \\ (\alpha,\beta)^* & (\alpha,\gamma)^* & (\beta,\gamma)^* \end{pmatrix}.$$

The full manifold is $W_{\text{3-poly}} = W_{\text{3-grid}} \cap W_{\text{3-cell}} \cap W_{\text{3-edges}}$, and hence $L_{n^3\text{-poly}} \in (\mathbf{k}\mathbb{M})^3/\mathbb{M}\mathbf{k}$.

More in general we call *polynomial languages* $L_{p(n)\text{-poly}} = \{a^{p(n)} \mid n \in \mathbb{N}_+\}$ where $p(n)$ is a polynomial with coefficients in $\mathbb{N}_+$. The *degree* of a polynomial language is the degree of the polynomial. Then we notate PG_d the class of polynomial languages with degree d.

The place in the bi-hierarchy of PG_d for $d = 0$ and $d = 1$ are a bit special because the manifolds in the examples consume some patterns in order to trim the trees and other subsidiary questions. But by the previous theorems we have $\mathsf{PG}_0 \subseteq \mathsf{FN} = \mathbf{k}^2\mathbb{H}/\mathbf{k}$ and that $\mathsf{PG}_1 \subseteq \mathsf{RG} \subseteq \mathsf{CF} = \mathbf{k}^2\mathbb{H}/\mathbb{H}\mathbf{k}$. For the case $d \geq 2$ we have a general formula:

Theorem 9.13. $\mathsf{PG}_d \subseteq {}^{(\mathbb{M}\mathbf{k})^d}\!\big/_{\mathbb{M}\mathbf{k}}$ *for any* $d \geq 2$.

Proof. First one must prove the result for monomials: $L_{n^d\text{-poly}} \subseteq (\mathbb{M}\mathbf{k})^d/(\mathbb{M}\mathbf{k})$. The cases quadratic and cubic viewed in the above Examples 33 and 34 can be extended using similar geometric tricks, although it is notationally a little tedious. The manifold $W_{d\text{-grid}}$ has type $\mathbf{k}^2\mathbb{M}$ and the manifold and $W_{d\text{-edges}}$ has type $\mathbb{M}$, regardless of the degree d. Really the type $(\mathbf{k}\mathbb{M})^d$ is contributed by the manifold $W_{d\text{-cell}}$.

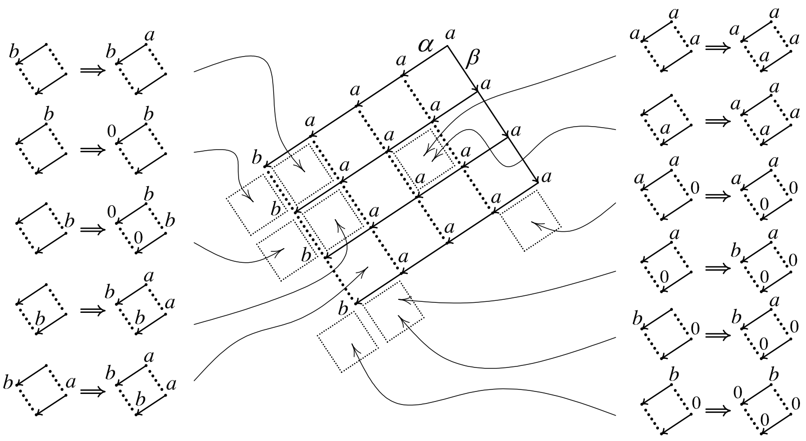
Now notice that a polynomial with coefficients in $\mathbb{N}_+$ is a sum of monomials (for example $3n^5 + 2n^3 = n^5 + n^5 + n^5 + n^3 + n^3$). Let us see how to sum two monomials, n^p, n^q. Geometrically we join (with disjoint syntactic functions) hypercubes of p and q dimensions with the same root. This is equivalent to considering the manifold $W_{p\text{-poly}} \cap W_{q\text{-poly}}$. We just need to make equal the sides of both hypercubes, which can be done by a similar manifold to $W_{d\text{-edges}}$. We are done. □

Now we consider the language $L_{\text{noin}} = \{(ba^n)^n \mid n \in \mathbb{N}^+\}$, which was proved by Gilman (1996) not to be an indexed language. This language has a quadratic growth $n^2 + n$ and hence it is very close to the polynomial language $L_{n^2\text{-poly}}$; this is born out in its classification in the general bi-hierarchy as the following example demonstrates.

Example 35. A NON-INDEXED LANGUAGE. We can show that:

$$L_{\text{noin}} \in {}^{(\mathbf{k}\mathbb{M})^2}\!\big/_{\mathbb{M}\mathbf{k}}.$$

We use a geometric construction similar to that for the 2-polynomial language. We use the same grid and the same cell conditions with some additional conditions depicted in the figure below. These conditions are redundant but they cover all the possibilities. Finally we need to make equal the sides of the square of a's. That is, the axes α^*, β^* are equal except for the b.



This works as follows. The totally null grid satisfies all these conditions, so the empty string is in the language. Now note that the fourth cell condition in the right column ensures that we have at least one b on the grid (because the grid is not null). From this b we can fill out all the grid following the grid conditions until we reach the axis. If the side on the axis β^* is greater than the other side (without counting the line of b's), then there is no syntagma with that initial b. If both sides coincides then the syntagma is in the manifold. With this we achieve that this syntagma contains a square number of a's, say n^2, and n b's.

Now it is easy to linearize with a pair of arrangements to obtain the string $(ba^n)^n$. If the side on the axis β^* is shorter than the other side we fill this axis with a's and then continue filling out the rest of the loci with the cell conditions until we have a square of a's. And then we can linearize it to obtain a string in the language for which we use the projective arrangement $A = \zeta^*\alpha + 1 + \zeta^*\beta$.

This process is exhaustive: for any $n \in \mathbb{N}_+$ we can find a syntagma in the manifold such that it yields $(ba^n)^n$. All the involved patterns are the same as for the 2-polynomial language.

9.3 Anti-Context-Free Languages

We introduced in the last chapter the notion of symmetry and anti-classes of a given class. In view of the above results it is trivial to prove that $-\mathsf{AB} = \mathsf{AB}$ and $-\mathsf{FN} = \mathsf{FN}$. So in these cases the operator "anti-" does not contribute anything. However we can show that $-\mathsf{CF} \neq \mathsf{CF}$. For example, since L_{squa} and L_{copy} are symmetric languages, and $L_{\text{squa}} \in \mathsf{CF}$, $L_{\text{copy}} \in -\mathsf{CF}$, and it is known that $L_{\text{copy}} \notin \mathsf{CF}$, follows that anti-context-free languages form a new class in the classical landscape of formal languages. Let us see some basic properties of the anti-context-free languages:

Proposition 9.14. *The following properties hold:*

(i) $\mathsf{FN} \subseteq -\mathsf{CF} \cap \mathsf{CF}$*;*

(ii) $L_{\text{mirr}}, L_{2\text{-resp}} \in \mathsf{CF} \cap -\mathsf{CF}$*;*

(iii) $L_{\text{q-copy}}, L_{q\text{-resp}} \in -\mathsf{CF}$*, for all* $q \geq 2$*;*

(iv) $-(-(\mathsf{CF})) = \mathsf{CF}$*;*

(v) $-\mathsf{CF}$ *is semi-linear.*

Proof. The majority of the statements are trivial by the results obtained for the bi-hierarchies, by Corollary 9.6 and by the fact that the operator anti- is monotonic (Corollary 8.9). (i) Since $\mathsf{FN} = \mathbf{k}^2\mathbb{H}/\mathbf{k} \subseteq \mathsf{CF}$ then $-\mathsf{FN} = -\mathbf{k}^2\mathbb{H}/\mathbf{k} \subseteq -\mathsf{CF}$. But notice that $-(\mathbf{k}^2\mathbb{H}/\mathbf{k}) = \mathbf{k}^2\mathbb{H}/\mathbf{k}$, i.e. the class is self-symmetric, and thus $-\mathsf{FN} = \mathsf{FN}$. Therefore $\mathsf{FN} \subseteq -\mathsf{CF} \cap \mathsf{CF}$. For (ii) just recall that $L_{\text{mirr}} \in \mathbf{k}^2\mathbb{H}/\mathbb{H}\mathbf{k} \cap \mathbb{H}\mathbf{k}^2/\mathbb{H}\mathbf{k}$. We can use the same shape of trees (up to the letters) and linearizations from the mirror language for the language $L_{2\text{-resp}}$ and we obtain also $L_{2\text{-resp}} \in \mathbf{k}^2\mathbb{H}/\mathbb{H}\mathbf{k} \cap \mathbb{H}\mathbf{k}^2/\mathbb{H}\mathbf{k}$. (iii) We saw this in §9.1.4. (iv) Since $\mathsf{CF} = \mathbf{k}^2\mathbb{H}/\mathbb{H}\mathbf{k}$, by the Corollary 8.9, we have $-(-(\mathsf{CF})) = \mathsf{CF}$. (v) Theorem 9.10 says that the semi-linearity is preserved by the operator anti-, and CF is semi-linear by the Parikh Theorem. □

Speaking *grosso modo*, CF comprehends languages with nested embedded dependencies (like well-balanced parentheses), while $-\mathsf{CF}$ comprehends languages with cross-serial dependencies; recall Example 29. Algebraic dependency grammar reveals that these are symmetric phenomena; Fig. 9.2 shows a scheme of both classes.

In addition, CF and $-\mathsf{CF}$ are semi-linear classes. We do not know if anti-context-free languages are parsable in polynomial time. However we suspect that more properties of ordinary context-free languages are transplantable to the anti-class. If this is so, we could consider the family $\mathsf{CF} \cup -\mathsf{CF}$ as a lower bound for the weak capacity of those mildly-context-sensitive formalisms which tolerate an unbounded copying power (as for example LCFRS's or MCFG's , but not TAG's nor LIG's).[5]

[5] By unbounded copying power we mean that the formalism can generate any q-copy language.

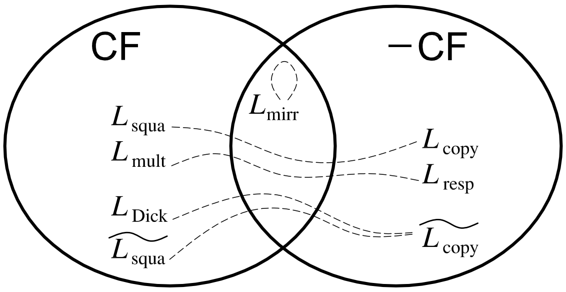


Figure 9.2: Diagram of the context-free and anti-context-free classes with some examples of languages. Two languages linked by a dashed line indicate that these are symmetric languages.

We can take advantage of the accumulated results to establish the strictness of some inclusions in $\mathcal{BH}(\mathbf{k}, \mathbb{H})$. We know that $\mathsf{CF} = \mathbf{k}^2\mathbb{H}/\mathbb{H}\mathbf{k}$ and that $-\mathsf{CF} = \mathbb{H}\mathbf{k}^2/\mathbb{H}\mathbf{k}$; thus $\mathbf{k}^2\mathbb{H}/\mathbb{H}\mathbf{k} \subsetneq \mathbb{H}\mathbf{k}^2\mathbb{H}/\mathbf{k}\mathbb{H}$. If we suppose that $\mathbf{k}^2\mathbb{H}/\mathbb{H}\mathbf{k} = \mathbb{H}\mathbf{k}^2\mathbb{H}/\mathbf{k}\mathbb{H}$, then since $-\mathsf{CF} \subseteq \mathbb{H}\mathbf{k}^2\mathbb{H}/\mathbf{k}\mathbb{H}$, we would have $-\mathsf{CF} \subseteq \mathsf{CF}$, but then $L_{\text{copy}} \in \mathsf{CF}$ which is absurd. Therefore $\mathbf{k}^2\mathbb{H}/\mathbb{H}\mathbf{k} \neq \mathbb{H}\mathbf{k}^2\mathbb{H}/\mathbf{k}\mathbb{H}$. By symmetry of the bi-hierarchy, $\mathbb{H}\mathbf{k}^2/\mathbb{H}\mathbf{k} \subsetneq \mathbb{H}\mathbf{k}^2\mathbb{H}/\mathbf{k}\mathbb{H}$.

9.4 A Fragment of the Homogeneous Bi-Hierarchy

Bi-hierarchies are rich systems for the classification of algebraic dependency languages, for which we have just explored the simplest classes. The results collected in these last two chapters sketch the character of these hierarchies. For instance we know that they are perfectly symmetric under the operator anti- (Corollary 8.9).

The homogeneous case seems to be the most feasible bi-hierarchy. We consider a small fragment of the homogeneous bi-hierarchy. Fig. 9.3 depicts a diagram of inclusions for the lowest positions. Fig. 9.4 shows the same diagram with the characterizations and relations obtained in this chapter.

In order to draw this diagram we have considered only classes X/Y such that $|X| + |Y| \leq 5$ which leads us to consider $\sum_{n=0}^{5}(n+1)2^n = 1137$ potential classes. However by using all the previous theorems these classes reduce to the following seventeen classes which are depicted in the diagram.[6]

In addition we have to consider possible subclasses of each X/Y which are not trivial, i.e. those which are not obtained by considering subsequences of X and Y.[7] We have the ascending chains $\mathbf{k}\mathbb{H}/\mathbf{k} \subseteq \mathbf{k}\mathbb{H}/\mathbf{k}^2 \subseteq \mathbf{k}\mathbb{H}/\mathbf{k}^3 \subseteq \mathbf{k}\mathbb{H}/\mathbf{k}^4 \subseteq \cdots \subseteq \mathbf{k}^2\mathbb{H}/\mathbf{k} = \mathsf{FN}$; and $\mathbb{H}\mathbf{k}\mathbb{H}/\mathbf{k} \subseteq \mathbb{H}\mathbf{k}\mathbb{H}/\mathbf{k}^2 \subseteq \mathbb{H}\mathbf{k}\mathbb{H}/\mathbf{k}^3 \subseteq \mathbb{H}\mathbf{k}\mathbb{H}/\mathbf{k}^4 \subseteq \cdots \subseteq \mathbf{k}^2\mathbb{H}/\mathbf{k} = \mathsf{FN}$. However we do

[6] Namely: $\mathbf{1/1}$ $\mathbf{k}\mathbb{H}/\mathbf{k}$, $\mathbf{k}\mathbb{H}/\mathbf{k}^2$, $\mathbf{k}\mathbb{H}/\mathbf{k}^3$, $\mathbb{H}\mathbf{k}\mathbb{H}/\mathbf{k}$, $\mathbb{H}\mathbf{k}\mathbb{H}/\mathbf{k}^2$, $\mathbf{k}^2\mathbb{H}/\mathbf{k}$, $\mathbf{k}\mathbb{H}/\mathbb{H}\mathbf{k}$, $\mathbf{k}\mathbb{H}/\mathbf{k}\mathbb{H}$, $\mathbf{k}^2\mathbb{H}/\mathbb{H}\mathbf{k}$, $\mathbf{k}^2\mathbb{H}/\mathbf{k}\mathbb{H}$, $\mathbb{H}\mathbf{k}\mathbb{H}/\mathbb{H}\mathbf{k}$, $\mathbf{k}\mathbb{H}\mathbf{k}/\mathbb{H}\mathbf{k}$, $\mathbf{k}\mathbb{H}/\mathbf{k}^2\mathbb{H}$, $\mathbf{k}\mathbb{H}/\mathbb{H}\mathbf{k}^2$, $\mathbf{k}\mathbb{H}/\mathbf{k}\mathbb{H}\mathbf{k}$, $\mathbf{k}\mathbb{H}/\mathbb{H}\mathbf{k}\mathbb{H}$. The diagram also includes at the top the class $\mathbf{k}^2\mathbb{H}/\mathbf{k}\mathbb{H}\mathbf{k}$ which contains the class $\mathsf{CF} \cup -\mathsf{CF}$.

[7] Notice that in general $X/Y \subseteq X'/Y'$ does not implies that $X \sqsubseteq X', Y \sqsubseteq Y'$, nor $|X|+|Y| \leq |X'|+|Y'|$. For example $\mathsf{AB} = 1/\mathbf{k}^{100} \subset \mathbf{k}^2\mathbb{H}/\mathbb{H}\mathbf{k} = \mathsf{CF}$.

not know if they collapse at some level, whereby these classes are not depicted in the diagram. Similar phenomena could affect higher positions.

The inclusions in the diagram are not necessarily proper (see Fig. 9.5 for known proper inlcusions), and it is not known whether this bi-hierarchy is finite. Although there remains some open questions, Fig. 9.3 should be a good first approximation to this space of classes of languages.

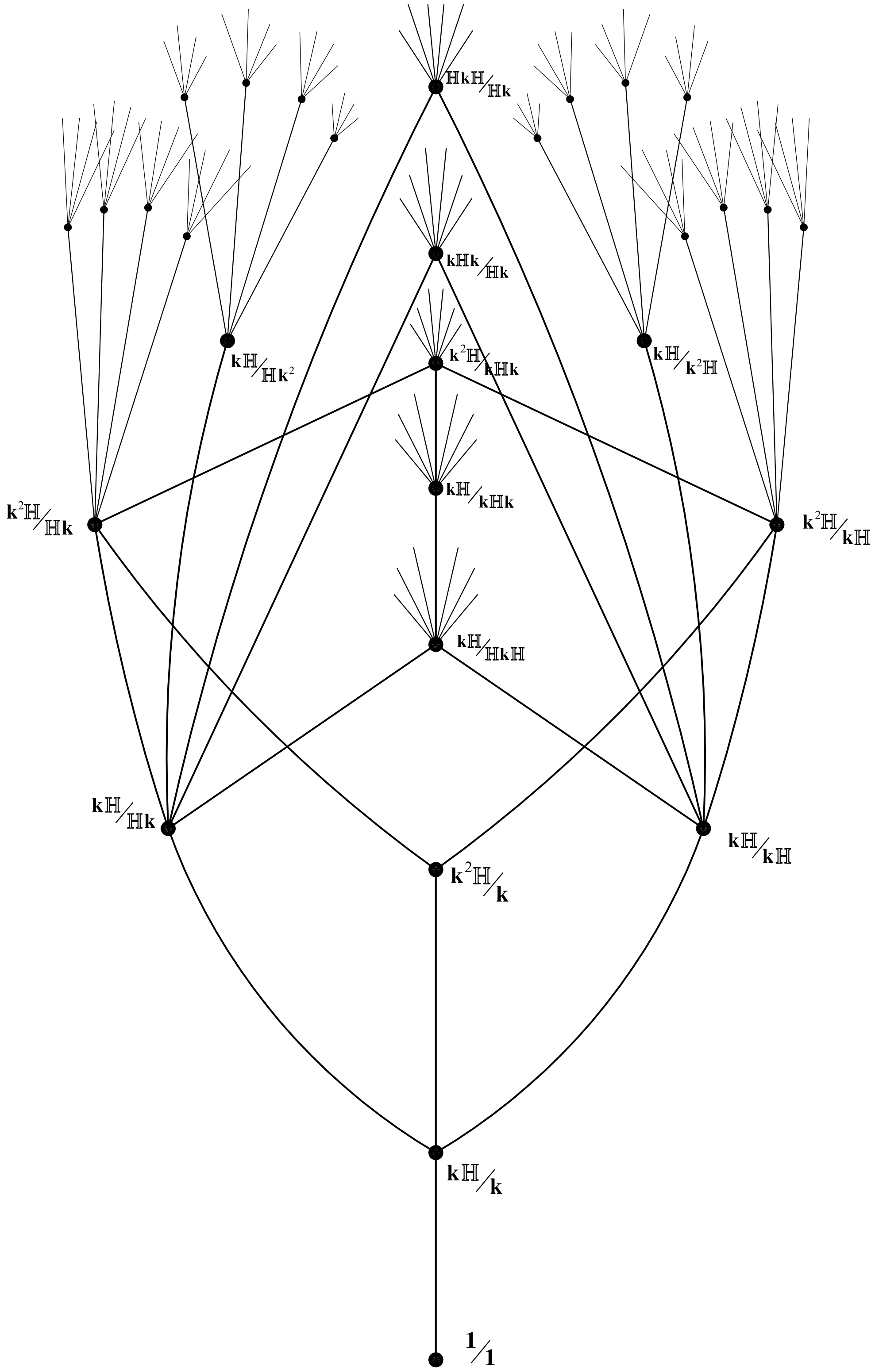

$\mathbb{H}\mathbf{k}\mathbb{H}/\mathbb{H}\mathbf{k}$
$\mathbf{k}\mathbb{H}\mathbf{k}/\mathbb{H}\mathbf{k}$
$\mathbf{k}\mathbb{H}/\mathbb{H}\mathbf{k}^2$
$\mathbf{k}^2\mathbb{H}/\mathbf{k}\mathbb{H}\mathbf{k}$
$\mathbf{k}\mathbb{H}/\mathbf{k}^2\mathbb{H}$
$\mathbf{k}^2\mathbb{H}/\mathbb{H}\mathbf{k}$
$\mathbf{k}\mathbb{H}/\mathbf{k}\mathbb{H}\mathbf{k}$
$\mathbf{k}^2\mathbb{H}/\mathbf{k}\mathbb{H}$
$\mathbf{k}\mathbb{H}/\mathbb{H}\mathbf{k}\mathbb{H}$
$\mathbf{k}\mathbb{H}/\mathbb{H}\mathbf{k}$
$\mathbf{k}\mathbb{H}/\mathbf{k}\mathbb{H}$
$\mathbf{k}^2\mathbb{H}/\mathbf{k}$
$\mathbf{k}\mathbb{H}/\mathbf{k}$
$\mathbf{1}/\mathbf{1}$

Figure 9.3: diagram of inclusions in the homogeneous bi-hierarchy.

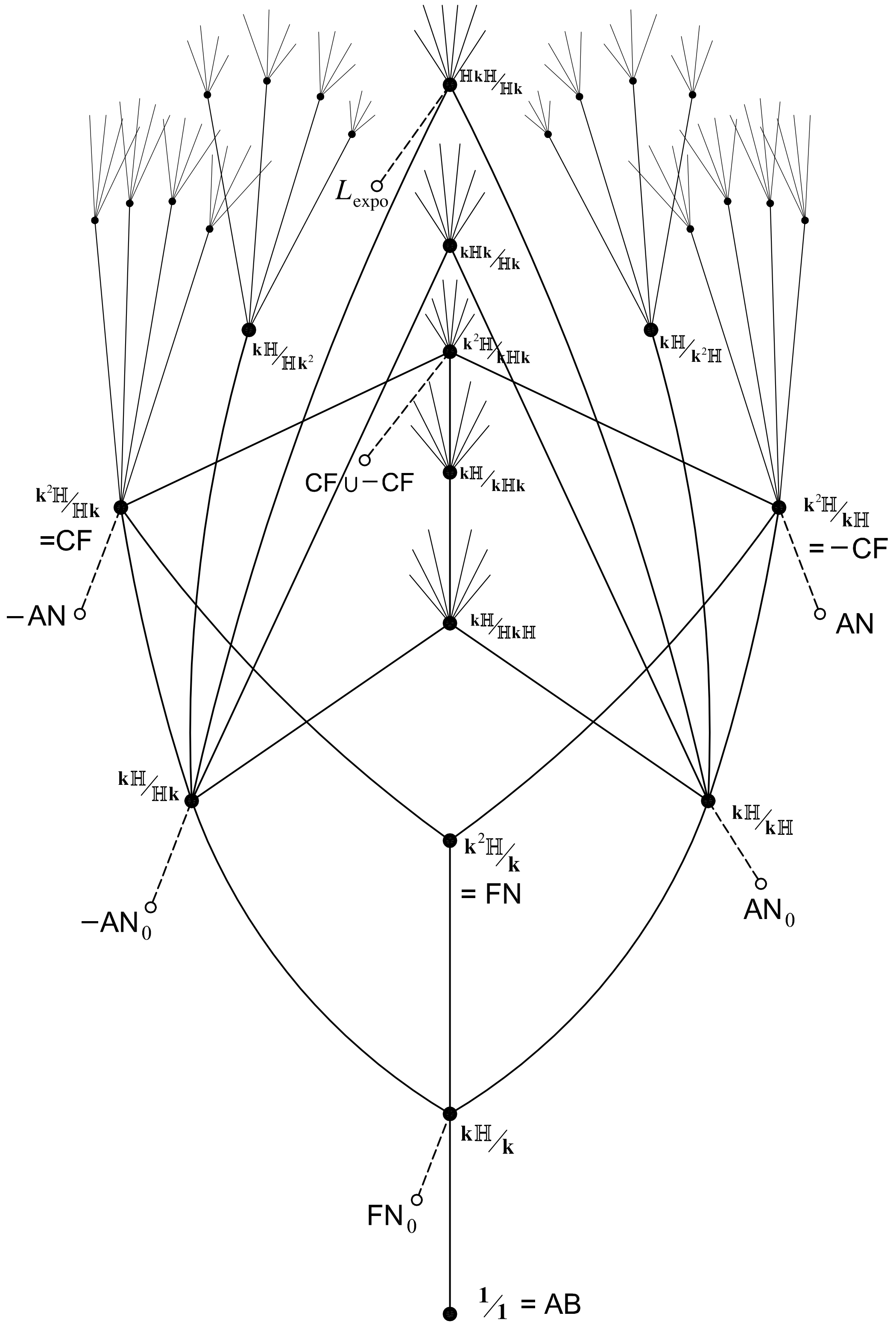

$\mathbb{H}\mathbf{k}\mathbb{H}/\mathbb{H}\mathbf{k}$
L_{expo}
$\mathbf{k}\mathbb{H}\mathbf{k}/\mathbb{H}\mathbf{k}$
$\mathbf{k}\mathbb{H}/\mathbb{H}\mathbf{k}^2$
$\mathbf{k}^2\mathbb{H}/\mathbf{k}\mathbb{H}\mathbf{k}$
$\mathbf{k}\mathbb{H}/\mathbf{k}^2\mathbb{H}$
$\mathsf{CF}\cup-\mathsf{CF}$
$\mathbf{k}\mathbb{H}/\mathbf{k}\mathbb{H}\mathbf{k}$
$\mathbf{k}^2\mathbb{H}/\mathbb{H}\mathbf{k}$
=CF
$\mathbf{k}^2\mathbb{H}/\mathbf{k}\mathbb{H}$
= −CF
−AN
AN
$\mathbf{k}\mathbb{H}/\mathbb{H}\mathbf{k}\mathbb{H}$
$\mathbf{k}\mathbb{H}/\mathbb{H}\mathbf{k}$
$\mathbf{k}\mathbb{H}/\mathbf{k}\mathbb{H}$
$\mathbf{k}^2\mathbb{H}/\mathbf{k}$
= FN
$-\mathsf{AN}_0$
AN_0
$\mathbf{k}\mathbb{H}/\mathbf{k}$
FN_0
$\mathbf{1}/\mathbf{1}$ = AB

Figure 9.4: diagram of inclusions in the homogeneous bi-hierarchy (•) together with the viewed classes from this and the last chapter (∘).

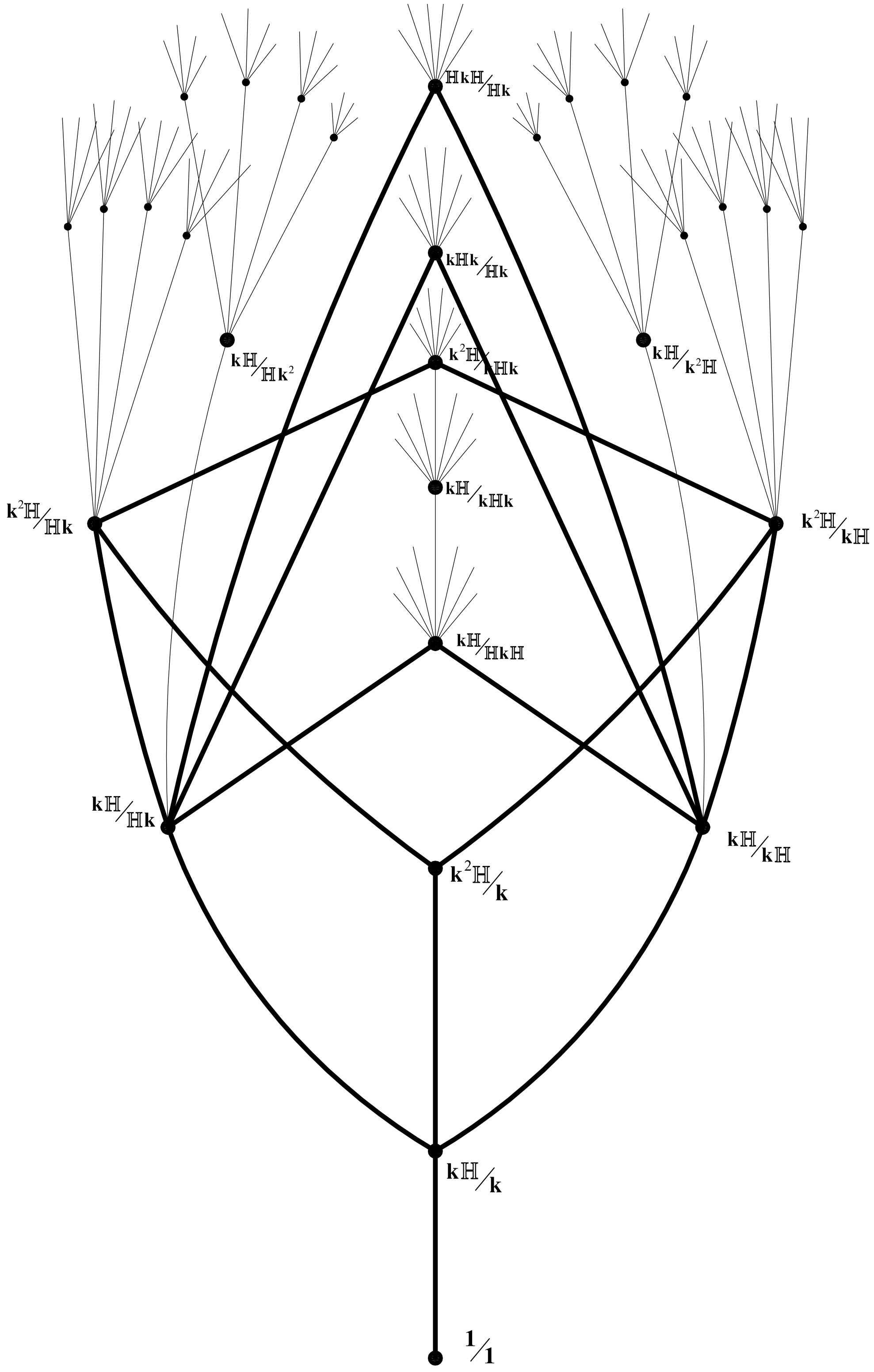

$\mathbb{H}k\mathbb{H}/\mathbb{H}k$
$k\mathbb{H}k/\mathbb{H}k$
$k\mathbb{H}/\mathbb{H}k^{2}$
$k^{2}\mathbb{H}/k\mathbb{H}k$
$k\mathbb{H}/k^{2}\mathbb{H}$
$k\mathbb{H}/k\mathbb{H}k$
$k^{2}\mathbb{H}/\mathbb{H}k$
$k^{2}\mathbb{H}/k\mathbb{H}$
$k\mathbb{H}/\mathbb{H}k\mathbb{H}$
$k\mathbb{H}/\mathbb{H}k$
$k\mathbb{H}/k\mathbb{H}$
$k^{2}\mathbb{H}/k$
$k\mathbb{H}/k$
$1/1$

Figure 9.5: diagram of known proper inclusions (in bold line) in the homogeneous bi-hierarchy.

10

Natural Language Constructions: Manifolds

This chapter and the following deal with natural language. Our purpose is to test the machinery presented in the previous chapters. The best way to achieve this is to confront our model with as many natural natural language constructions as we can. We do not provide a complete grammar, but choose a range of examples for their relevance. Here we study manifolds for natural phenomena and, in the next chapter, linearizations.

First of all we introduce a system of syntactic functions and some related concepts. With this we will examine several natural constructions involving manifolds. A first stage to establish a grammar consists in defining rules of the flow of syntactic functions attending valence and selection. A second stage consists in defining local agreements. These two points can be achieved using simple patterns.

Cluster functions are functions which can be iterated indefinitely. They are responsible for long distance agreements. The effect of implementing clusters in a pattern is the interposition of a monoid in its description. So long distance agreements causes more complex patterns.

Finally we will implement three non-semi-linear constructions: Chinese numeration system, *Suffixaufnahme* in Old Georgian and Recursive Copy in Yoruba.

10.1 Some Previous Concepts

10.1.1 Classificatory Monoids

We are going to show several constructions which refer for the most part to the manifold component. In the last chapters we have studied the homogeneous and general bi-hierarchies. Here these natural constructions will require other specific patterns. We will see that in general local phenomena such as valence, selection or local agreements are described by manifolds of type $\mathbf{k}^p\mathbb{H}$. However, long distant agreement seems to appeal to pivoting monoids; recall that we notated $\mathbb{P}$ pivoting monoids, which are of the form $1 \oplus \xi^*$ where $\xi \subseteq \zeta$ (see Example 3). These appear in combination with the homogeneous monoids, whereby we need as a classificatory monoid $\{\mathbf{k}, \mathbb{P}, \mathbb{H}\}^*$.

Furthermore, extreme constructions examined at the end of the chapter exceed these patterns. So we will need to appeal to other kinds of types in $\{\mathbf{k}, \mathbb{H}, \mathbb{G}\}^*$.

Every construction can be regarded as a sublanguage of a natural language. To obtain a full natural language we have to consider all the manifolds together and linearize it. The linearization for these constructions are in general independent of the manifolds. For example two words agree morphologically independently of the way we linearize the syntagma. For simplicity the reader can suppose a projective linearization. We will not make special comments about linearizations, which will be treated in the following chapter.

We comment issues regarding weak capacity of these languages in Chapter 12. Here we concentrate on describing natural constructions and classify them according their structural character.

10.1.2 Lexical Categories

Although we are not going to construct a grammar, in order to implement one partially we need to introduce some concepts. A first is the *lexical category*. By this we mean simply a subset of Σ. For example:

Verb = {*catch, catches, caught, smile, smiles, smiled*},
TransitiveVerb = {*catch, catches, caught*},
IntransitiveVerb = {*smile, smiles, smiled*},
Noun = {*John, Mary, boy, boys, frog, frogs*},
Name = {*John, Mary*},
Determiner = {*a, the, this, these, that, those*},
Adjective = {*big, small*},
Modifier = {*very*},
Singular = {*a, the, this, that, John, Mary, boy, frog, catches, caught, smiles, smiled*},
Plural = {*the, these, those, boys, frogs, catch, caught, smile, smiled*}.

We use categories in order to construct valuations. Sometimes it is necessary that the null word 0 belong to a certain category. Recall that given a subset $\Sigma' \subseteq \Sigma$ we notated $\Sigma'_+ = \Sigma' \cup \{0\}$. So sets such as: $\mathbf{Noun}_+$ or $\mathbf{Adjective}_+$ will be used. Natural language uses a rich variety of such sets. Even though it would be really interesting to draft a comprehensive system of categories and show their interrelations, here we introduce just a few of them.

10.1.3 Organization of a Table of Functions, Primitive and Isotope Functions

Since we use a considerable number of syntactic functions we devote some lines to their organization. Appendix §B contains a detailed inventory of functions and examples. The name of the functions often varies among grammarians, however the meaning of a syntactic function is given by its combinatorial usage and its interrelations rather than by the name.[1]

Although the interplaying of the functions is quite complex, some regularities can be extracted regarding governance features and the lexical category with which they are related. Functions can be organized as in Table 10.1 which shows a generic table of functions on which a particular language draws, see Appendix B, Table B.7, Table B.8 and Table B.8.

This table is structured as the Cartesian product of groups of functions. We survey them. First we distinguish five groups: VERBAL, NOMINAL, MODIFIER, INTRODUCTORY, and COORDINATOR functions, according to their governing lexical category. Verbal functions yield verb phrases when we linearize them. Similarly nominal functions yield noun phrases.

Transversally the three first groups can be organized by isotopes. These give information about the lexical category of the governed word. We say that X is an *isotope* function of Y when they share the same semantic role but they differ on some syntactic peculiarities such as the lexical category selected or the position in a sentence. This is an informal notion which help to organize functions.[2] Mathematically they must be considered as different functions. We introduced six forms of isotope: PRIMITIVE, SUBORDINATION, INTERROGATION, RELATIVE PRONOUN, TOPICALIZATION, and PRONOUN.

Consider for example the primitive function object *Ob* which introduces a noun as the object of the main verb: *Kids hate vegetables*$_{Ob}$. If we want to introduce a subordinate clause as the object we must use the subordinate isotope Ob_S, for example *I know [that kids hate*$_{Ob_S}$ *vegetables]*, where - - - indicates the the head and __, the governed item. However if we want to form a question about the object we have to use the interrogative isotope of object: *What*$_{Ob_?}$ *do kids hate ?*. Beside the change of lexical category we need this isotope because in many languages subordinate clauses and main clauses linearize in different ways. The relative pronoun isotope for object is used to introduce the relative pronoun which announces the subordinate clause and which fills the role of object in the subordinate clause: *what*$_{Ob_R}$ *kids like is important*. Topicalization isotope for object is used to topicalize the object, as in *Vegetables*$_{Ob_!}$, *kids hate*. Fig. 10.1 collects the main features of the organization of a table of functions.

[1] For example, the word "modifier" can be a bit tricky since a lot of functions are in the end modifying something. Some grammarians call adjectives "modifiers" since they "modify" the noun. For us a modifier introduces a word which modifies the intensity or mode of a complementer. For example *very long trip* which modifies the adjective or *almost one hundred penguins* which modifies the quantifier.

[2] We borrow the name from Chemistry. Chemic isotopes are behaved under reactions very similarly. In our case, isotope functions are behaved semantically very similarly.

VERBAL													NOMINAL						MODIFIER				
UNITS	CANONICAL						ADJUNCT																
1	*Sb*	*Ob*	*In*	*Pd*	*Rg*	· · ·	*Ap*	*At*	*Am*	*Ab*	· · ·	*Qn*	*Dt*	*Ps*	*Ad*	*Nc*	· · ·	*Md*	*Ma*	· · ·	Primitive	
Ax	Sb_S	Ob_S	In_S	Pd_S	Rg_S	· · ·	Ap_S	At_S	Am_S	Ab_S	· · ·	Qn_S	Dt_S	Ps_S	Ad_S	Nc_S	· · ·	Md_S	Ma_S	· · ·	Subord.	Isotopes
· · ·	$Sb_?$	$Ob_?$	$In_?$	$Pd_?$	$Rg_?$	· · ·	$Ap_?$	$At_?$	$Am_?$	$Ab_?$	· · ·	$Qn_?$	$Dt_?$	$Ps_?$	$Ad_?$	$Nc_?$	· · ·	$Md_?$	$Ma_?$	· · ·	Interrog.	
Re	Sb_R	Ob_R	In_R	Pd_R	Rg_R	· · ·	Ap_R	At_R	Am_R	Ab_R	· · ·	Qn_R	Dt_R	Ps_R	Ad_R	Nc_R	· · ·	Md_R	Ma_R	· · ·	Relative Pronoun	
· · ·	$Sb_!$	$Ob_!$	$In_!$	$Pd_!$	$Rg_!$	· · ·	$Ap_!$	$At_!$	$Am_!$	$Ab_!$	· · ·	$Qn_!$	$Dt_!$	$Ps_!$	$Ad_!$	$Nc_!$	· · ·	$Md_!$	$Ma_!$	· · ·	Topical.	
· · ·	Sb_P	Ob_P	In_P	Pd_P	Rg_P	· · ·	Ap_P	At_P	Am_P	Ab_P	· · ·	Qn_P	Dt_P	Ps_P	Ad_P	Nc_P	· · ·	Md_P	Ma_P	· · ·	Pronoun	

INTRODUCTORY	
Ip	*Ic*

COORDINATION & CONNECTER			
Co_{and}	Co_{or}	Co_{if}	· · ·

▨ Cluster functions · · · Other functions of the same kind

Table 10.1: generic table of functions.

For practical purposes it is convenient to introduce the following notation:

$$\begin{aligned}
\Delta &= \text{Primitive Verbal Func.} = \{Sb, Ob, In, Pd, Rg, Ap, At, Am, \ldots\}\\
\Delta_S &= \text{Subordinated Verbal Func.} = \{Ax, Sb_S, Ob_S, In_S, Pd_S, Rg_S, Ap_S, At_S, Am_S, \ldots\}\\
\Delta_? &= \text{Interrogation Verbal Func.} = \{Sb_?, Ob_?, In_?, Pd_?, Ap_?, At_?, Am_?, \ldots\}\\
\Delta_R &= \text{Relative Verbal Func.} = \{Re, Sb_R, Ob_R, In_R, Pd_R, Ap_R, At_R, Am_R, \ldots\}\\
\Delta_! &= \text{Topicalized Verbal Func.} = \{Sb_!, Ob_!, In_!, Pd_!, Ap_!, At_!, Am_!, \ldots\}\\
\Delta_P &= \text{Pronoun Verbal Func.} = \{Sb_P, Ob_P, In_P, Pd_P, Ap_P, At_P, Am_P, \ldots\}
\end{aligned}$$

10.1.4 Orthogonality and Cluster Functions

We present two more concepts before listing examples. Examining syntagmata for natural languages we observe that the main groups of functions, VERBAL and NOMINAL, never appear mixed in a same level. We call this property *orthogonality*.

A *clustering group* is a group of functions that they can be concatenated by other functions in the same group. For example the determiner function *Dt* does not form a clustering group because we cannot find a sentence with a determiner which determines another determiner. Or for example the modifier function *Md* does form a clustering group: *very very very tall*. We have detected five clustering groups:

- Auxiliary cluster: $\{Ax\}$;
- Subordination cluster: $\Delta_S = \{Sb_S, Ob_S, \ldots\}$;
- Noun complement cluster: $\{Nc\}$;
- Modifier cluster: $\{Md\}$;
- Coordination cluster: $\{Co_{and}, Co_{or}, \ldots\}$.

These functions create loops in the system of arrangements and permit the language to be infinite. Let us see some examples. Subordination cluster: $\textit{she thinks}_1$ $\textit{that you believe}_{Ob_S}$ $\textit{that he said}_{Ob_S \cdot Ob_S}$ *that* .... Coordination cluster: $\textit{John}_1$, $\textit{Bill}_{Co_{and}}$, $\textit{Mary}_{Co^2_{and}}$ *and* $\textit{Peter}_{Co^3_{and}}$. Noun complement cluster: $\textit{The father}_1$ $\textit{of the mother}_{Nc}$ $\textit{of the father}_{Nc^2}$. Modifier cluster: $\textit{very}_{Md^3 \cdot Ad}$ $\textit{very}_{Md^2 \cdot Ad}$ $\textit{very}_{Md \cdot Ad}$ $\textit{long}_{Ad}$ *trip*.

10.2 Main Verb Principle

This principle states that all sentences must be analyzed with the main verb on the top of the syntagma. This excludes *headlines* from natural language sentences. So, titles of books like:

- Verbal functions are governed, or headed, by verbs. In a system of arrangements these form the verb complements. In order to clarify we have subdivided them in three subgroups: canonical, adjuncts and unities. Canonical functions are the most basic functions and most grammarians accept them for a syntactic representation. Adjuncts contextualize the *deixis*. Unities is a special group. Certain functions (*Ax* and *Re*) seem to be better accommodated in the tables if we understand them as isotopes of the identity 1. Transversally we have the isotopes:
  - Primitive verbal functions govern a noun;
  - Subordinate verbal functions govern a verb;
  - Interrogative verbal functions govern an interrogative pronoun;
  - Relative verbal functions govern a relative pronoun;
  - Topicalized verbal functions govern a noun which is topicalized;
  - Pronoun verbal functions govern a pronoun.
- Nominal functions are governed by nouns. In a system of arrangements these form the noun phrases. Transversally we have the isotopes:
  - Primitive nominal functions govern the category suggested by the noun of the function (e.g. quantifier function governs quantifiers, and so forth);
  - Subordinate nominal functions govern a verb;
  - Interrogative nominal functions govern an interrogative pronoun;
  - Topicalized nominal functions govern the category suggested by the noun of the function which is topicalized;
  - Relative nominal functions govern a relative pronoun.
- Modifier functions can be governed by several lexical categories: adjectives, adverbs, quantifiers or even modifiers themselves. Transversally we have the isotopes (topicalized and pronoun isotopes seem not to exist for modifier functions, at least for the examined languages):
  - Primitive modifier functions govern several categories;
  - Subordinate modifier functions govern a verb;
  - Interrogative modifier functions govern an interrogative pronoun;
  - Relative modifier functions govern a relative pronoun.
- Introductory functions govern prepositions and coordinators which are introduced isolatedly.
- Coordination and Connecter functions are mainly coordination functions which concatenate conjuncts and they can govern and be governed by any category and functions which connect clauses.

Figure 10.1: main features of the organization of the table of functions.

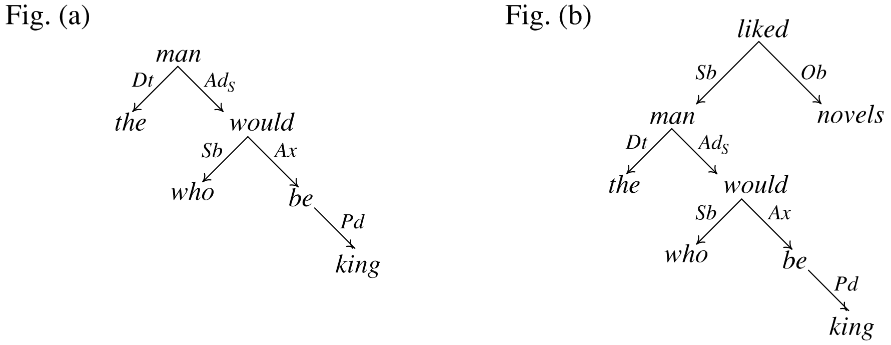


Figure 10.2: (a) non-correct syntagma because the root 1 is not a finite verb, (b) correct syntagma.

(24) *The man$_1$ who would be king

are excluded, see the syntagma in Fig. 10.2(a), while the sentence:

(25) The man $_{Sb}$ who would be king liked$_1$ novels.

is accepted, Fig. 10.2(b). This principle can be paraphrased by a valuation Verb(x) which returns the true value iff x satisfies:

$$\text{TopVerb}(x) = (x \in \textbf{FiniteVerb}),$$

where **FiniteVerb** is the set of all finite verbs. We need to apply the valuation on the top, which is the locus 1, so our pattern is the trivial pattern {1} (or simply 1). Thus the manifold:

$$\mathbf{Synt}\begin{pmatrix}\text{TopVerb}\\ 1\end{pmatrix}$$

only accepts syntagmata with a finite verb at the top. If we would want to incorporate headlines as part of the language, we have to remove this rule from the manifold.

10.3 Orthogonality, Valence and Selection

Since not all verbs accept the same arguments we have to define for each verb which arguments can follow the verb: which ones are optional, which ones are obligatory, and which ones are forbidden. For example a rule of English is that verbs always need a subject. Some grammarians say that the verb *selects* a subject to indicate that this function is obligatory.

These constraints must be extended to other categories such us nouns, adjectives or modifiers. For example, a finite verb cannot govern a determiner; and vice versa a determiner cannot govern a verb. These are always orthogonal functions. Fig. 10.3(a) shows verbal functions which are orthogonal to nominal functions, Fig. 10.3(b). Adjunct functions are usually optional in the context of verbal functions but they are also orthogonal to nominal functions, Fig. 10.3(c). This topic was already treated in Example 9 where we took a general manifold:

$$\textsc{govern}(\Sigma_0, \Sigma_1, \zeta_0) = \bigcap_{\lambda \in \zeta_0} \mathbf{Synt} \begin{pmatrix} x \in \Sigma_0 \longrightarrow y \in \Sigma_1 \\ (1, \lambda)(\zeta)^*_2 \end{pmatrix},$$

and we took the manifolds:

(b.1) $\textsc{govern}(\mathbf{Verb}, \{0\}, \{Dt, Ad, Md\})$,

(b.2) $\textsc{govern}(\mathbf{Noun}, \{0\}, \{Sb, Ob, Md\})$,

(b.3) $\textsc{govern}(\mathbf{Determiner}, \{0\}, \{Sb, Ob, Dt, Ad, Md\})$,

(b.4) $\textsc{govern}(\mathbf{Adjective}, \{0\}, \{Sb, Ob, Ad\})$,

(b.5) $\textsc{govern}(\mathbf{Modifier}, \{0\}, \{Sb, Ob, Dt, Ad\})$,

which imposed orthogonality constraints. We used the same kind of manifolds to impose selection constraints:

(c.1) $\textsc{govern}(\mathbf{Verb}, \mathbf{Noun}_+, \{Sb, Ob\})$,

(c.2) $\textsc{govern}(\mathbf{Noun}, \mathbf{Adjective}_+, \{Ad\})$,

(c.3) $\textsc{govern}(\mathbf{Noun}, \mathbf{Determiner}_+, \{Dt\})$,

(c.4) $\textsc{govern}(\mathbf{Adjective}, \mathbf{Modifier}_+, \{Md\})$,

(c.5) $\textsc{govern}(\mathbf{Modifier}, \mathbf{Modifier}_+, \{Md\})$,

(d.1) $\textsc{govern}(\mathbf{Verb}, \Sigma, \{Sb\})$,

(d.2) $\textsc{govern}(\mathbf{TransitiveVerb}, \Sigma, \{Ob\})$,

(d.3) $\textsc{govern}(\mathbf{IntransitiveVerb}, \{0\}, \{Ob\})$.

We can expand this set of rules with more syntactic functions, for example: $\textsc{govern}(\mathbf{Verb}, \mathbf{TimeAdjunct}_+, \{At\})$, where $\mathbf{TimeAdjunct} = \{$*yesterday, tomorrow,* $\ldots\}$. In §4.4 we concluded that all these features can be achieved locally through manifolds in $\mathbf{Man}(\mathbf{k}\mathbb{H})$. If we do not add more constraints these arguments will be optional. Of course we have to include that a determiner cannot govern an adjunct: $\textsc{govern}(\mathbf{Determiner}, \{0\}, \{At\})$, and so forth.

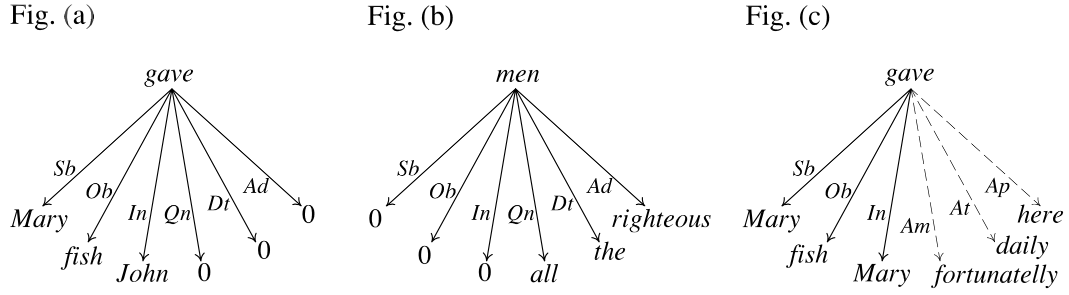


Figure 10.3: (a) and (b) show verbal functions which are orthogonal to nominal functions; (c) show optional adjuncts (dashed lines).

10.4 Local Agreements

10.4.1 Subject-Verb and Subject-Predicate Agreements

A large number of morphological agreements can be described by patterns in $\mathbf{k}\mathbb{H}$; see for example the agreement between the subject and the verb in English, Fig. 10.4(a). We just need a valuation which says that if the subject is a singular (plural) word, then the verb is a singular (plural) word, where:[3]

$$\textbf{Singular} = \{\textit{a, the, this, that, John, Mary, boy, catches, caught, smiled}\},$$
$$\textbf{Plural} = \{\textit{the, these, those, boys, catch, caught, smiled}\}.$$

We saw in Example 9 that this is achieved with the manifold:

$$\textsc{Rgovern}(\Sigma_0, \Sigma_1, \zeta_0) = \bigcap_{\lambda \in \zeta_0} \textbf{Synt}\begin{pmatrix} x \in \Sigma_0 \leftarrow y \in \Sigma_1 \\ (1, \lambda)(\zeta)_2^* \end{pmatrix}.$$

(e.1) $\textsc{Rgovern}(\textbf{Singular}, \textbf{Singular}, \{\mathit{Sb}, \mathit{Dt}\})$,

(e.2) $\textsc{Rgovern}(\textbf{Plural}, \textbf{Plural}, \{\mathit{Sb}, \mathit{Dt}\})$,

(e.1) $\textsc{govern}(\textbf{Singular}, \Sigma, \{\mathit{Dt}\})$.

Since we require that these agreements hold everywhere we use the pattern $(1, \mathit{Sb})\cdot(\zeta)_2^* \in \mathbf{k}\mathbb{H}$. Fig. 10.4(a) shows two pairs of places where the agreement can occur: the pair $(1, \mathit{Sb})$ and the pair $(\mathit{Ob}, \mathit{Sb}\cdot\mathit{Ob})$, both in the pattern $(1, \mathit{Sb})\cdot(\zeta)_2^*$.

Notice that when, for example, the subject is null, the valuation returns the true value, which is correct in Romance languages where the subject is optional. This permits that when some of the parts are null, there is nothing to agree.[4]

[3] Notice that some words can be singular and plural simultaneously.

[4] If we want to include the possibility of a gapped verb, for example, *John eats vegetables, and Mary (∅) meat*, we have to take the manifold $\textsc{Rgovern}(\textbf{Singular}, \textbf{Singular}_+, \{\mathit{Sb}\})$.

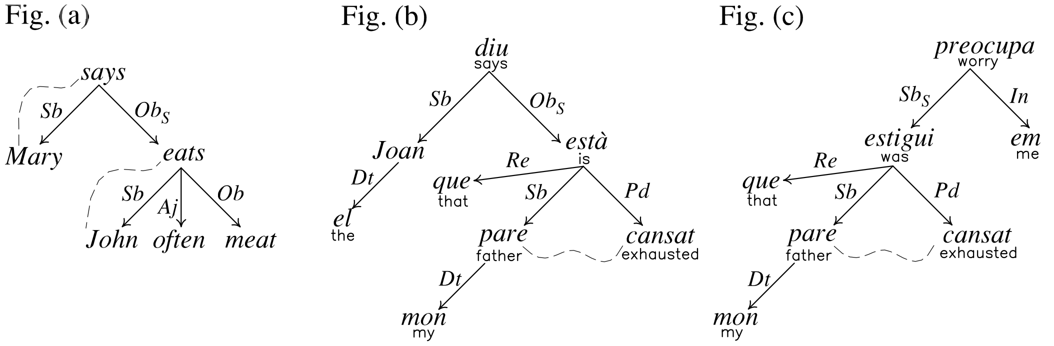


Figure 10.4: local agreements.

Romance languages exhibit a more complex usage of agreement. For example, the predicate (the attribute of a copulative verb) *Pd* and subject *Sb* must agree in gender and number. Fig. 10.4(b) and Fig. 10.4(c) show in Catalan that we have to use a pattern $(Sb, Pd)\cdot(\zeta)^*_2 \in \mathbf{k}\mathbb{H}$.[5] This changes slightly the manifolds:

$$\mathbf{Synt}\begin{pmatrix} x \in \mathbf{Singular} \rightarrow y \in \mathbf{Singular} \\ (Sb, Pd)(\zeta)^*_2 \end{pmatrix}.$$

This now says that if the subject is singular, then the predicate is; and the same manifold works for other agreement features.

We have called these agreements local because the agreed loci must be immediate family (parents, children or sibling). In algebraic terms this means that the involved constants are in **k** and then the pattern is in $\mathbf{k}\mathbb{M}$. The next example we will use a pattern in $\mathbf{k}^2\mathbb{M}$, but this can still be considered a local constraint (or if the reader wants, locally second order). Later we will see non-local agreements which manifests even in morphologically poor languages like English.

10.4.2 Obligatory Determiners in Catalan

In Catalan the subject generally needs a determiner. The following sentence (a) is allowed but not (b) which requires a determiner:

(26) a. Les/aquestes/algunes$_{Dt\cdot Sb}$ platges$_{Sb}$ estan desertes.
The/these/some$_{Dt\cdot Sb}$ beaches$_{Sb}$ are deserted
'The/these/some beaches are deserted.'

b. *platges estan desertes.
beaches are deserted
'beaches are deserted.'

[5] In §4.1 we already examined these cases linguistically.

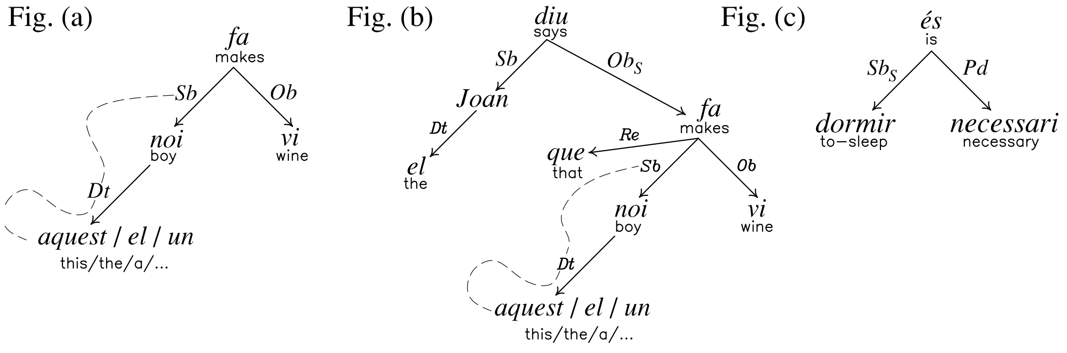


Figure 10.5: obligatory determiners in Catalan.

This is not necessary in other syntactic functions like object where the determiner is optional (although the semantic reading varies slightly):

(27) Jo compraré el/∅$_{Dt\cdot Ob}$ cafè$_{Ob}$.
I will-buy the/∅$_{Dt\cdot Ob}$ coffee$_{Ob}$
'I will buy the/∅ coffee.'

In Catalan determiners are even necessary for person names:

(28) El$_{Dt\cdot Sb}$ Joan$_{Sb}$ pren cafè a les 11:00.
The$_{Dt\cdot Sb}$ Joan$_{Sb}$ has coffee at the 11:00.
'Joan has coffee at 11:00'.

However the determiner is forgotten when the subject is a gerund verb,[6] a pronoun or a celebrity person name:

(29) a. *El$_{Dt\cdot Sb}$ dormir$_{Sb}$ és necessari.
The$_{Dt\cdot Sb}$ to-sleep$_{Sb}$ is necessary
'It is necessary to sleep.'

b. *La$_{Dt\cdot Sb}$ ella$_{Sb}$ vindrà demà.
The$_{Dt\cdot Sb}$ she$_{Sb}$ will-come tomorrow
'She is coming tomorrow.'

c. *El$_{Dt\cdot Sb}$ Salvador Dalí$_{Sb}$ no acabà aquesta obra.
The$_{Dt\cdot Sb}$ Salvador Dalí$_{Sb}$ not finished this work
'Salvador Dalí didn't finished this work.'

This can be summarized as: *a non-null subject is not a verb nor a pronoun nor a celebrity noun, then this subject forces a determiner* (and conversely *when the determiner of a*

[6] Other Romance language tolerates a determiner for gerunds, like spoken Spanish.

subject appears, the subject is not null, nor a verb, nor a celebrity noun).[7] This can be translated into valuations as:

$$B(x, y) = (x \notin \mathbf{Verb} \cup \mathbf{Pronoun} \cup \mathbf{CelebrityNoun} \cup \{0\}) \leftrightarrow (y \not\approx 0),$$

This valuation is applied to the loci: $(Sb, Dt\cdot Sb)$. But as in the previous examples this rule must be reproduced throughout the syntagma. So we have to multiply by the pattern $(\zeta)_2^*$. So the whole pattern is:

$$(Sb, Dt\cdot Sb)\cdot(\zeta)_2^* \in \mathbf{k}^2\mathbb{H}.$$

10.5 Long Distance Agreements

Most morphological agreements occurs locally: that is, the pattern inhabits $\mathbf{k}\mathbb{H}$ or $\mathbf{k}^2\mathbb{H}$ or in general $\mathbf{k}^p\mathbb{H}$ for some $p \geq 0$. The exponent p means that the distance in the tree between the linked parts cannot be greater than $2p$. However there are agreements where the distance is unbounded. Unbounded agreement is a consequence of clustering functions.

10.5.1 Long Distance Agreement of Predicates in English and Romance Languages

Consider the following sentences in English:

(30) a. $\underline{\text{She}}_{Sb}$ wants to be an $\underline{\text{actress}}_{Pd\cdot Obs_S}$.
b. $\underline{\text{She}}$ would want to be an $\underline{\text{actress}}_{Pd\cdot Obs_S\cdot Ax}$.
c. $\underline{\text{She}}$ would want to try to be an $\underline{\text{actress}}_{Pd\cdot Obs_S\cdot Obs_S\cdot Ax}$.

(31) a. *She wants to be an actor.
b. *She would want to be an actor.
c. *She would want to try to be an actor.

The agreement is not semantical (or not exclusively); in Romance languages gender agreement occurs extensively between words which do not have a natural gender.

[7] The situation is even more complex, because Catalan allows one to say: *el Dalí dels anys 70 era irreverent* with the determiner *el* (in English: *the Dalí from the 70's was irreverent*) but not **Dalí dels anys 70 era irreverent*, without the determiner. The noun complement *from the 70's* supposes several Dalís from several artistic periods, and we need an article which chooses one of them. This contingency can be also implemented by a valuation and a pattern with a pattern, with arity 3: one component for the determiner, one for the noun and one for the noun complement. Patterns of arity 3 are rare in natural language.

Fig. 10.6(a) shows an unbounded cluster between the main copulative verb and the predicate. If we ignore the verbal cluster the pattern is:

$$(Sb, At)(\zeta)_2^* \in \mathbf{k}\mathbb{H},$$

as in the case of local agreements, §10.4. But we have to consider the possibility of interposing several verbs which means a collection of possible verbal auxiliary functions and subordinated object functions. So the pattern is (we underline the new submonoid):

$$(Sb, Pd) \cdot \underline{\{(1, Ax), (1, Ob_S)\}^*} \cdot (\zeta)_2^* \in \mathbf{k}\underline{\mathbb{P}}\mathbb{H}.$$

If we take $\Delta = \{Ax, Ob_S\}$ we can abbreviate:

$$(Sb, Pd) \cdot (1 \oplus \Delta^*) \cdot (\zeta)_2^* \in \mathbf{k}\mathbb{P}\mathbb{H},$$

Notice that this pattern subsumes that above; we only have to realize that $(1, 1) \in (1 \oplus \Delta^*)$, and then:

$$(Sb, Pd) \cdot (\zeta)_2^* \subset (Sb, Pd) \cdot (1 \oplus \Delta^*) \cdot (\zeta)_2^*.$$

10.5.2 Long Distance Agreement of Coordinated Adjectives in Romance Languages

In Romance languages adjectives must agree in gender and number with the noun in a similar way to the case of predicates (attributes) and copulative verbs. Now we do not have a verb or a cluster of verbs. However we can add several adjectives and all of them must agree individually with the noun. Consider the following phrases in Catalan:

(32) Una noia encantadora, forçuda i treballadora.
A girl charming, strong and hard-working
'A charming, strong and hard-working girl.'

(33) *Un noia encantadora, forçuda i treballador-∅.

We commented in Chapter 1 that we advocate a style of analisis *a la* Mel'čuk, as in Fig. 10.6(b). We use syntactic functions named Co_{and} = *coordination-and*, Co_{or} = *coordination-or* and others similar, which concatenates several items (not only adjectives); we abbreviate Co. This sets the loci:

$$Ad,\ Co{\cdot}Ad,\ Co^2{\cdot}Ad,\ Co^3{\cdot}Ad, \ldots.$$

We have to enforce that all these loci behave as adjectives and that in addition they agree with the noun in the locus 1. So we need a manifold that says *if the word at the locus* 1 *is in* **Masculine** *then the word at the locus* $Co^n{\cdot}Ad$ *is in* **Masculine** ∩ **Adjective**. And the same for other agreement features. So the pattern should be $(1 \oplus Co^*)(1, Ad) \in \mathbb{P}\mathbf{k}$.

However, this constraint must hold at every place of the syntagma because the noun phrase containing the coordination can appear anywhere. So the pattern is really $(1 \oplus Co)^*{\cdot}(1, Ad){\cdot}(\zeta)_2^* \in \mathbb{P}\mathbf{k}\mathbb{H}$.

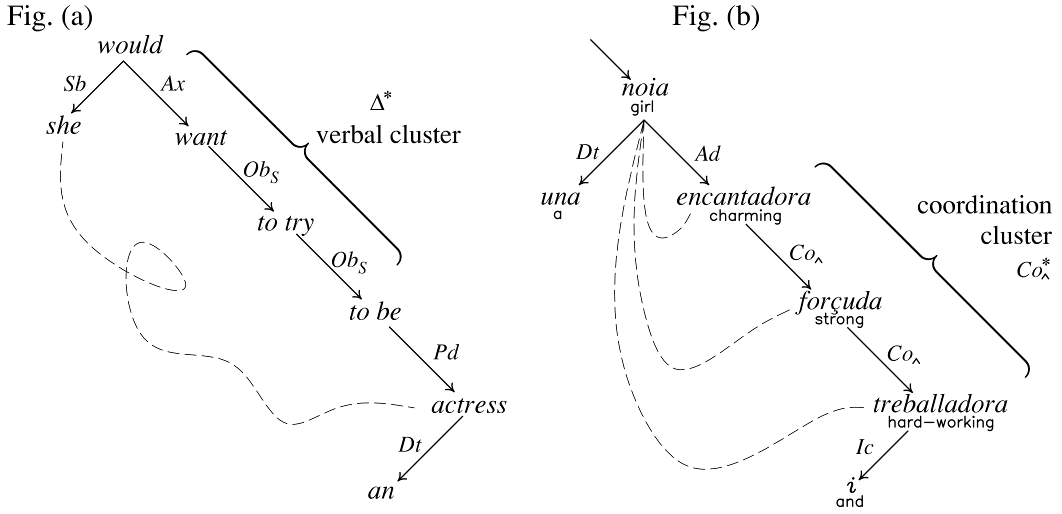


Figure 10.6: long distance agreement of coordinated adjectives in Romance languages.

10.5.3 Combination of Long Distance Agreement in Predicates and Coordinated Adjectives

It is possible to combine both constructions in sentences such as (see Fig. 10.7):

(34) Ella hauria de tornar a provar de ser una noia encantadora, forçuda
She would-have to be-again to try to be a girl charming, strong
i treballadora.
and hard-working
'She should try again to be a charming, strong and hard-working girl.'

To achieve the agreement of gender and number we need the pattern:

$$(Sb, 1)\cdot(1 \oplus Co^*)\cdot(1, Pd)\cdot(1 \oplus \Delta^*)\cdot(\zeta)_2^* \in \mathbf{k}\mathbb{P}\mathbf{k}\mathbb{P}\mathbb{H}.$$

This pattern generalizes doubly the pattern for the agreement of subject and predicate:

$$(Sb, 1)\cdot\underline{(1 \oplus Co^*)}\cdot(1, Pd)\cdot\underline{(1 \oplus \Delta^*)}\cdot(\zeta)_2^*,$$

since if we take $Co^0 = 1$ we have the pattern $(Sb, Pd) \cdot (1 \oplus \Delta^*) \cdot (\zeta)_2^*$, and if in addition we we take $1 \in \Delta^*$ we obtain the pattern $(Sb, Pd) \cdot (\zeta)_2^*$.

We see that in general long distance agreements are a generalization of local agreements. In a definition of a grammar for natural language we have to take the most general case. In addition notice that each cluster signifies the interposition of a new monoid $\mathbb{P}$ in the middle of a pattern. An interesting questions is how complicated can the patterns be? Before attempting to answer that we will see some still more complex patterns.

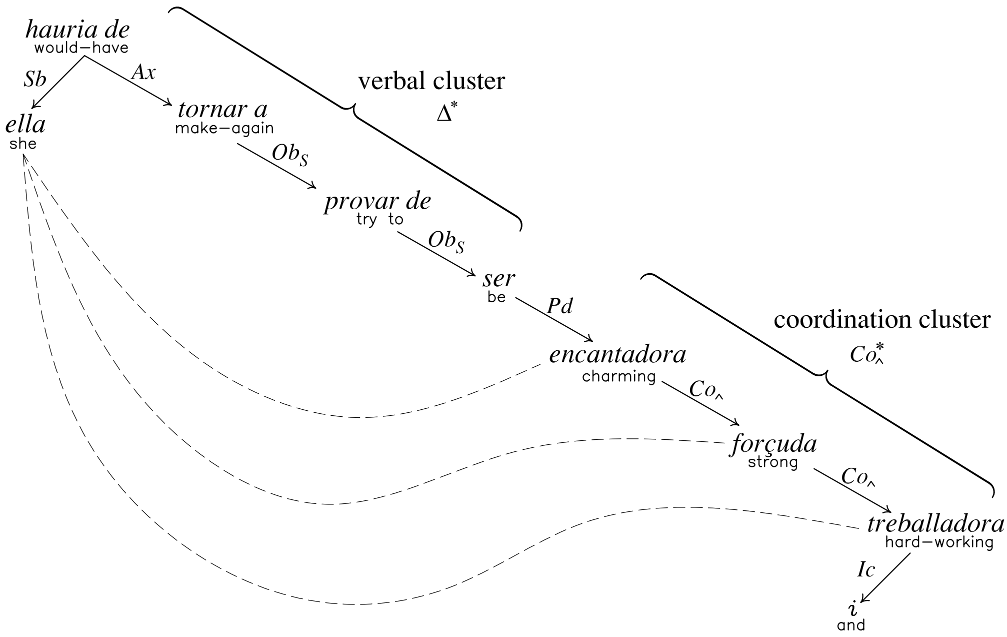


Figure 10.7: combination of long distance agreement in predicates and coordinated adjectives.

10.5.4 Long Distance Agreement in Pied-Piping in Romance Languages

We saw in §4.1.3 the agreement of pied-piping in Romance language with sentences like:

(35) Aquesta reina, el pare del pare ... del pare de la qual fou
This queen the father of-the father ... of-the father of the who was
destronat, regnà justament.
dethroned, reigned rightly
'This queen the father of the father ... of the father of whom was dethroned reigned rightly.'

The underlying words in the example and the following sentences indicates the agreed parts. Pied-piping is in terms of clustering simply a relativization of a noun complement Nc_R which is governed by a noun complement cluster Nc^*, see Fig. 10.8(a). The sentence invites initially a pattern such as:

$$(1, Dt)\cdot(1, Nc_R)\cdot(1 \oplus Nc^*)\cdot(1, Sb)\cdot(1, Ad_S)\cdot(Sb)_2.$$

But this phenomenon can occur anywhere, for example in the object, Fig. 10.8(b):

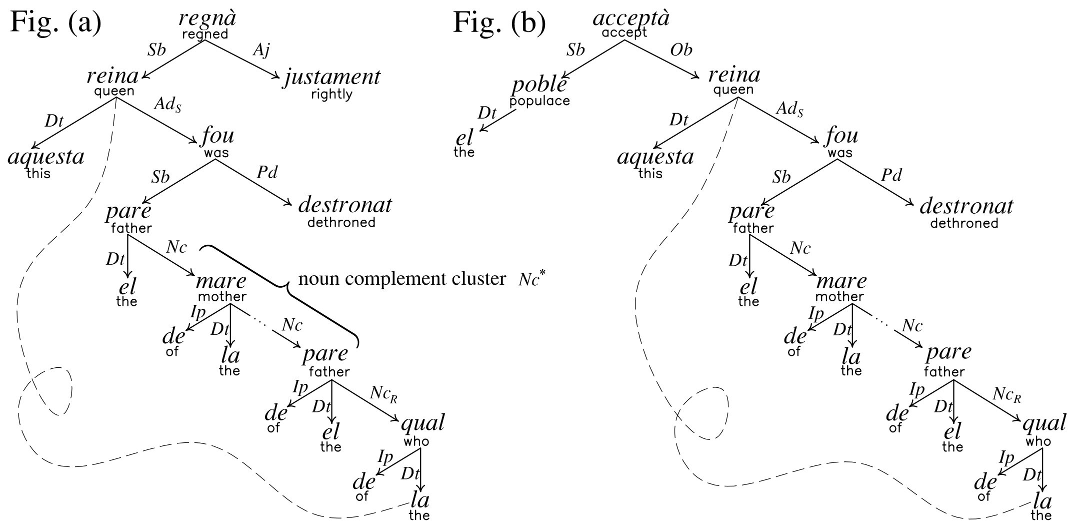


Figure 10.8: agreement in pied-piping on the subject (a) and on the object (b).

(36) El poble acceptà aquesta reina el pare del pare ... del pare
The populace accepted this queen the father of-the father ... of-the father
de la qual fou destronat.
of the who was dethroned
'The populace accepted this queen the father of the father ... of the father of whom was dethroned.'

Then the pattern turns to $(1 \oplus Dt)\cdot(1, Nc_R)\cdot(1 \oplus Nc^*)\cdot(1, Sb)\cdot(1, Ad_S)\cdot(Ob)_2$. In fact we can embed it in other functions like indirect object, or deeper positions like the sentential subordinated object Ob_S of a sentential subordinated object which leads us to infer a more general pattern, substituting $(Ob)_2$ or the above $(Sb)_2$ by $(\zeta)_2^*$:

$$(1, Dt)\cdot(1, Nc_R)\cdot(1 \oplus Nc^*)\cdot(1, Sb)\cdot(1, Ad_S)\cdot(\zeta)_2^* \in \mathbf{kk}\mathbb{P}\mathbf{kk}\mathbb{H}.$$

However we can we can interpose a verbal cluster Δ_S^*. See the following sentence and see Fig. 10.9.

(37) Aquesta reina, el pare de la mare de la qual Pere II podria haver
This queen the father of the mother of the who Pere II could have
destronat, regnà justament.
dethroned, reigned rightly
'This queen the father of the mother of the whom Pere II could have dethroned, reigned rightly.'

Now the pattern is (we have underlined the new submonoid interposed):

$$(1, Dt)\cdot(1, Nc_R)\cdot(1 \oplus Nc^*)\cdot(1, Ob)\cdot\underline{(1 \oplus \Delta_S^*)}\cdot(1, Ad_S)\cdot(\zeta)_2^* \in \mathbf{kk}\mathbb{P}\mathbf{k}\underline{\mathbb{P}}\mathbf{k}\mathbb{H}.$$

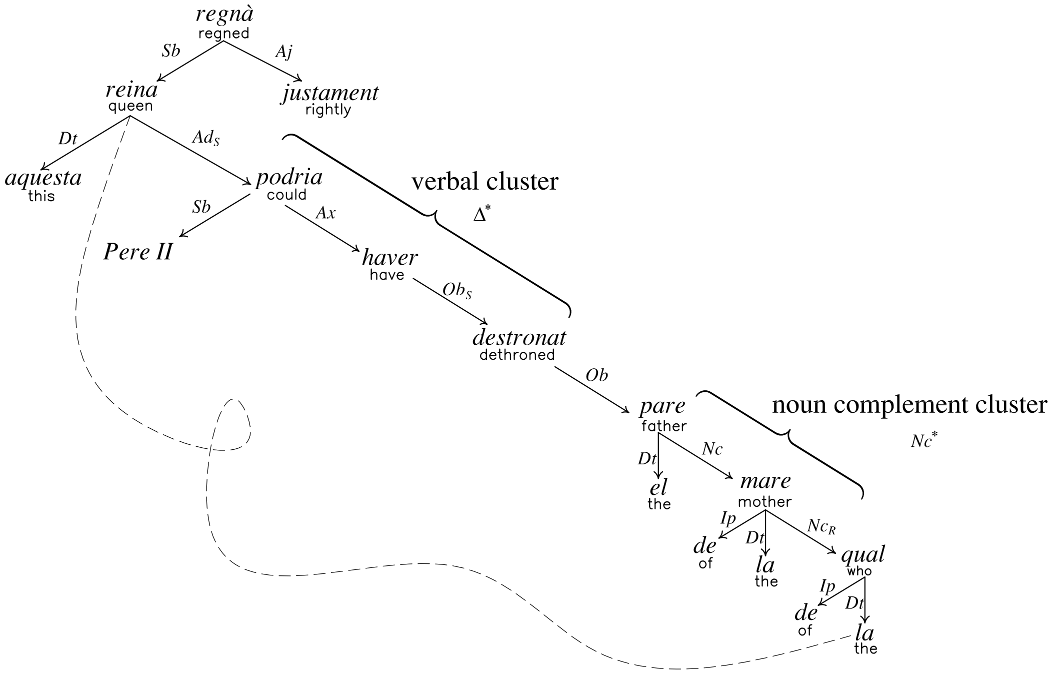


Figure 10.9: agreement in pied-piping which shows a combination of noun complement cluster and verbal cluster.

Even the *Ob* in the middle can be any other primitive verbal function in Δ, so a general pattern for pied-piping is:

$$(1, Dt)\cdot(1, Nc_R)\cdot(1 \oplus Nc^*)\cdot(1, \Delta)\cdot(1 \oplus \Delta_S^*)\cdot(1, Ad_S)\cdot(\zeta)_2^* \in \mathbf{kk}\mathbb{P}\mathbf{k}\mathbb{P}\mathbf{k}\mathbb{H}.$$

Nevertheless this could be not definitive. We have combined a Noun complement cluster with a verbal cluster. We can try to add a coordination cluster with a sentence like (see Fig. 10.10):

(38) ?Aquesta reina, el pare de la mare de la qual, la mare de la qual
This queen the father of the mother of the who, the mother of the who
i el germà de la qual Pere II podria haver destronat, regnà justament.
and the brother of the who Pere II could have dethroned, reigned rightly

?'This queen the father of the mother of the whom, the mother of the whom and the brother of the whom Pere II could have dethroned, reigned rightly.'

This leads us to interpose a new submonoid for the coordination cluster Co^*:

$$(1, Dt)\cdot(1, Nc_R)\cdot(1 \oplus Nc^*)\cdot\underline{(1 \oplus Co^*)}\cdot(1, \Delta)\cdot(1 \oplus \Delta_S^*)\cdot(1, Ad_S)\cdot(\zeta)_2^* \in \mathbf{kk}\mathbb{P}\underline{\mathbb{P}}\mathbf{k}\mathbb{P}\mathbf{k}\mathbb{H}.$$

Finally, where is the limit? Can we interpose more and more clusters? We think that this is not the case. The validity of the above sentence (38) is not clear at all, and nobody

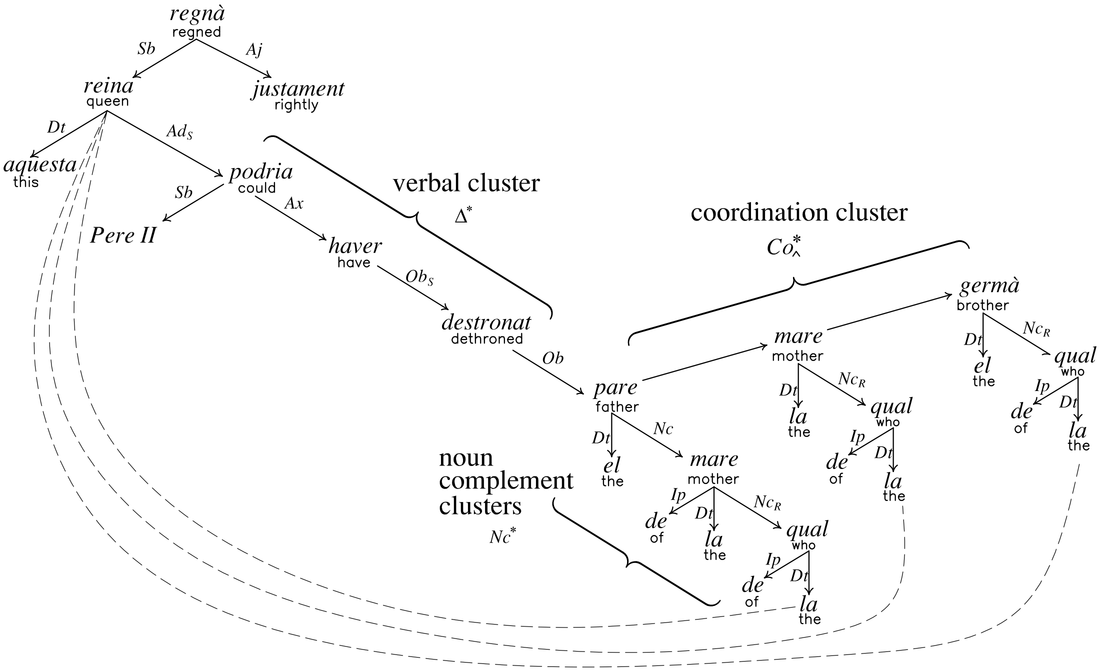


Figure 10.10: eccentric combination of noun complement cluster, coordination cluster, and verbal cluster.

makes such a construction in a conversation, not even in a formal text. We have detected in Romance languages and English three clusters which can be combined (although other languages could enjoy more kinds): noun complement, verbal and coordination cluster. One can try to combine indefinitely these clusters however some combinations are grammatically impossible or at least unproductive: for example a verbal cluster with a verbal cluster would result simply in a verbal cluster. In addition other forms of combining clusters provokes ellipses in the relative pronoun which removes the agreement and the pattern becomes unnecessary.[8] For these reasons we believe that there is a limit to combining clusters.

10.6 Non-Semi-Linear Constructions

We conclude the chapter with three constructions sharing the particularity that they question semi-linearity as a linguistic invariant. Clark and Yoshinaka (2012) collect the three phenomena and give formalizations in the framework of parallel multiple context-free grammars.

[8] For this reason we marked with $^?$ the acceptability of the sentence (38).

10.6.1 Chinese Numbers

The number-name system in the Mandarin dialect of Chinese works for large numbers in the following way.[9] The name for 10^{12} is *zhao*, and the word for 5 is *wu*. In order to simplify we do not consider the other basic numerals 1,2,3, . . .

The sequence *wu zhao zhao wu zhao* is well-formed although *wu zhao wu zhao zhao* is not. The rule is that the number of consecutives *zhao*'s must strictly decrease from left to right. So the language of Chinese numbers that we can construct in Mandarin with the words *zhao* and *wu* is:[10]

$$L_{\text{Chin}} = \{wu\ zhao^{k_1}\ wu\, zhao^{k_2} \cdots wu\ zhao^{k_p} \mid k_1 > k_2 > \cdots > k_p > 0\}.$$

According to Radzinski (1991), Chinese numbers seem to call the Joshi conditions into question. In particular Radzinski (1991) shows that the Chinese numbers language is not a Linear Context-Free Rewriting language.[11] For this reason Boullier (1999) introduced *Range Concatenation Grammars* which can generate L_{Chin}.

In an algebraic dependency grammar we can describe this construction through a manifold which contains syntagmata like in Fig. 10.11(a) and a projective linearization, see Fig. 10.11(b). We need a pair of functions $\zeta = \{\alpha, \beta\}$ and some basic rules stating that α takes the word *wu* and β takes the word *zhao*; in addition we need that the manifold be non-elliptic. The main constraint is informally the following:

$$\textit{If } \beta^{n+1}\alpha^m \textit{ is null then } \beta^n\alpha^{m+1} \textit{ is null,}$$

for any $n, m \in \mathbb{N}_+$. So formally we take the valuation $B(x, y) = (x \approx 0) \to (y \approx 0)$ which is applied along the pattern:

$$\{(\beta^{n+1}\alpha^m, \beta^n\alpha^{m+1}) \mid n, m \in \mathbb{N}_+\} = (\beta, 1)\cdot(\beta, \beta)^*\cdot(\alpha, \alpha)^*\cdot(1, \alpha) \in \mathbf{k}\mathbb{G}^2\mathbf{k}.$$

Notice that if we fix a natural m the rule states that the m-th column of β's in the picture Fig 10.11 is strictly longer than the $(m+1)$-th column. So the number of *zhao*'s necessarily decreases. In order to linearize the manifold we simply take the arrangement $1 + \zeta^*\beta + \zeta^*\alpha$.

10.6.2 *Suffixaufnahme* in Old Georgian

Old Georgian is a languages which exhibit the phenomenon called *Suffixaufnahme* (literally: taking up of suffixes, or *suffix stacking*). Following Michaelis and Kracht (1997), from whom we take the example:[12]

[9] We will follow definitions and the notation from (Boullier, 1999) and (Radzinski, 1991).

[10] According to Kanazawa (2017) the last k_p must satisfies $k_p \geq 0$ because strings such as *wu zhao* are perfectly grammatical. However we follow the original article (Radzinski, 1991); the manifold which will be introduced later can be modified in order to incorporate Kanazawa's refinement.

[11] As we commented, §9.4, these kinds of languages are an attempt to account for mild context sensitivity, (Vijay-Shanker et al., 1987; Weir, 1988).

[12] See also (Clark and Yoshinaka, 2012).

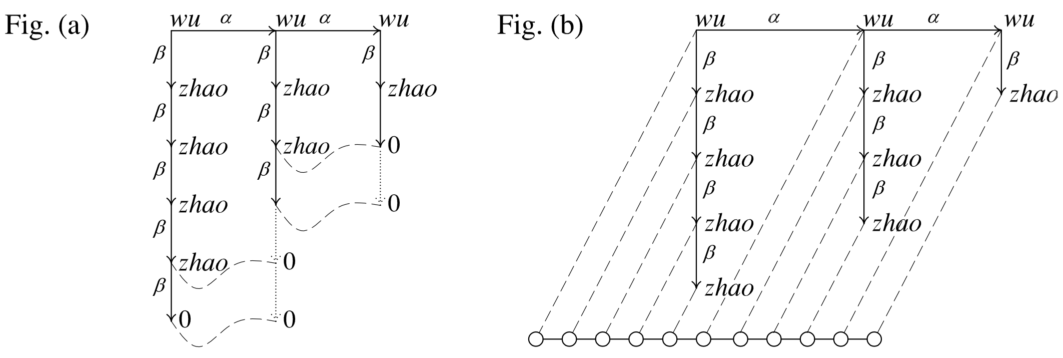


Fig. (a) Fig. (b)

Figure 10.11: example of a syntagma for the Chinese numbers language.

(39) tkuenda micemul ars cnob-ad saidumlo-j igi sasupevel-isa
to=you given is knowing-Adv mystery-Nom Art=Nom kingdom-Gen
m-is γmrt-isa-jsa-j
Art-Gen god-Gen-Gen-Nom
'Unto You it is given to know the mystery of the kingdom of God.'

It is observed that the infixes, *isa* or *jsa*, are repeated in *γmrt-isa-jsa-j* in order to express a doubly nested genitive. It has been suggested that if English had *suffixaufnahme* one could say noun phrases like:

(40) a. John's cat,
b. John's's father's cat,
c. John's's's friend's's father's cat,
d. John's's's's friend's's's father's's cat's tail (. . .)

Notice that the number of genitive *'s* for the *n*-th example grows with the triangular numbers $\frac{n(n+1)}{2} \in O(n^2)$. A major problem for the grammatical acceptance of such construction is that there are no native Old Georgian speakers with whom to test controversial examples.

Assuming grammaticality, these noun phrases can be schematically represented by the formal language:

$$\begin{aligned} L_{\text{OldG}} &= \{a, aab, aababb, aababbabbb, aababbabbbabbbb, \ldots\} \\ &= \{\prod_{k=0}^{n} ab^k \mid n \geq 0\}, \end{aligned}$$

where the *a*'s are nouns and the *b*'s are genitive affixes. We show a manifold and a linearization for this language. We take $\zeta = \{\alpha, \beta\}$ and $\Sigma = \{a, b\}$. The function α always takes the letter *a*, and β, letter *b*. Syntagmata are non-elliptic. The main constraint is stated informally as follows:

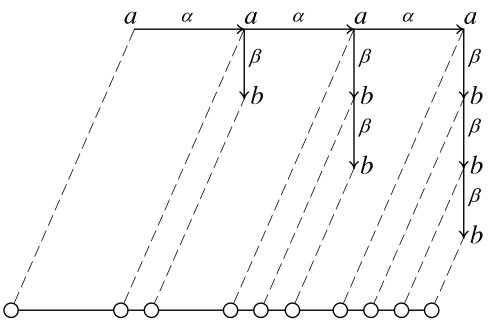


Figure 10.12: example of a syntagma for the *Suffixaufnahme* in Old Georgian.

If a column takes n+1 *b's then the preceding column takes exactly n b's;*

more formally, for any $n, m \in \mathbb{N}_+$:

If $\alpha^{n+2}\beta^{m+1}$ *is null and* $\alpha^{n+1}\beta^{m+1}$ *is not null (i.e. the* $(m+1)$*-th column takes* $n+1$ *a's) then* $\alpha^{n+1}\beta^{m}$ *is null and* $\alpha^{n}\beta^{m}$ *is not null (i.e. the m-th column takes n a's).*

So, we take the valuation:

$$B(x, y, z, t) = \big((x{\approx}0) \wedge (y{\not\approx}0)\big) \rightarrow \big((z{\approx}0) \wedge (t{\not\approx}0)\big),$$

together with the pattern:

$$(\alpha^2, \alpha, \alpha, 1)(\alpha)_4^*(\beta)_4^*(\beta, \beta, 1, 1) \in \mathbf{k}^2\mathbb{G}^2\mathbf{k}.$$

We have to add a manifold with a constant pattern to makes sure that the *a*'s begin right in the second column, consisting in a rule saying: *if* α *is not null then* $\beta\alpha$ *is not null*. The linearization is given by the single projective arrangement $1 + \zeta^*\alpha + \zeta^*\beta$. Fig. 10.12 shows a syntagma and its linearization for the string *aababbabbb*.

10.6.3 Recursive Copy in Yoruba

The following relative clauses belong to Yoruba, a Nigerian language:

(41)	rira	NP ti	Ade ra	NP
	buying	NP that	Ade buy	NP
	'the fact that Ade bought NP'			

where the noun phrases NP must be identical. As Kobele (2006) and Clark and Yoshinaka (2012) point out, copying constructions occur in several West African languages, though Yoruba seems to have the most complex system.

Since the noun phrase must be repeated, this suggests a recursive construction of the form: $X ::= v_1 + X + ti + Ade + v_2 + X$, where v_1 and v_2 are verbs and where the substitution is applied simultaneously to both occurrences of X. This generates a set of noun phrases with an exponential growth. According to Clark and Yoshinaka (2012), by intersecting Yoruba with a suitable regular language the resulting set can be reduced by an homomorphism to the exponential language $L_{\text{expo}} = \{a^{2^n} \mid n \geq 1\}$, whereby Yoruba is not described by any MCFG. This motivates Clark and Yoshinaka (2012) to introduce PMCFG.

We already constructed the exponential language in Example 32 through the manifold:[13]

$$\mathbf{Nell} \cap \mathbf{Synt}\left(\begin{array}{c} \approx \\ (\zeta)^*_2(\alpha,\beta)(\zeta)^*_2 \end{array}\right) \in \mathbf{Man}(\mathbb{H}\mathbf{k}\mathbb{H}).$$

A weak imitation to this construction in a Romance language could be sentences such as (in Catalan):

(42) Quan dic que vull gelat, és que vull gelat!
when I-say that I-want ice-cream, it-is that I-want ice-cream
'When I say I want ice cream, I want ice cream!'

Usually when a hearer hears the fragment *when I say X...* with an exigent imperative emphasis, they expect to hear again X finishing the sentence. The construction is not entirely rigid because it appears that one can say variants in the second X, even a completely different clause, which makes it a non-well defined phenomenon. But neglecting this, one could say:

(43) Quan dic que (quan dic que vull gelat, és que vull gelat),
when I-say that (when I-say that I-want ice-cream, it-is that I-want ice-cream)
és que (quan dic que vull gelat, és que vull gelat).
it-is that (when I-say that I-want ice-cream, it-is that I-want ice-cream)
'When I say (when I say I want ice cream I want ice cream) (when I say I want ice cream I want ice cream).'

and so forth. Although one cannot isolate this fragment of the language by intersecting with a regular language because of the variants in the second X, the examples suggest that a copy mechanism could intervene in the natural language.

This construction, *quan dic que X és que X*, can be transfered to manifolds by a rule stating that when certain particles occur at the top of the tree (*quan dic que ...*) then the subtree $\zeta^* \cdot Ob_S$ and the subtree $\zeta^* \cdot Pd_S \cdot Co$ (where Co is a connecter function) must be identical; see Fig. 10.13(a). More formally $S(\varphi \cdot Ob_S) = S(\varphi \cdot Pd_S \cdot Co)$ for any $\varphi \in \zeta^*$. This yields the pattern $(\zeta)^*_2(Ob_S, Pd_S \cdot Co)$ with type $\mathbb{H}\mathbf{k}^2$. But this construction can be embedded as a clause in itself (for example *El Pere creu que quan dic X és que X*, in English *Peter thinks that when I say X, X*; see Fig. 10.13(b)), whereby the

[13] More exactly, the manifold above yields the language $\{a^{2^n-1} \mid n \geq 0\}$, but it can be adapted to obtain the language $\{a^{2^n} \mid n \geq 0\}$ with the same types of patterns.

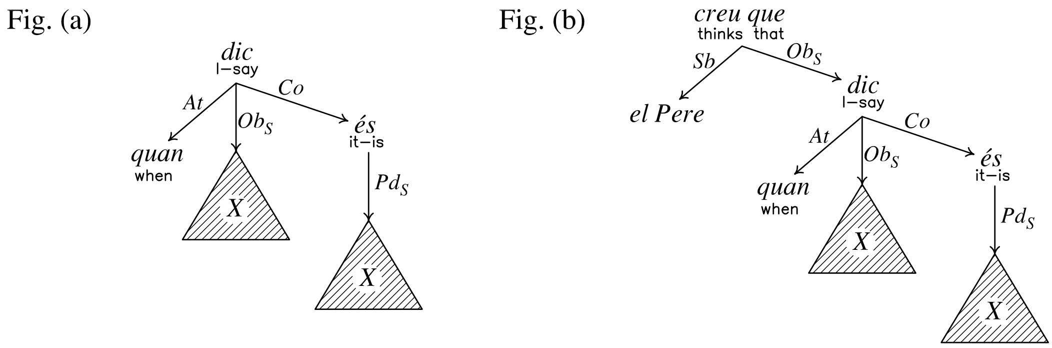


Figure 10.13: examples of copied clauses in Catalan.

pattern becomes $(\zeta)^*_2(Ob_S, Pd_S\cdot Co)(\zeta)^*_2 \in \mathbb{H}\mathbf{k}^2\mathbb{H}$, which is a case similar to that of the exponential language.

11

Natural Language Constructions: Linearizations

We continue examining natural language constructions. In this chapter we focus on arrangements and systems of arrangements. The philosophy of linearizations in algebraic dependency grammar consists in decomposing the syntagma in patterns and writing them in an adequate order. That is, no restructuring movement or transformation is required in the linearization process.

Since we need to show a large number of arrangements, first of all we introduce a few notational devices in order to describe several arrangements in one go.

We start with the canonical word-order of English and Romance languages, which shows us how to organize a system of arrangements for natural language. Then we show how to use isotopes in order to achieve some constructions which are projective: *subordination, coordination*, and *shifting*. In continuation we will examine the typical difficult non-projective cases: *wh-fronting, floating quantifiers, extraposition, topicalization, discontinuities in classical Latin, serial dependencies in Germanic languages, scrambling in German*, and finally *respectively construction.*

11.1 Linearization in Algebraic Dependency Grammar

Typical cases of discontinuity are wh-fronting, floating quantifier, extraposition, topicalization, cross-serial dependencies or scrambling. We related in §5.3.2 that the question of non-projectivity can be tackled by two means: either we recast the analysis of dependency trees, or we reconsider the constraint of projectivity.

With respect to the first means, the strategy consists essentially in that the particle, or constituent, or the fragment in the dependency tree, which is problematic climbs up to the root, which facilitates a projective linearization. Several such proposals are made in the literature: *climbing*, (Duchier and Debusmann, 2001); *emancipation*, (Gerdes and Kahane, 2001); *raising*, (Hudson, 2000); *lifting*, (Bröker, 2000); *adjunction*, (Eroms and Heringer, 2003); and *rising*, (Groß and Osborne, 2009).

Nevertheless restructuring the syntactic tree, although simple and effective for some cases, comports some non-desirable consequences; consider (Groß and Osborne, 2009) which analyzes several cases by this means.[1] The issue is that the tricky constituent must be placed in a non-natural position. This implies that either we are forcing an artificial syntactic analysis or else we are assuming covertly some kind of movement. In the first case we would have to incorporate a mechanism capable to describe these new artificial analysis. In the second case, this movement must be incorporated in the architecture of the model which complicates it with the interposition of an extra layer or module between the syntactic structure and the word-order.

We would like to address linearization phenomena exclusively through arrangements, which situates us in the second means.[2] This, in the framework of our formalism, consists in choosing carefully a decomposition of the syntagma in patterns and then writing them in the adequate order. The interrelation of the several arrangements (the system of arrangements) will describe the general word-order of the language.

11.2 Abbreviation Operators

Since a natural language can exhibit a large number of word-orders, which implies a large number of arrangements, we need some abbreviations. We emphasize that no operator which we are going to introduce confers any additional computational power to the linearizations: the number of arrangements continues being finite. They must be viewed just as abbreviatory devices. We introduce three operators (in the definitions we use the multiplicative notation for reasons of space).

Set extension operator: We extend the product in Σ^* to sets. Given $x \in \Sigma^*$ and $X, Y \in \wp(\Sigma^*)$ we write:

$$\begin{aligned} x\cdot Y &= \{xy \mid y \in Y\};\\ Y\cdot x &= \{yx \mid y \in Y\};\\ X\cdot Y &= \{xy \mid x \in X, y \in Y\}. \end{aligned}$$

Floating operator:

$$\begin{aligned} \bowtie\colon \Sigma \times \Sigma^* &\longrightarrow \wp(\Sigma^*)\\ (x, a_1 \cdots a_n) &\longmapsto \{xa_1 \cdots a_n,\ a_1xa_2 \cdots a_n,\ \dots,\ a_1 \cdots a_nx\}. \end{aligned}$$

[1] Even with this however, there are some cases which are not well explained. Consider for example the case of floating quantifiers, as it is related in (Osborne, 2013).

[2] We also investigate the first means in (Cardó, 2016) which led us to observe an underlying symmetry between some cases of projectivity and non-projectivity. Also Groß and Osborne (2009) address several cases of discontinuity by the first means.

This operator can be extended as:

$$\begin{aligned} \bowtie\colon \Sigma \times \wp(\Sigma^*) &\longrightarrow \wp(\Sigma^*) \\ (x, Y) &\longmapsto x \bowtie Y = \bigcup_{y \in Y} x \bowtie y. \end{aligned}$$

Scrambling operator:

$$\begin{aligned} \mathbf{scr} : \Sigma^* &\longrightarrow \wp(\Sigma^*) \\ a_1 \cdots a_n &\longmapsto \{a_{\sigma(1)} \cdots a_{\sigma(n)} \mid \sigma \text{ is a permutation}\}. \end{aligned}$$

It extends as:

$$\begin{aligned} \mathbf{scr} : \wp(\Sigma^*) &\longrightarrow \wp(\Sigma^*) \\ X &\longmapsto \bigcup_{x \in X} \mathbf{scr}(x). \end{aligned}$$

The set extension operator is useful to write several arrangements in one go. For example the pair of arrangements: $\zeta^*\alpha + 1 + \zeta^*\gamma, \quad \zeta^*\beta + 1 + \zeta^*\gamma$ can be written as: $\zeta^*\Theta + 1 + \zeta^*\gamma$, where $\Theta = \{\alpha, \beta\}$. Regarding the second operator we have for example:

$$\begin{aligned} x \bowtie a &= \{xa, ax\}, \\ x \bowtie ab &= \{xab, axb, abx\}, \\ x \bowtie abc &= \{xabc, axbc, abxc, abcx\}. \\ &\vdots \end{aligned}$$

We will abbreviate:

$$x_1 \bowtie x_2 \bowtie \cdots x_n \bowtie y = x_1 \bowtie (x_2 \bowtie (\cdots (x_n \bowtie y) \cdots))$$

An interesting property is $x \bowtie y \bowtie z = y \bowtie x \bowtie z$. Using this we have that for any permutation σ of the subscripts we have that:

$$x_1 \bowtie x_2 \bowtie \cdots x_n \bowtie y = x_{\sigma(1)} \bowtie x_{\sigma(2)} \bowtie \cdots x_{\sigma(n)} \bowtie y.$$

The third operator will be useful to describe the scrambling construction in German and others. We have for example:

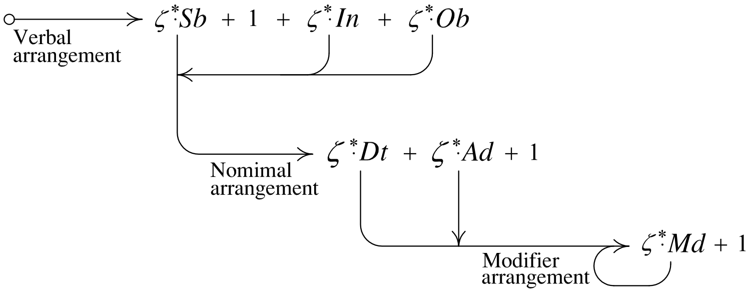
$$\begin{aligned} \mathbf{scr}(a) &= \{a\}, \\ \mathbf{scr}(ab) &= \{ab, ba\}, \\ \mathbf{scr}(abc) &= \{abc, acb, bac, bca, cab, cba\}, \\ &\vdots \end{aligned}$$

Some obvious properties are: $\mathbf{scr}(\mathbf{scr}(x)) = \mathbf{scr}(x)$; $\mathbf{scr}(ab) = a \bowtie b$, where $a, b \in \Sigma$; $\mathbf{scr}(x \bowtie y) = \mathbf{scr}(xy)$; $|\mathbf{scr}(x)| = |x|!$.

11.3 Canonical Word-Orders

By canonical word-orders we mean a system of arrangements describing the order of the most basic functions: *Sb*, *Ob*, *In*, *Dt*, *Ad*, *Md*. For many languages canonical word-orders can be described using projective arrangements. The issue of projectivity has been addressed in previous chapters and we concluded that these linearizations use patterns in $\mathbb{H}\mathbf{k}$. We proved that $\mathbf{Lin}(\mathbb{H}\mathbf{k}) = \mathbf{Lin}(\mathbb{M}\mathbf{k}) = \mathbf{Lin}(\mathbb{G}\mathbf{k})$; see Corollary 7.17. Since we are going to use in general patterns which are not always the full ζ^*, we will use the classificatory monoid $\{\mathbf{k}, \mathbb{G}\}^*$.

The system of arrangements in English for these functions is:



We call them respectively *verbal arrangement*, *nominal arrangement*, and *modifier arrangement*. We already saw an example in §22.

Canonically Romance languages position the indirect object after the direct object and the adjective after the noun. That is $\zeta^*{\cdot}Sb + 1 + \zeta^*{\cdot}Ob + \zeta^*{\cdot}In$ and $\zeta^*{\cdot}Dt + 1 + \zeta^*{\cdot}Ad$.

The above system of arrangements displays only one cluster which is represented as a loop in the modifier. It permits sentences of unbounded length by adding modifiers: *John caught a very very . . . very big frog*.

11.4 Subordination and Coordination

Let us add a subordinate isotope Ad_S which permits complementation of a noun through a subordinate clause. Consider the sentence:

(44) John met the man who caught frogs.

The subordinate clause is complementing the noun, so *man* dominates the subordinate clause. More specifically *man* governs the verb of the clause *caught* through the subordinate isotope Ad_S. In addition there is a relative pronoun isotope function which introduces the pronoun *who* which is the subject of the subordinate clause, so we need the isotope Sb_R. See the first figure of Fig. 11.1 and consider the system of arrangements:

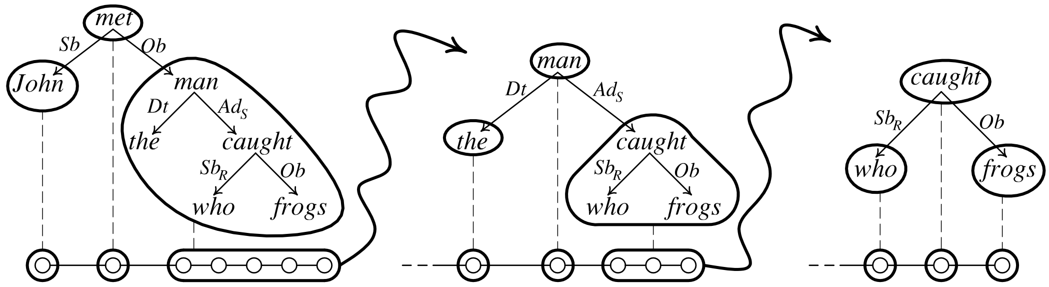


Figure 11.1: linearization of a syntagma with a subordinate clause.

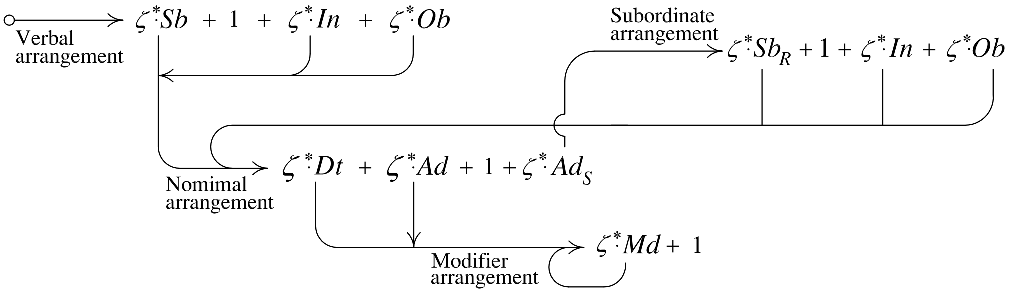
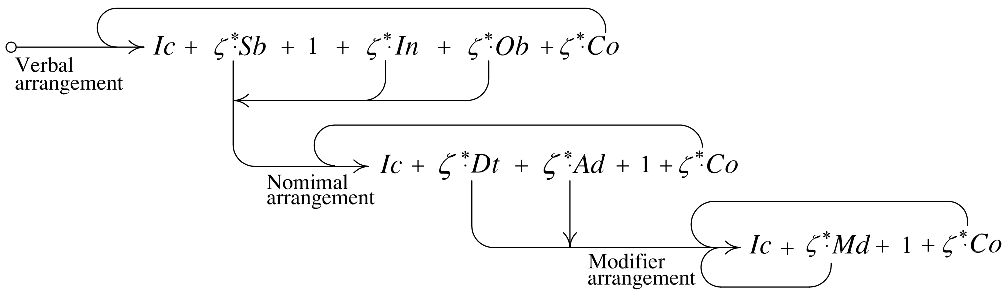


Fig. 11.1 shows the process of linearization: first we apply the verbal arrangement for the main clause; second, the subordinate arrangement and finally, the nominal arrangement. We dispose of a new loop which permits linearization of sentences like *John met the man who met the man who met the man . . . who caught frogs*. Similarly we can add to the system the other possible subordinate functions: Sb_S, Ob_S, ...

Consider now the coordination function which creates loops in each arrangement. We are going to write simply *Co*, instead of $Co_{and}, Co_{or}, \ldots$ in order to simplify; the several specific coordination functions can be implemented similarly. We take the system of arrangements for English:



This shows how coordination is allowed at several levels.

By way of example consider the sentence with *gapping*:

(45) John caught frogs and Mary, rabbits.

The process of linearization can be followed in Fig. 11.2.

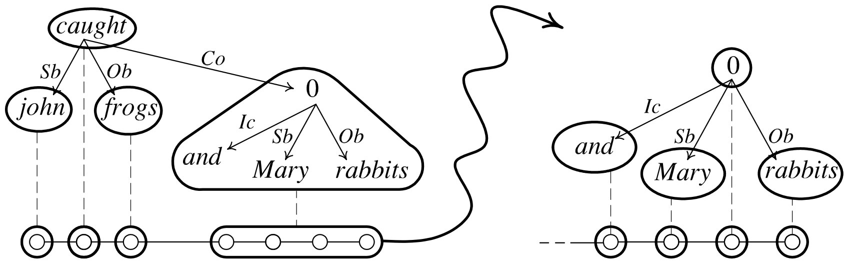
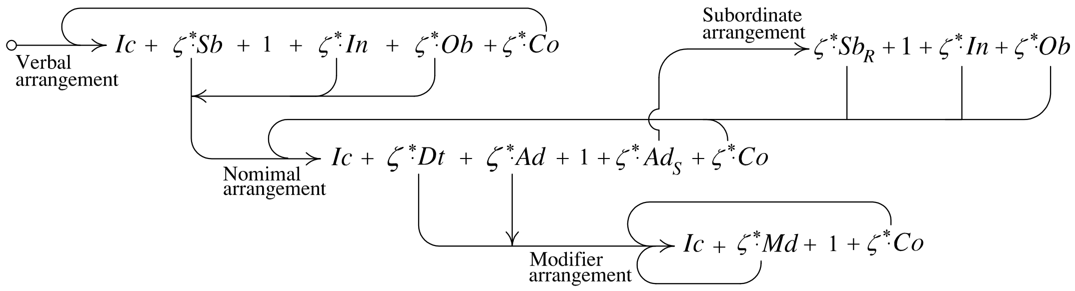


Figure 11.2: linearization of a syntagma with a coordinate clause.

In order to describe a grammar we should merge both systems of arrangements, subordination + coordination, which yields:


Verbal arrangement
$Ic + \zeta^{*}Sb + 1 + \zeta^{*}In + \zeta^{*}Ob + \zeta^{*}Co$
Subordinate arrangement
$\zeta^{*}Sb_R + 1 + \zeta^{*}In + \zeta^{*}Ob$
Nomimal arrangement
$Ic + \zeta^{*}Dt + \zeta^{*}Ad + 1 + \zeta^{*}Ad_S + \zeta^{*}Co$
Modifier arrangement
$Ic + \zeta^{*}Md + 1 + \zeta^{*}Co$

However, we will illustrate the following constructions separately. We start every time from the canonical system and we add some new arrangements (or groups of arrangements) or we susbtitute some arrangement for another more complex one (usually the verbal arrangement which enjoys a richer phenomenology). So every case is studied separately. In order to formulate a complete grammar we would have to merge all the cases.

11.5 Shifting

A main virtue of isotopes is in achieving a more specific control of the linearization, since the addition of a new function permits a new position in the arrangement.

11.5.1 Shifting of Object and Indirect Object in English

The canonical verbal arrangement explains the sentence:

(46) The President awarded John a gold medal.

However when the indirect object is introduced by a preposition or it is quite long it shifts with the object, as in for example:

(47) He distributed chocolates to all the boys.

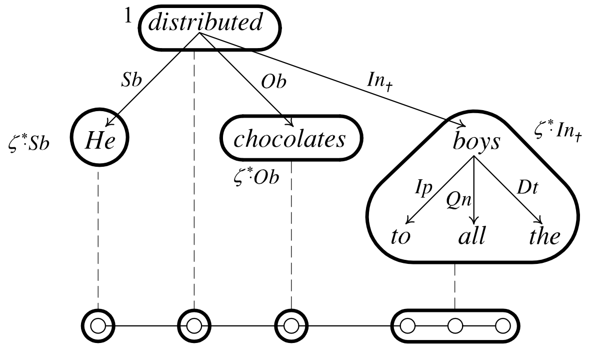


Figure 11.3: shifting of indirect object in English.

In order to distinguish both cases we introduce the isotope $In_\dagger$. This function needs to be introduced by the preposition *to*. That is for any syntagama S we have that for any $x \in \zeta^*$ if $S(In_\dagger \cdot x) \neq 0$ then $S(Ip \cdot In_\dagger \cdot x) \in \{to\}$. This can be implemented easily in the manifold. We need another rule to ensure that *In* and $In_\dagger$ are orthogonal, that is, they do not appear simultaneously. See Fig. 11.3 for the analyses. Finally we must consider an alternative verbal arrangement for this alternative indirect object:

$$\zeta^* \cdot Sb + 1 + \zeta^* \cdot Ob + \zeta^* \cdot In_\dagger .$$

However, since the functions $In, In_\dagger$ are orthogonal, we can merge the last arrangement and the original verbal arrangement:

$$\zeta^* \cdot Sb + 1 + \zeta^* \cdot In + \zeta^* \cdot Ob + \zeta^* \cdot In_\dagger .$$

11.5.2 Shifting of the Subject in Catalan

The canonical word-order in Catalan is SVO, however some specific verbs allows other positions of the subject. For example:

(48) a. La finestra ha caigut.
The window has fallen
'The window has fallen.'

b. Ha caigut la finestra.
Has fallen the window
'The window has fallen.'

We consider the lexical category: **PostverbalSubjectVerbs** = {*arribar, sortir, marxar, caure, morir, . . .* }, in English: to arrive, to leave, to march, to fail, to die, ... All these kind of verbs allow the subject in postverbal position. We can introduce an isotope of the subject $Sb_{\dagger}$ which is orthogonal to the primitive subject. In addition $Sb_{\dagger}$ only can be non-null when the main verb is in **PostverbalSubjectVerbs**, which can be easily achieved in a manifold. Then we take the pair of arrangements:

$$\zeta^{*}\cdot Sb + 1 + \zeta^{*}\cdot Sb_{\dagger} + \zeta^{*}\cdot Ob + \zeta^{*}\cdot In.$$

$$\zeta^{*}\cdot Sb_{\dagger} + \zeta^{*}\cdot Sb + 1 + \zeta^{*}\cdot Ob + \zeta^{*}\cdot In.$$

We do not need to worry about the interrelation between the shifting subject and the object and indirect object, because the verbs from the list above turn to be intransitive and consequently these functions are empty (cf. Solà et al. (2010) §1.3.3.2). We can implement shifting adjectives in Romance languages similarly. Regarding the type of the patterns we are still using patterns of the form $\mathbb{H}\mathbf{k} \subset \mathbb{G}\mathbf{k}$.

11.6 Some Romance Constructions[3]

11.6.1 Verb Clusters in Romance Languages

The term *verb cluster* refers to a chain of verbs which could be auxiliary, modal, subordinate verbs, and so forth. These tend to form a syntactic unit which authors try to capture by some means.[4] Here we use patterns to capture them. It is necessary to distinguish two kinds of verb cluster. On the one hand, one which only contains the auxiliary function Ax (as for example *hauries hagut d'agafar*, in English *you had to have taken*),[5] and, on the other hand, one which can contain auxiliary functions and subordinate functions $\Delta_S = \{Ax, Sb_S, Ob_S, \ldots\}$ (as for example *hauries volgut intentar començar*, in English *you had to want to try to start*).

In Romance languages verbal clusters behave more monolithically than in English, whereby we are going to differentiate both cases in this and the following section. In addition the condition of unity only arises when we consider the cluster in relation with some constructions. As an evidence of this last consider a sentence like:

(49)	El	Joan	ha	pogut	donar	diners	a	la	Maria.
	The	John	have	been-able	give	money	to	the	Mary

‘John has been able to give Mary some money’.

Even though this sentence can be linearized projectively as in Fig. 11.4(a) (we have just to take the projective arrangement $\zeta^{*}\cdot Sb + 1 + \zeta^{*}\cdot Ax$), this turns out to be inadequate in relation to some other constructions such as wh-fronting, adjuncts or floating quantifiers. In fact we must to consider the verbal arrangement:

$$\zeta^{*}\cdot Sb + Ax^{*} + \zeta^{*}\cdot Ob\cdot Ax^{*} + \zeta^{*}\cdot In\cdot Ax^{*}.$$

[3] For the Romances language we have taken examples in Catalan. This shares features with Spanish, French or Italian, for example.

[4] See the issue of substructures in §6.1.

[5] For simplicity we will understand modal verbs as auxiliary verbs.

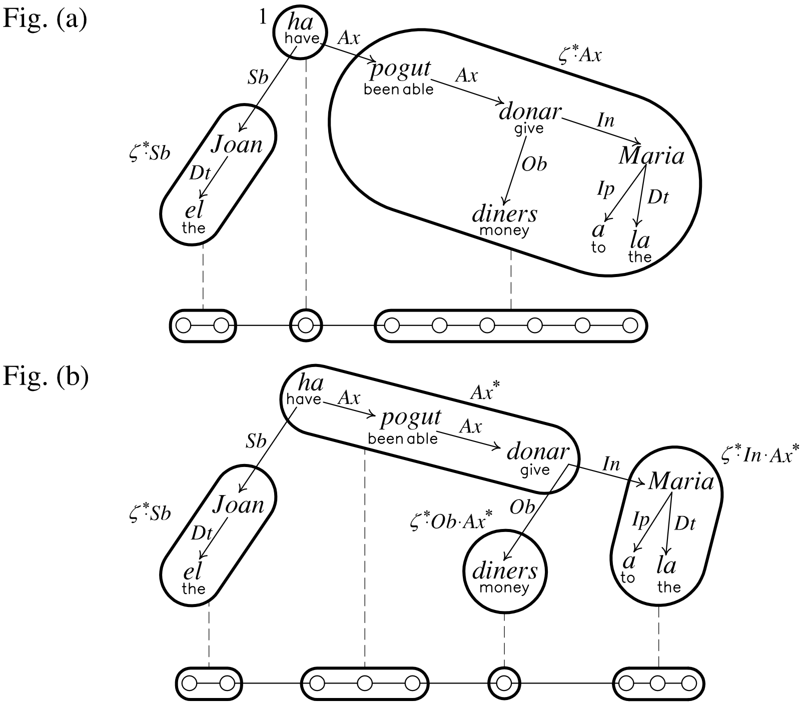
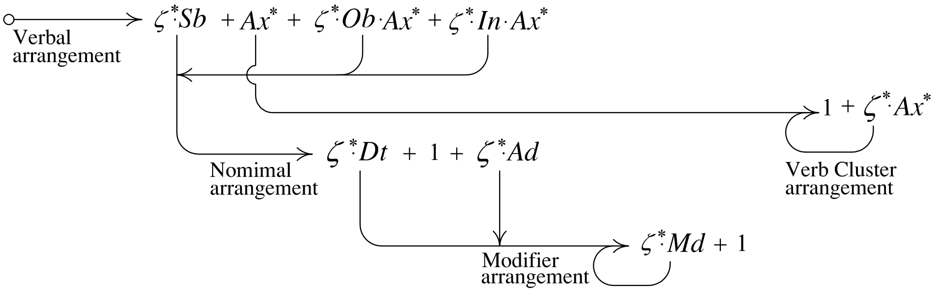


Figure 11.4: (a) non-adequate arrangement for a sentence with a verb cluster; (b) adequate arrangement for a sentence with a verb cluster.

See Fig. 11.4(b). Here the monoid Ax^* captures the chain of verbs, so we need an extra arrangement to linearize this, namely $1+\zeta^*{\cdot}Ax$. The system of arrangements for Catalan becomes:



The consequence of capturing the verb cluster with a pattern is that it does not allow the intercalation of material between the verbs.

11.6.2 Wh-Fronting in Romance Languages

First let us examine the case of wh-fronting with simple verbs. We form questions for the functions *Sb*, *Ob* and *Ap*; for the others functions there is a very similar treatment. Consider the following affirmative sentence and their interrogative forms:

(50) a. El Joan amaga les claus sota la catifa.
The John hides the keys under the carpet
'John hides the keys under the carpet.'

b. Què amaga el Joan sota la catifa? / Què amaga sota la catifa el Joan?
What hides the John under the carpet / What hides under the carpet the John
'What does John hide under the carpet?'

c. Qui amaga les claus sota la catifa? / Qui amaga sota la catifa les claus?
Who hides the keys under the carpet / Who hides under the carpet the keys
'Who hides the keys under the carpet?'

d. On amaga el Joan les claus? / On amaga les claus el Joan?
Where hides the John the keys / What hides the keys the John
'Where does John hide the keys?'

We consider the interrogative isotope functions $Sb_?$, $Ob_?$, $Ap_?$. They work argumentally similarly to the primitive functions but there are some important syntactic constraints. First, when they are not null, they must take an interrogative pronoun. That is, $Sb_?$ must take the interrogative pronoun *qui*; $Ob_?$, the pronoun *què*; $Ap_?$ the pronoun *on*, and so forth.[6] Second, we treat the case of only one interrogative isotope function per sentence. Third, the primitive function and its isotopic are orthogonal. These rules are implementable in a manifold using patterns in $\mathbf{k}\mathbb{H}$. Now we define the arrangements, for example, for the interrogative object. In Romance languages the interrogative pronoun goes to the left followed by the verb and then the remaining non-interrogative functions which can be placed in any order:

$$\begin{gathered}
\zeta^* \cdot Ob_? + 1 + \zeta^* \cdot Sb + \zeta^* \cdot Ap, \\
\zeta^* \cdot Ob_? + 1 + \zeta^* \cdot Ap + \zeta^* \cdot Sb, \\
\zeta^* \cdot Sb_? + 1 + \zeta^* \cdot Ob + \zeta^* \cdot Ap, \\
\zeta^* \cdot Sb_? + 1 + \zeta^* \cdot Ap + \zeta^* \cdot Ob, \\
\zeta^* \cdot Ap_? + 1 + \zeta^* \cdot Sb + \zeta^* \cdot Ob, \\
\zeta^* \cdot Ap_? + 1 + \zeta^* \cdot Ob + \zeta^* \cdot Sb.
\end{gathered}$$

[6] In English: *who, what, where*.

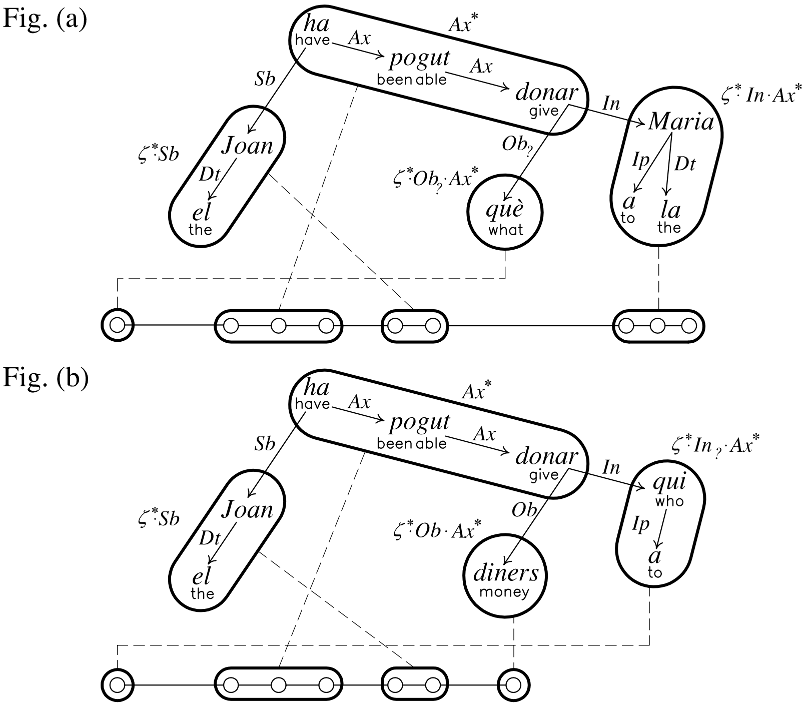


Figure 11.5: (a) and (b) wh-fronting with verbal cluster in Catalan.

The following expression (the first pair of parentheses are just a visual help):

$$\Big(\zeta^* \cdot Sb_? + \zeta^* \cdot Ob_? + \zeta^* \cdot Ap_?\Big) + 1 + \mathrm{scr}\Big(\zeta^* \cdot Sb + \zeta^* \cdot Ob + \zeta^* \cdot Ap\Big)$$

indeed does the same as the six arrangements above since primitive functions and their interrogative isotopes are orthogonal.

The last is a projective arrangement in 𝔾**k**. However wh-fronting, even in Romance languages, is not projective in general. One reason is the presence of clustering verbs. Consider the questions for object and indirect object with a verb cluster:

(51) a. Què ha pogut donar el Joan a la Maria?
What has been-able give the John to the Mary
'What has John been able to give to Mary?'

b. A qui ha pogut donar el Joan diners?
To whom has been-able give the John money
'To whom has John been able to give money?'

See the analyses in Fig. 11.5(a) and Fig. 11.5(b). From these interrogatives we can infer an arrangement for the object interrogative:

$$\zeta^* \cdot Ob_? \cdot Ax^* + Ax^* + \zeta^* \cdot Sb + \zeta^* \cdot In \cdot Ax^*,$$

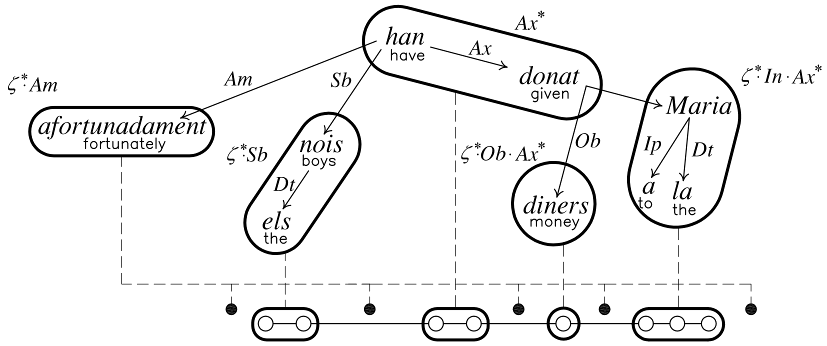


Figure 11.6: adjunct with auxiliary cluster in Catalan.

and for the indirect object interrogative:

$$\zeta^* \cdot In_? \cdot Ax^* + Ax^* + \zeta^* \cdot Sb + \zeta^* \cdot Ob \cdot Ax^* .$$

Since the interrogative isotopes are mutually orthogonal and orthogonal to their primitive functions, we can summarize in one go the verbal arrangement:

$$\Big(\zeta^* \cdot Sb_? + \zeta^* \cdot Ob_? \cdot Ax^* + \zeta^* \cdot In_? \cdot Ax^*\Big) + Ax^* + \mathrm{scr}\Big(\zeta^* \cdot Sb + \zeta^* \cdot Ob \cdot Ax^* + \zeta^* \cdot In \cdot Ax^*\Big).$$

Some word-orders are more usual than others and the emphasis or focus can change. Notice in addition that the complexity has been increased with respect to the arrangements for an interrogative without verb cluster: now all the patterns inhabits $\mathbb{G}\mathbf{k}\mathbb{G}$.

11.6.3 Adjuncts in Romance Languages

Adjuncts are complements which have a great mobility in Romance languages. However as we can deduce from last subsection, an adjunct cannot penetrate a verb cluster, nor another verb complement. We list all the combinations of the following sentence the analysis of which is depicted in Fig. 11.6 (parentheses indicate the possible positions):

(52) (Afortunadament) el Joan (afortunadament) ha donat (afortunadament)
(Fortunately) the John (fortunately) has given (fortunately)
diners (afortunadament) a la Maria (afortunadament).
money (fortunately) to the Mary (fortunately)
'Fortunately John have given Mary money.'

However, the adjunct cannot appear in the domain of the auxiliary verb, unlike English:

(53) *El Joan ha afortunadament donat diners a la Maria.
The John has fortunately given money to the Mary

So the arrangement should be:

$$\zeta^* \cdot Am \bowtie \left(\zeta^* \cdot Sb + Ax^* + \zeta^* \cdot Ob \cdot Ax^* + \zeta^* \cdot In \cdot Ax^*\right).$$

In Catalan the adjuncts of place, time and mode seem to have the same status. So we can condense them in the arrangement:

$$\zeta^* \cdot At \bowtie \zeta^* \cdot Ap \bowtie \zeta^* \cdot Am \bowtie \left(\zeta^* \cdot Sb + Ax^* + \zeta^* \cdot Ob \cdot Ax^* + \zeta^* \cdot In \cdot Ax^*\right).$$

Recall that we have the commutation $x \bowtie y \bowtie z = y \bowtie x \bowtie z$.

11.6.4 Floating Quantifiers in Romance Languages

A floating quantifier is a quantifier of the subject with the capability of moving over the sentence creating non-projective linearizations.[7] This particle should not penetrate in any domain defined by the patterns of the basic functions. So in this sense floating quantifier and adjuncts behave very similarly.[8]

However from the point of view of algebraic complexity floating quantifiers are a bit more sophisticated. We need to extract an element the depth of which is greater than one (the quantifier of the subject and its possible modifiers, $\zeta^* \cdot Qn \cdot Sb \in \mathbb{G}\mathbf{k}^2$), which is new in our inventory of patterns. Consider the sentences (analysis in Fig 11.7):

(55) (Tots) els nois (tots) han donat (tots) diners (tots) a la Maria (tots).
(All) the boys (all) have given (all) money (all) to the Mary (all)
'All the boys have given Mary money.'

[7] Solà et al. (2010) §8.2.5.4 define a floating quantifier as a quantifier which "appears in postverbal position" (translation is ours). However the definition of Osborne (2013) is more general and adequate in the ambit of dependencies: "a quantifier is floating if, for whatever reason, it cannot be construed as a dependent of the nominal that it quantifies", where "construed" must be understood as "projectively construed".

[8] According to Osborne (2013) a floating quantifier and an adjunct of mode have a similar distribution in English. Consider the sentences from (Osborne, 2013):

(54) a. ?The kids all will have been seen.
b. The kids will all have been seen.
c. The kids will have all been seen.
d. ??The kids will have been all seen.

The reason for the difficulty in comparation to Romance languages is, beside reasons that Osborne relates, the intercalation of material in the verb cluster. However we are not going to treat the construction in English.

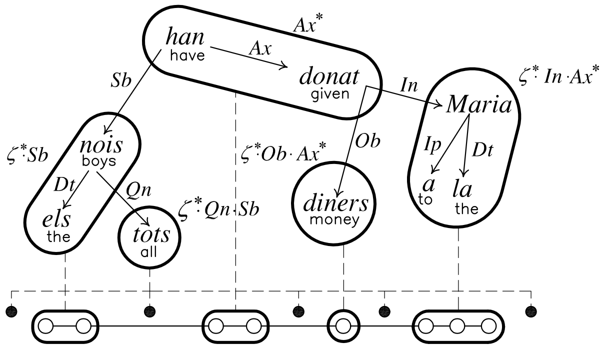


Figure 11.7: floating quantifiers in Catalan.

The following arrangement yields all the combinations:[9]

$$\zeta^*{\cdot}Qn{\cdot}Sb \bowtie \Big((\zeta - Qn)^*{\cdot}Sb + Ax^* + \zeta^*{\cdot}Ob{\cdot}Ax^* + \zeta^*{\cdot}In{\cdot}Ax^*\Big).$$

Notice that we have extracted the quantifier *Qn* from the set of functions in the subtree pattern of the subject, otherwise we would not have a partition of the syntagma. Now the involved patterns inhabits $\mathbb{G}\mathbf{k}\mathbb{G}\mathbf{k}$, or alternatively, $\mathbb{G}\mathbf{k}^2\mathbb{G}$.

11.7 Some English Constructions

11.7.1 Wh-Fronting in English

English, unlike Romance languages, can split the auxiliary cluster. Since questions for the most case are made through an auxiliary, this effect is observable with simple questions. These constructions suggests that the head of the verb cluster must be split from the rest of the cluster in order to interpose the subject, as in the sentence:

(56) What will John have given Mary?

The analysis is in Fig. 11.8 and it works with the decomposition: $\zeta^*{\cdot}Sb$, 1, $Ax^*{\cdot}Ax$, $\zeta^*{\cdot}Ob{\cdot}Ax^*$, and $\zeta^*{\cdot}In{\cdot}Ax^*$. With this we can construct the arrangement:

$$\zeta^*{\cdot}Ob_{?}{\cdot}Ax^* + 1 + \zeta^*{\cdot}Sb + Ax^*{\cdot}Ax + \zeta^*{\cdot}In{\cdot}Ax^*.$$

This arrangement must be generalized. Sometimes the interrogative function is in a deep position headed by a verb cluster containing not only auxiliary functions, but

[9] Floating quantifier can appear in other positions, for example in a subordinate clause, as in *Aquests reis els quals van morir tots sense descendència, van regnar breument* (these kings who died all without descendents reigned briefly). However such positions can be explained locally in the subordinate clause: the quantifier is attaching the relative pronoun *quals* ('who') which substitutes the subject of the main clause. So the same arrangement is still valid.

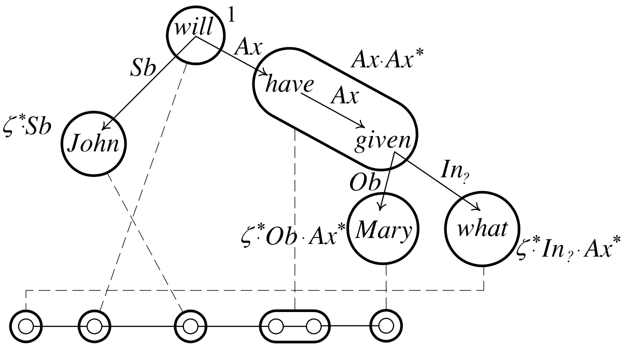


Figure 11.8: wh-fronting in English.

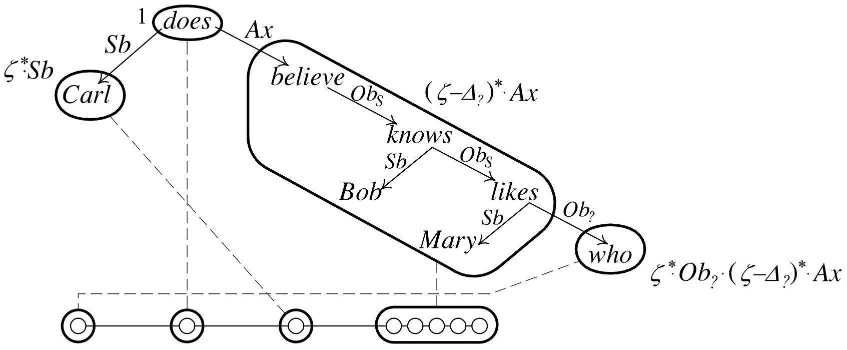

1
does
Ax
Sb
ζ^*Sb
Carl
believe
Ob_S
$(\zeta\text{–}\Delta_?)^*\cdot Ax$
knows
Sb
Ob_S
Bob
likes
Sb
$Ob_?$
Mary
who
$\zeta^*Ob_?\cdot(\zeta\text{–}\Delta_?)^*\cdot Ax$

Figure 11.9: a deep position of the interrogative function.

subordinate functions, see Fig. 11.9. So in order to capture the verbal cluster we must consider the set of functions $\Delta_S = \{Ax, Sb_S, Ob_S, In_S, \ldots\}$, and in addition possible other complements containing any other function. So we have to consider the monoid: $(\zeta - \Delta_?)^*$ which contains any function excepting the interrogative isotopes, which must be allocated at the left margin of the sentence. So the following arrangement:

$$\zeta^*\cdot Ob_?\cdot(\zeta - \Delta_?)^*\cdot Ax + 1 + \zeta^*\cdot Sb + (\zeta - \Delta_?)^*\cdot Ax,$$

with type $\mathbb{G}\mathbf{k}\mathbb{G}\mathbf{k}$, linearizes the following sentence depicted in Fig. 11.9:

(57) Who does Carl believe that Bob knows that Mary likes?

Notice that the subtraction of interrogative isotopes $\Delta_?$ from the set of all the functions ζ grants that the patterns $\zeta^*\cdot Ob_?\cdot(\zeta - \Delta_?)^*\cdot Ax$ be able to induce subsyntagmata. This last can be then generalized in order to make a question of any other verbal function in $\Delta_?$

$$\zeta^*\cdot\Delta_?\cdot(\zeta - \Delta_?)^*\cdot Ax + 1 + \zeta^*\cdot Sb + (\zeta - \Delta_?)^*\cdot Ax.$$

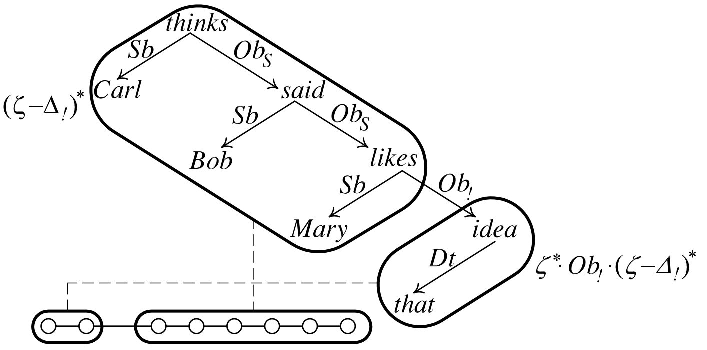


Figure 11.10: Topicalization in English.

11.7.2 Topicalization in English

Topicalization occurs when a speaker emphasizes some part of the sentence by displacing it to the beginning. This construction is closely related to wh-fronting. The following sentence topicalizes the embedded object of the sentence *Carl thinks Bob said Mary likes that idea*. See Fig. 11.10:

(58) That idea, Carl thinks Bob said Mary likes.

The arrangement simply picks up the topicalizated subsyntagma (which is marked by the isotope $Ob_!$) and places it at the left margin:

$$\zeta^*\cdot Ob_!\cdot(\zeta-\Delta_!)^* + (\zeta-\Delta_!)^*,$$

with type $\mathbb{G}\mathbf{k}\mathbb{G}$.

11.7.3 Extraposition in English

Extraposition consists in the possibility of extracting a noun complement and locating it at the right. Consider the pair of the sentences in English:

(59) a. A hearing on the issue is scheduled today.
b. A hearing is scheduled on the issue today.

The first is the non-problematic version; we can use the projective arrangement in $\mathbb{G}\mathbf{k}$: $\zeta^*\cdot Sb + 1 + Ax + \zeta^*\cdot At\cdot Ax$. However the second version has “moved” a constituent (the subtree $\zeta^*\cdot Nc\cdot Sb$) to the right. Fig. 11.11(a) and Fig. 11.11(b) show both versions. To recreate this apparent movement we can use an arrangement in $\mathbb{G}\mathbf{k}^2$. In this space of linearizations we are allowed to pick up subtrees down until depth 2. For example:

$$(\zeta-Nc)^*\cdot Sb + 1 + Ax + \zeta^*\cdot Nc\cdot Sb + \zeta^*\cdot At\cdot Ax,$$

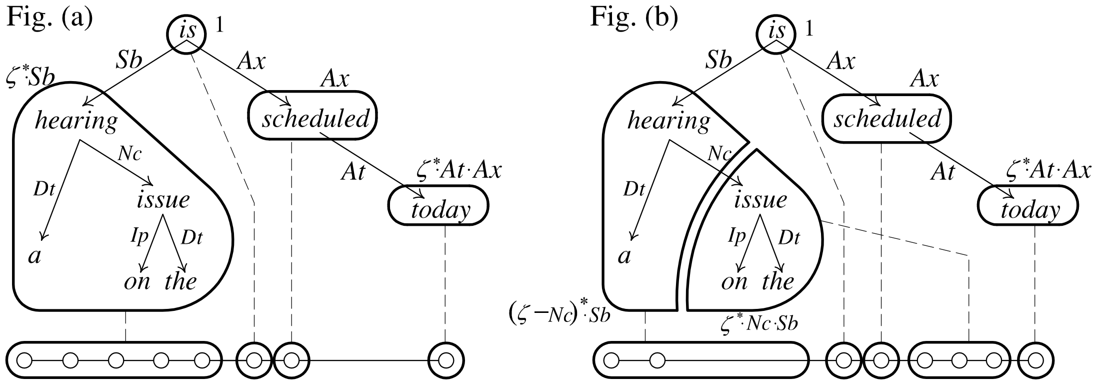


Figure 11.11: (a) canonical order in English; (b) the same sentence with extraposition.

11.8 Discontinuities in Classical Latin

We can increase the depth of patterns a bit more. In classical Latin texts we can find examples of extensive free word-order, Kessler (1995). Consider the following sentence of Cicero which exhibits two discontinuities:

(60) Hic optimus illis temporibus est patronus habitus.
He the-best those days is lawyer considered
'He was considered the best layer in those days.'

The sentence can be linearized (see Fig. 11.12) by an arrangement in $\mathbb{G}\mathbf{k}^3$:

$$\zeta^* \cdot Sb + \zeta^* \cdot Md \cdot Ob \cdot At + \zeta^* \cdot Nc \cdot Ob \cdot Ax + (\zeta - \{Md, Nc\})^* \cdot Ob \cdot Ax + \zeta^* \cdot Ax.$$

The exponent 3 arises in the type of the pattern because, firstly, the word *optimus* is allocated at depth 3 ($Md \cdot Ob \cdot At$) and secondly, the subsyntagmata *illis temporibus* is also at depth 3 ($\zeta^* \cdot Nc \cdot Ob \cdot Ax$). Furthermore the order of the subsyntagma *patronus* is controled by the pattern $(\zeta - \{Md, Nc\})^* \cdot Ob \cdot Ax$ where we need to subtract the functions responsible for the discontinuities.

11.9 Some Germanic Constructions

Swiss-German, Dutch and German, and in general West Germanic languages, have been extensively studied as a paradigm of complex word-order. These languages set the verbal cluster and its arguments (which we name *serial dependences*) in very particular configurations. There are two modalities of linearizing the serial dependencies in West Germanic languages which create: *cross-serial dependences* and *nested-serial dependences*, see Fig. 11.13. As discussed in §4.3.1, in the abstraction of formal languages, the first is related to the copy language L_{copy}, while the second is related to the mirror language L_{mirr}. We have implemented manifolds and linearizations capturing L_{copy} and

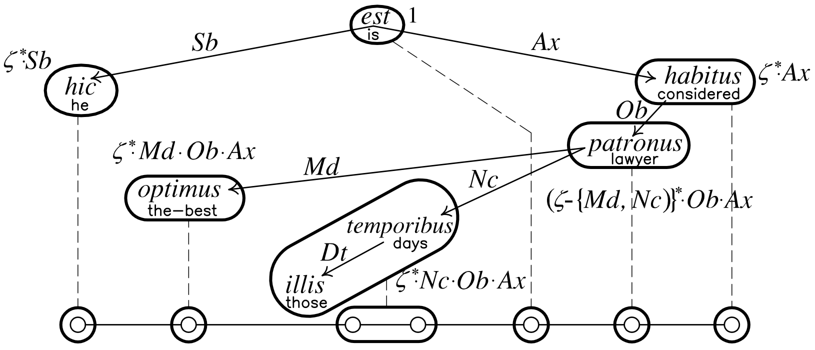


Figure 11.12: Discontinuity in Classical Latin.

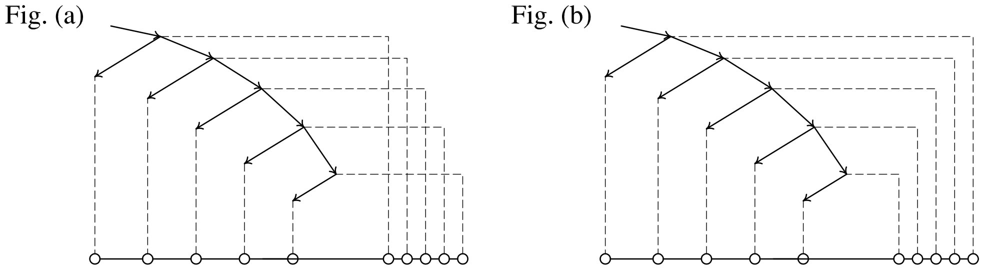


Figure 11.13: scheme of linearization of (a) cross-serial dependencies; (b) nested-serial dependencies.

L_{mirr}; in particular the mirror language can be implemented in two ways (which are symmetric).

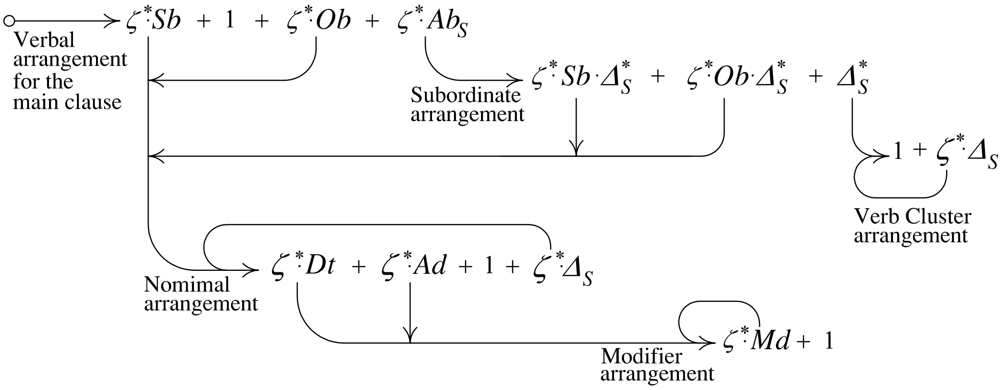
11.9.1 Cross-Serial Dependencies in Dutch

We examine first the Dutch case. For example the following sentence exhibits an object at the end of the verbal cluster which in the linearization intercalates between the subject arguments and the verb cluster: [10]

(61) . . . omdat ik Cecilia henk de nijlpaarden zag jelpen voeren.
. . . because I Cecilia Henk the hippopotamses saw help feed
‘. . . because I saw Cecilia help Henk feed the hippos.’

Fig. 11.14 depicts the analysis. The tree is controlled by a manifold in **k**ℍ and the linearization is provided by the system of arrangements in 𝔾**k**𝔾:

[10] Example from Steedman (1985).


$\zeta^{*}\cdot Sb + 1 + \zeta^{*}\cdot Ob + \zeta^{*}\cdot Ab_S$
Verbal arrangement for the main clause
$\zeta^{*}\cdot Sb\cdot\Delta_S^{*} + \zeta^{*}\cdot Ob\cdot\Delta_S^{*} + \Delta_S^{*}$
Subordinate arrangement
$1 + \zeta^{*}\cdot\Delta_S$
Verb Cluster arrangement
$\zeta^{*}\cdot Dt + \zeta^{*}\cdot Ad + 1 + \zeta^{*}\cdot\Delta_S$
Nominal arrangement
$\zeta^{*}\cdot Md + 1$
Modifier arrangement

The main clause is linearized by a projective arrangement (the verbal arrangement). It separates on the left side the main clause and puts on the right side the subordinate clause. Let us examine the subordinate arrangement which contains the key part for the linearization. The decomposition of the patterns is depicted in Fig. 11.14. Notice that the first pattern $\zeta^{*}\cdot Sb\cdot\Delta_S^{*}$ is a new linguistic substructure, it is not a full subtree nor a catena. Although this pattern shows up discontinuous in its graphical representation, it induces a subsyntagma consisting of a succession of subject arguments chained by subordinate isotopes in Δ_S. This chain $S_{\zeta^{*}\cdot Sb\cdot\Delta_S^{*}}$ indicates the order to linearize: *ik Cecilia Henk*. More exactly, for the next step in the linearization we have to take the nominal arrangement $\zeta^{*}\cdot Dt + \zeta^{*}\cdot Ad + 1 + \zeta^{*}\cdot\Delta_S$. So the chain $S_{\zeta^{*}\cdot Sb\cdot\Delta_S^{*}}$ behaves as a stack. The nominal arrangement extracts the head of the stack (the first subject) and puts it at the left in the word-order. We repeat the nominal arrangement until the stack is empty.[11]

The second pattern in the subordinate arrangement is $\zeta^{*}\cdot Ob\cdot\Delta^{*}$ and this is isolating the *hippos*. Finally it remains to place the verbal cluster, *zag helpen voeren*. This is achieved by the verb cluster arrangement. This arrangement is the one responsible for the crossing distribution.

11.9.2 Nested-Serial Dependencies in German

Now we consider the German sentence:

(62) . . . Hans Peter Marie schimmen lassen sah.
. . . Hans Peter Marie swim help saw
. . . ‘Hans saw Peter help Marie to swim.’

Curiously the German case could be achieved simply through a projective arrangement, namely: $\zeta^{*}\cdot Sb + \zeta^{*}\cdot\Delta_S + 1$. see Fig. 11.15(a). We would not need any further arrangement: applying it several times we would obtain the nested dependences. However, and given the familiarity and closeness between both languages, one could hope for some

[11] Notice that the summand $\zeta^{*}\cdot\Delta_S$ does not intervene when we come from the main clause, since in this case the pattern $\zeta^{*}\cdot\Delta_S$ is null. So the nominal arrangement can be used after the verbal arrangement for the main clause or after the subordinate arrangement.

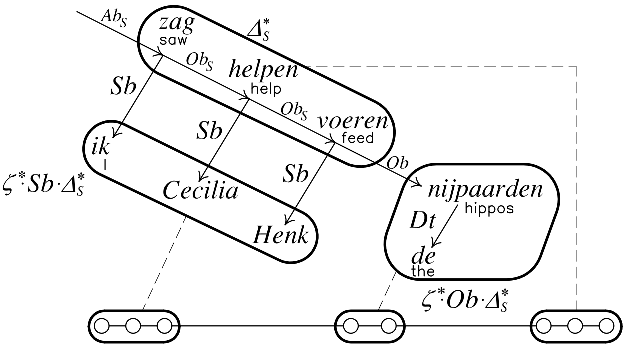
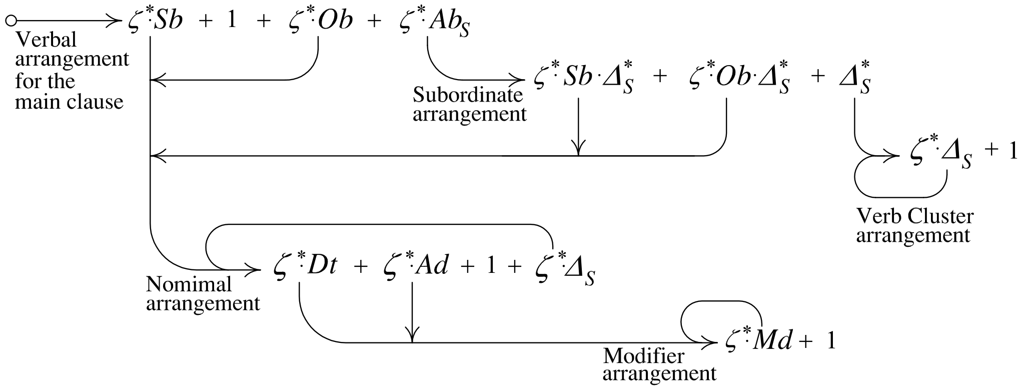


Figure 11.14: cross-serial dependences in Dutch.

deeper algebraic relation. In fact, we think that the last arrangement is not adequate. As in the abstract case of the mirror language for which there were two ways to obtain the same language (one using arrangements in $\mathbb{G}\mathbf{k}$ and other in $\mathbf{k}\mathbb{G}$), we have also the non-projective alternative in order to linearize the German sentences, see §25. We consider the same system of arrangements as for Dutch with just a small difference:


$\zeta^*Sb + 1 + \zeta^*Ob + \zeta^*Ab_S$
Verbal arrangement for the main clause
$\zeta^*Sb{\cdot}\Delta_S^* + \zeta^*Ob{\cdot}\Delta_S^* + \Delta_S^*$
Subordinate arrangement
$\zeta^*\Delta_S + 1$
Verb Cluster arrangement
$\zeta^*Dt + \zeta^*Ad + 1 + \zeta^*\Delta_S$
Nomimal arrangement
$\zeta^*Md + 1$
Modifier arrangement

Notice the transposition of the patterns 1 and $\zeta^* \cdot \Delta_S^*$ in the verb cluster arrangement. This small change alone produces the nested dependencies instead of the crossing dependencies, see Fig. 11.16. So, under this point of view, German and Dutch are very similar, and the types used are the same: $\mathbb{G}\mathbf{k}\mathbb{G}$.

11.9.3 Scrambling in German

The last German sentence leads on to another much studied phenomenon called *scrambling*. While the verb cluster must continue being ordered inversely causing nested dependencies (*schimmen lassen sah*, unlike the English formula *saw help to swim*), when we have several distinct complements hanging from the verbal cluster, these complements can permute. We begin with a simple case (examples have been extract from (Becker et al., 1992)):

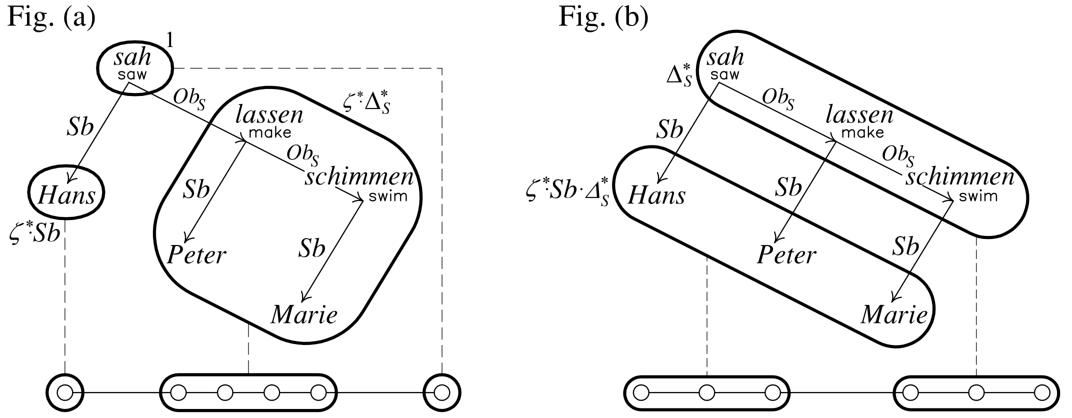


Figure 11.15: (a) non-adequate decomposition of German; (b) adequate decomposition for nested-serial dependences in German.

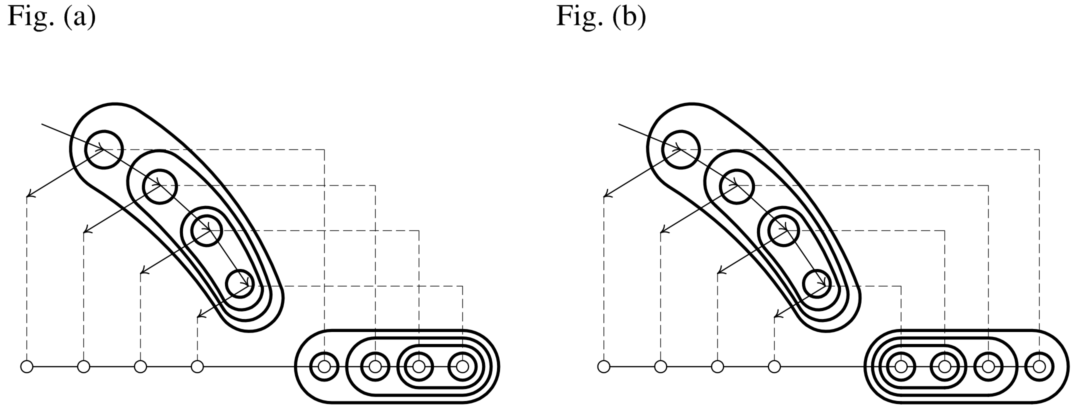


Figure 11.16: the two forms to linearize the verbal cluster, (a) Dutch and (b) German.

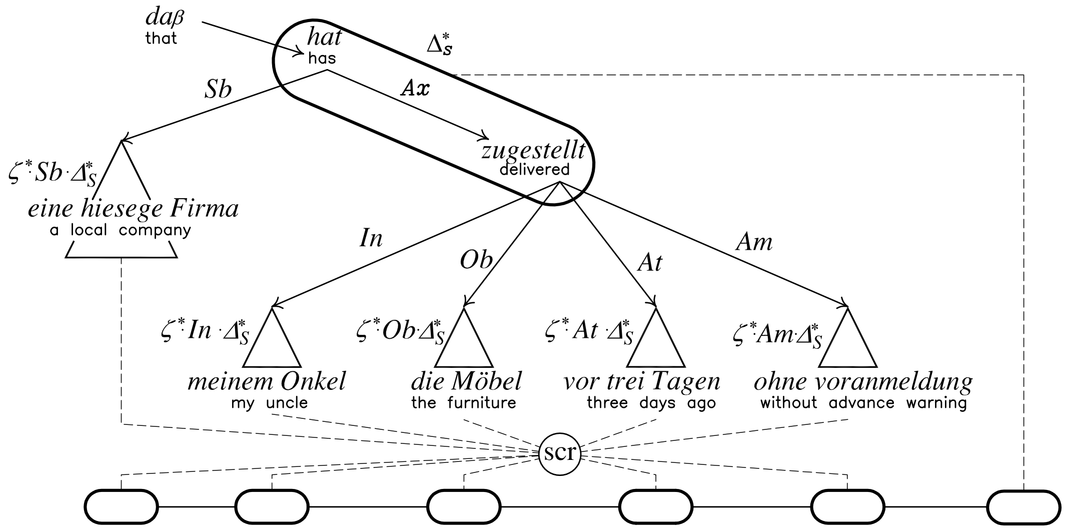


Figure 11.17: scrambling in German.

(63) . . . daß eine hiesige Firma meinem Onkel die Möbel vor trei Tagen
. . . that a local company my uncle the furniture three days ago
ohne Voranmeldung zugestellt hat.
without advance-warning delivered has
. . . 'that a local company delivered the furniture to my uncle three days ago without advance warning.'

According to these authors this sentence accepts any reordering of the elements between brackets:

(64) . . . daß [eine hiesige Firma] [meinem Onkel] [die Möbel] [vor trei Tagen] [ohne Voranmeldung] zugestellt hat.

which yields 5! = 120 possible word-orders, see Fig 11.17.

However it is not necessary that complements be at the same level to permute (in fact the subject is already at a different level). We also have scrambling when the complements are distributed along the verbal cluster, as in the sentence (see Fig 11.18):

(65) . . . daß dem Kunden den Kuehlschrank bisher noch niemand zu reparieren zu
. . . that the client the refrigerator so-far yet no-one to repair to
versuchen versprochen hat.
try promised has
'. . . that so far no-one has promised the client to try to repair the refrigerator.'

Such constructions can be treated with the scrambling operator:

$$\mathbf{scr}\left(\zeta^*\cdot Sb\cdot\Delta_S^* + \zeta^*\cdot Ob\cdot\Delta_S^* + \zeta^*\cdot In\cdot\Delta_S^* + \zeta^*\cdot At\cdot\Delta_S^*\right) + \Delta_S^*$$

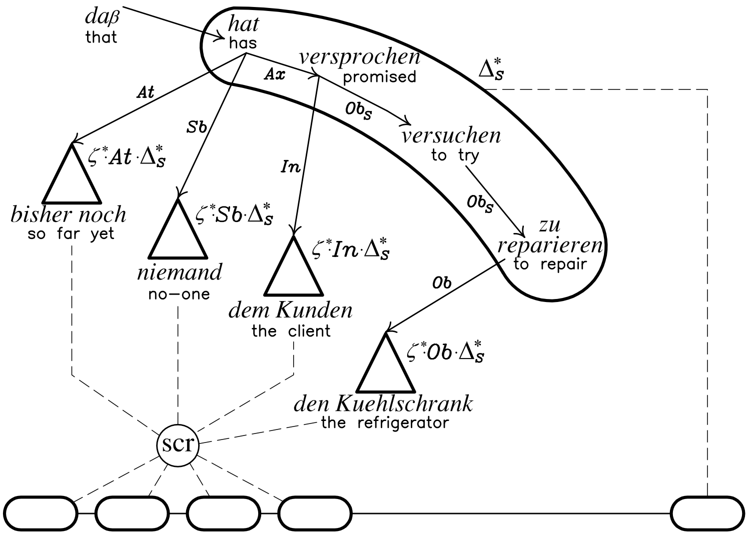
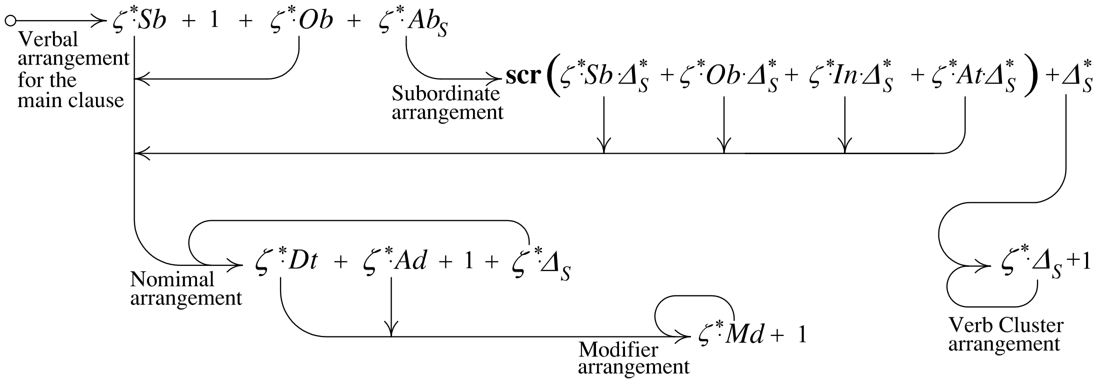


Figure 11.18: scrambling in German with several complements distributed at several levels.

Notice that this is a generalization of the arrangement from the last section. So the system of arrangements for German should be:


$\zeta^* Sb + 1 + \zeta^* Ob + \zeta^* Ab_S$
Verbal arrangement for the main clause
Subordinate arrangement
$\mathbf{scr}\left(\zeta^* Sb \cdot \Delta_S^* + \zeta^* Ob \cdot \Delta_S^* + \zeta^* In \cdot \Delta_S^* + \zeta^* At \Delta_S^*\right) + \Delta_S^*$
$\zeta^* Dt + \zeta^* Ad + 1 + \zeta^* \Delta_S$
Nomimal arrangement
$\zeta^* Md + 1$
Modifier arrangement
$\zeta^* \Delta_S + 1$
Verb Cluster arrangement

All the arrangements used in the West Germanic examples are in $\mathbb{G}\mathbf{k}\mathbb{G}$.

11.10 The Respectively Construction

Ordinary coordination of sentences can be treated comfortably with projective arrangements. Let us consider the sentence:

(66) The young boy is English, the fat man German, and the blond woman Dutch.

Fig. 11.19 shows how to obtain the word-order with the arrangement:

$$\zeta^* \cdot Sb + 1 + \zeta^* \cdot Pd + \zeta^* \cdot Co;$$

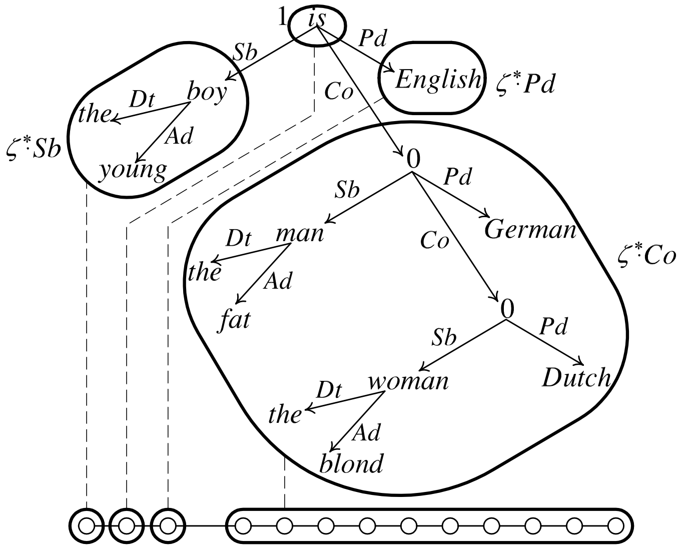


Figure 11.19: projective linearization of coordination.

We repeat this arrangement in $\zeta^* \cdot Co$. (Here Co is the function which introduces the conjuncts). Notice that we are supposing that there are two elliptic verbs: *The young boy is English, the fat man ∅(is) German, and the blond ∅(is) woman Dutch.*[12]

However the coordination must exhibit a non-projective form in the presence of the adverb *respectively*:

(67) The young boy, the fat man and the blond woman are respectively English, German and Dutch.

Some authors claim that respectively sentences are not natural or at least they are stylistically inadequate. Here we take such sentences at face value as constituting grammatical parts of natural language.[13] As discussed in §4.3.1, in the abstraction of formal languages, the first is related to the language L_{mult} while the second is related to the language L_{resp}, which are symmetric.

Our formalism can achieve this linearization as in Fig. 11.20 by applying first the arrangement:

$$\zeta^* \cdot Sb \cdot Co^* + Co^* + \zeta^* \cdot Pd \cdot Co^*.$$

The linearization then continues with the projective arrangement:

$$\zeta^* \cdot Dt + \zeta^* \cdot Ad + 1 + \zeta^* \cdot Co.$$

So the patterns of the system of arrangements inhabit $\mathbb{G}\mathbf{k}\mathbb{G}$.

[12] See Appendix A for the question of the ellipsis and coordination in general.

[13] See arguments in pro in: (Kac et al., 1987), (Bar-Hillel and Shamir, 1960), and arguments in contra in: (Pullum and Gazdar, 1982), (de Cornulier, 1973).

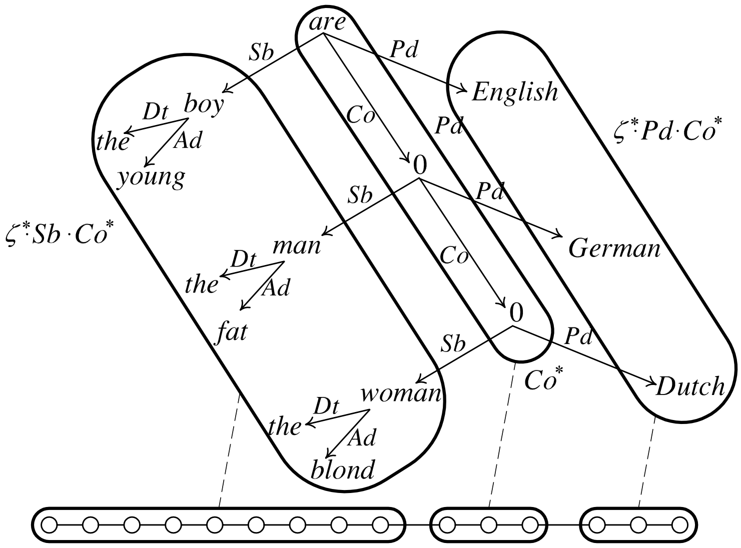


Figure 11.20: linearization of the respectively construction.

12

Algebraic Complexity of Natural Language, Prospects, and Conclusions

We collect the patterns of the cases examined in Chapters 10 and 11 and we study which bi-hierarchy is the most adequate in order to locate natural languages. This constitutes one of our final goals.

In order to carry this out we have to take two methodological decisions. First, it is not clear whether the naturalness of some extreme constructions must be accepted. We see that these seem to separate themselves from the well-accepted constructions by their algebraic description. Second, we have to distinguish structural descriptions from descriptions based only on weak-capacity. We examine both cases. In the first case we observe that natural constructions can be grouped into phenomena according their description in a classificatory monoid. In the second case we show that all the phenomena can be reduced to a certain class. We discuss some significant properties of this class in relation to mild context-sensitivity.

We introduce the last concept of the thesis: *transposition arrangement*. This permits a decomposition of an arrangement into minimal units, allowing explanation of the fine structure of a linearization. We suggest that this could be the key in order to introduce semantics in the model in future work.

12.1 The Algebraic Complexity of Natural Languages

12.1.1 Data

Having reviewed several constructions in natural languages, it is natural to ask which bi-hierarchy is the most adequate to organize them, and whether the whole class of possible natural languages can be located.

To respond to this let us collect all the cases viewed in Chapters 10 and 11, in a pair of tables, Table 12.1 and Table 12.2. The first lists the patterns and their types involved in manifolds and the second does the same for linearizations.

Table 12.1: patterns in the manifolds and their types in the natural constructions reviewed in Chapter 10. Abbreviations: English **(Eng)** , Romance languages **(Rom)** , classical Latin **(Lat)** , German **(Ger)** , Dutch **(Dut)** , Chinese **(Chi)** , Old Georgian **(OGe)** , Yoruba (West African) **(Yor)** .

Description of the natural construction	**Pattern in the manifold**	**Type**
Verb Main Principle, **(Eng)** , **(Rom)** , . . .	1	$\mathbf{1}$
Orthogonality, Valence and Selection, **(Eng)** , **(Rom)** , . . .	$(1, Sb)\cdot(\zeta)^*_2$	$\mathbf{k}\mathbb{H}$
Local Agreements, **(Eng)** , **(Rom)**	$(1, Sb)\cdot(\zeta)^*_2$	$\mathbf{k}\mathbb{H}$
Obligatory Determiner, **(Rom)**	$(Sb, Dt\cdot Sb)\cdot(\zeta)^*_2$	$\mathbf{k}^2\mathbb{H}$
Long Distance Agreement of Predicate, **(Eng)** , **(Rom)**	$(Sb, Pd)\cdot(1\oplus\Delta^*)\cdot(\zeta)^*_2$	$\mathbf{k}\mathbb{P}\mathbb{H}$
Long Dist. Agreement of Coordination Adjectives, **(Rom)**	$(1\oplus Co^*_\wedge)\cdot(1, Ad)\cdot(\zeta)^*_2$	$\mathbb{P}\mathbf{k}\mathbb{H}$
Combination of the previous two, **(Rom)**	$(Sb, 1)\cdot(1\oplus Co^*_\wedge)\cdot(1, Pd)\cdot(1\oplus\Delta^*)\cdot(\zeta)^*_2$	$(\mathbf{k}\mathbb{P})^2\mathbb{H}$
Long Distance Agreement in Pied-Piping, **(Rom)**	$(1, Dt)\cdot(1, Nc_R)\cdot(1\oplus Nc^*)\cdot(1\oplus Co^*_\wedge)\cdot(1,\Delta)\cdot(1\oplus\Delta^*_S)\cdot(1, Ad_S)\cdot(\zeta)^*_2$	$\mathbf{k}^2\mathbb{P}^2\mathbf{k}\mathbb{P}\mathbf{k}\mathbb{H}$
Chinese Numbers, **(Chi)**	$(\beta, 1)\cdot(\beta,\beta)^*\cdot(\alpha,\alpha)^*\cdot(1,\alpha)$	$\mathbf{k}\mathbb{G}^2\mathbf{k}$
Suffixaufnahme, **(OGe)**	$(\alpha^2,\alpha,\alpha,1)(\alpha)^*_4(\beta)^*_4(\beta,\beta,1,1)$	$\mathbf{k}^2\mathbb{G}^2\mathbf{k}$
Recursive Copy, **(Yor)** , **(Rom)**	$(\zeta)^*_2(Ob_S, Pd_S\cdot Co)(\zeta)^*_2$	$\mathbb{H}\mathbf{k}^2\mathbb{H}$

Table 12.2: patterns in the arrangements and their types in the natural constructions reviewed in Chapter 11. We only show some representative arrangements from the system of arrangement of the construction. Abbreviations: English **(Eng)**, Romance languages **(Rom)**, classical Latin **(Lat)**, German **(Ger)**, Dutch **(Dut)**, Chinese **(Chi)**, Old Georgian **(OGe)**, Yoruba (West African) **(Yor)**.

Description of the natural construction	**Arrangement**	**Type**
Canonical Word-Orders, **(Eng)**, **(Rom)**, . . .	$\zeta^*{\cdot}Sb + 1 + \zeta^*{\cdot}In + \zeta^*{\cdot}Ob$ $\zeta^*{\cdot}Dt + \zeta^*{\cdot}Ad + 1$ $\zeta^*{\cdot}Md + 1$	$\mathbb{G}\mathbf{k}$ $\mathbb{G}\mathbf{k}$ $\mathbb{G}\mathbf{k}$
Subordination, **(Eng)**, **(Rom)**, . . .	$\zeta^*{\cdot}Dt + \zeta^*{\cdot}Ad + 1 + \zeta^*{\cdot}Ad_S$ $\zeta^*{\cdot}Sb_R + \zeta^*{\cdot}Sb + 1 + \zeta^*{\cdot}In + \zeta^*{\cdot}Ob$	$\mathbb{G}\mathbf{k}$ $\mathbb{G}\mathbf{k}$
Coordination, **(Eng)**, **(Rom)**, . . .	$Ic + \zeta^*{\cdot}Sb + 1 + \zeta^*{\cdot}In + \zeta^*{\cdot}Ob + \zeta^*Co$ $Ic + \zeta^*{\cdot}Dt + \zeta^*{\cdot}Ad + 1 + \zeta^*Co$ $Ic + \zeta^*{\cdot}Md + 1 + \zeta^*Co$	$\mathbb{G}\mathbf{k}$ $\mathbb{G}\mathbf{k}$ $\mathbb{G}\mathbf{k}$
Shifting of Object and Indirect Ob., **(Eng)**	$\zeta^*{\cdot}Sb + 1 + \zeta^*{\cdot}In + \zeta^*{\cdot}Ob + \zeta^*{\cdot}In_\dagger$	$\mathbb{G}\mathbf{k}$
Verbal Cluster, **(Rom)**, **(Eng)**, . . .	$1 + \zeta^*{\cdot}\Delta_S$	$\mathbb{G}\mathbf{k}$
Shifting of the Subject, **(Rom)**	$\zeta^*{\cdot}Sb + 1 + \zeta^*{\cdot}Sb_\dagger + \zeta^*{\cdot}Ob + \zeta^*{\cdot}In$ $\zeta^*{\cdot}Sb_\dagger + \zeta^*{\cdot}Sb + 1 + \zeta^*{\cdot}Ob + \zeta^*{\cdot}In$	$\mathbb{G}\mathbf{k}$ $\mathbb{G}\mathbf{k}$
Wh-Fronting, **(Rom)**	$\left(\zeta^*{\cdot}Sb_? + \zeta^*{\cdot}Ob_?{\cdot}Ax^* + \zeta^*{\cdot}In_?{\cdot}Ax^*\right) + Ax^* + \mathrm{scr}\left(\zeta^*{\cdot}Sb + \zeta^*{\cdot}Ob{\cdot}Ax^* + \zeta^*{\cdot}In{\cdot}Ax^*\right)$	$\mathbb{G}\mathbf{k}\mathbb{G}$
Adjuncts, **(Rom)**	$\zeta^*{\cdot}At \bowtie \zeta^*{\cdot}Ap \bowtie \zeta^*{\cdot}Am \bowtie \left(\zeta^*{\cdot}Sb + Ax^* + \zeta^*{\cdot}Ob{\cdot}Ax^* + \zeta^*{\cdot}In{\cdot}Ax^*\right)$	$\mathbb{G}\mathbf{k}\mathbb{G}$
Floating Quantifiers, **(Rom)**	$\zeta^*{\cdot}Qn{\cdot}Sb \bowtie \left((\zeta - Qn)^*{\cdot}Sb + Ax^* + \zeta^*{\cdot}Ob{\cdot}Ax^* + \zeta^*{\cdot}In{\cdot}Ax^*\right)$	$\mathbb{G}\mathbf{k}^2\mathbb{G}$
Wh-Fronting, **(Eng)**	$\zeta^*{\cdot}\Delta_?{\cdot}(\zeta - \Delta_?)^*{\cdot}Ax + 1 + \zeta^*{\cdot}Sb + (\zeta - \Delta_?)^*{\cdot}Ax$	$\mathbb{G}\mathbf{k}\mathbb{G}\mathbf{k}$
Topicalisation, **(Eng)**	$\zeta^*{\cdot}Ob_!{\cdot}(\zeta - \Delta_!)^* + (\zeta - \Delta_!)^*$	$\mathbb{G}\mathbf{k}\mathbb{G}$
Extraposition, **(Eng)**	$(\zeta - Nc)^*{\cdot}Sb + 1 + Ax + \zeta^*{\cdot}Nc{\cdot}Sb + \zeta^*{\cdot}At{\cdot}Ax$	$\mathbb{G}\mathbf{k}^2$
Discontinuities in Classical Latin, **(Lat)**	$\zeta^*{\cdot}Sb + \zeta^*{\cdot}Md{\cdot}Ob{\cdot}At + \zeta^*{\cdot}Nc{\cdot}Ob{\cdot}Ax + (\zeta - \{Md, Nc\})^*{\cdot}Ob{\cdot}Ax + \zeta^*{\cdot}Ax$	$\mathbb{G}\mathbf{k}^3$
Cross-serial dependencies, **(Dut)**	$\zeta^*{\cdot}Sb{\cdot}\Delta_S^* + \zeta^*{\cdot}Ob{\cdot}\Delta_s^* + \Delta_S^*$	$\mathbb{G}\mathbf{k}\mathbb{G}$
Nested-serial dependencies, **(Ger)**	$\zeta^*{\cdot}Sb{\cdot}\Delta_S^* + \zeta^*{\cdot}Ob{\cdot}\Delta_s^* + \Delta_S^*$	$\mathbb{G}\mathbf{k}\mathbb{G}$
Scrambling, **(Ger)**	$\mathbf{scr}\left(\zeta^*{\cdot}Sb{\cdot}\Delta_S^* + \zeta^*{\cdot}Ob{\cdot}\Delta_S^* + \zeta^*{\cdot}In{\cdot}\Delta_S^* + \zeta^*{\cdot}At{\cdot}\Delta_S^*\right) + \Delta_S^*$	$\mathbb{G}\mathbf{k}\mathbb{G}$
Respectively construction, **(Eng)**, **(Rom)**, . . .	$\zeta^*{\cdot}Sb{\cdot}Co^* + Co^* + \zeta^*{\cdot}Pd{\cdot}Co^*$	$\mathbb{G}\mathbf{k}\mathbb{G}$

12.1.2 Strong Capacity of Natural Languages

Examining the data from Tables 12.1 and 12.2 we observe an interesting scheme. Constructions can be grouped in linguistic phenomena according to fragments of the classificatory monoid $\{\mathbf{k}, \mathbb{H}, \mathbb{P}, \mathbb{G}\}^*$ (given by regular expressions):

MANIFOLDS		
Top constraints	main verb principle.	$\mathbf{1}$
Local constraints	orthogonality, valence, selection, local agreements. obligatory determiner.	$\mathbf{k}\mathbb{H}$ $\mathbf{k}^*\mathbb{H}$
Non-local constraints	long distance agreements.	$\{\mathbf{k}, \mathbb{P}\}^*\mathbb{H}$
Non-semi-linearity	quadratic growth: Chin. numbers, *suffixaufnahme*. exponential growth: recursive copy.	$\mathbf{k}^2\mathbb{G}^2\mathbf{k}$ $\mathbb{H}\mathbf{k}^2\mathbb{H}$
LINEARIZATIONS		
Projectivity	canonical orders, subordination, coordination, shifting, verbal clusters.	$\mathbb{G}\mathbf{k}$
Extrapositions	extraposition English, discont. in classical Latin.	$\mathbb{G}\mathbf{k}^*$
Movements	wh-fronting, topicalization, floating quantifiers.	$\mathbb{G}\mathbf{k}^*\mathbb{G}\mathbf{k}^*$
Serial dependencies	cross-serial, nested-serial, scrambling, respectively construction.	$\mathbb{G}\mathbf{k}\mathbb{G}$

The fragment $\mathbf{k}^*\mathbb{H}$ in *local constraints* only contains one construction (*obligatory determiner* with type $\mathbf{k}^2\mathbb{H}$). However the increment of the exponent of a constant does not seems to create a new linguistic phenomenon: assuming a constraint of type say $\mathbf{k}^9\mathbb{H}$, although not likely, continues being a local constraint. Similarly the fragment $\mathbb{G}\mathbf{k}^*$ in *extrapositions* only contains two constructions (*extraposition in English* $\mathbb{G}\mathbf{k}^2$ and *discontinuity in classical Latin* $\mathbb{G}\mathbf{k}^3$), whereby one could imagine an extraposition of depth four or five. The same reasoning serves to establish the fragment $\mathbb{G}\mathbf{k}^*\mathbb{G}\mathbf{k}^*$.

Well-Accepted Constructions

We have a hierarchy of phenomena. For fragments concerning manifolds we have the inclusions: $\mathbf{Man}(1) \subseteq \mathbf{Man}(\mathbf{k}\mathbb{H}) \subseteq \mathbf{Man}(\mathbf{k}^*\mathbb{H}) \subseteq \mathbf{Man}(\{\mathbf{k}, \mathbb{P}\}^*\mathbb{H})$.[1] Assuming that the non-semi-linear constructions are not legitimately fragments of natural languages, every manifold should be described in $\mathbf{Man}(\{\mathbf{k}, \mathbb{P}\}^*\mathbb{H})$, note that $\mathbb{H}$ only appears once. Regarding linearizations we have the inclusions $\mathbf{Lin}(\mathbb{G}\mathbf{k}) \subseteq \mathbf{Lin}(\mathbb{G}\mathbf{k}^*) \subseteq \mathbf{Lin}(\mathbb{G}\mathbf{k}^*\mathbb{G}\mathbf{k}^*)$

[1] Here $\mathbf{Man}(\mathscr{X}) = \bigcup_{X\in\mathscr{X}} \mathbf{Man}(X)$, and $\mathbf{Lin}(\mathscr{X}) = \bigcup_{X\in\mathscr{X}} \mathbf{Lin}(X)$ where $\mathscr{X}$ is a subset of the classificatory monoid.

and $\mathbf{Lin}(\mathbb{G}\mathbf{k}\mathbb{G}) \subseteq \mathbf{Lin}(\mathbb{G}\mathbf{k}^*\mathbb{G}\mathbf{k}^*)$. We want to stress that here we are interested mostly in the structural description in the sense that we gave in Remark 7.11. This leads to the claim that:

Claim 5. Strong capacity. In view of data, and assuming exclusively semi-linear constructions, natural languages can be structurally described by grammars inhabiting the bi-hierarchy $\mathcal{BH}(\mathbf{k}, \mathbb{H}, \mathbb{P}, \mathbb{G})$, however only a fragment is used:

$$\{\mathbf{k}, \mathbb{P}\}^*\mathbb{H}\big/_{\{\mathbf{k}, \mathbb{G}\}^*}.$$

We could strengthen this claim: we can suppose that the complexity of natural language is bounded. Perhaps by natural brain limitations, exponents in constants cannot be too large: say two or three.[2] Equally the number of monoids of type $\mathbb{P}$ and $\mathbb{G}$ cannot be too large. From the examined cases one is tempted to write:

$$\mathsf{NT} \subseteq {}^{(\mathbf{k}^2\mathbb{P})^3\mathbb{H}}\big/_{(\mathbb{G}\mathbf{k}^3)^2}.$$

However doing this is a little precipitated. Data from the tables are not exhaustive and in addition more complex phenomena could be discovered. Notwithstanding, the number of these constructions should be finite and then the complexity should be bounded.

Extreme Constructions

There is controversy on the legitimacy of the non-semi-linear constructions, although the general opinion is that they are not really natural constructions. Suppose that these constructions were definitively approved. Then we would have to consider a larger fragment, say: $\{\mathbf{k}, \mathbb{P}, \mathbb{G}\}^*/\{\mathbf{k}, \mathbb{G}\}^* \subseteq \mathcal{BH}(\mathbf{k}, \mathbb{P}, \mathbb{G})$.

We do not have a firm opinion regarding the naturalness of the extreme constructions. However in algebraic terms, these constructions appear to separate themselves from the well-accepted ones. Notice that these have types of the form $\mathbf{k}^2\mathbb{G}^2\mathbf{k}$ and $\mathbb{H}\mathbf{k}^2\mathbb{H}$ which are not in the fragment $\{\mathbf{k}, \mathbb{P}\}^*\mathbb{H}$ of the well-accepted constructions. The significant characteristic of these constructions is that the type for monoids appears twice.

12.1.3 Weak Capacity of Natural Languages

The descriptions made in Chapters 10 and 11 aim to capture the structural features of natural constructions. They are made following what is considered a correct syntactic analysis, up to minor discrepancies. However, turning to questions of weak capacity,

[2] Even the sentence of Cicero (§11.8) which used a linearization of type $\mathbb{G}\mathbf{k}^3$ is considered a literary extreme example.

we can reanalyze these constructions using fewer resources. Let us demonstrate in the following example how long distance constraints can be recast locally with manifolds in $\mathbf{Man}(\mathbf{k}^2\mathbb{H})$. The trick consists in appending a "memory" to certain functions which remember the agreement feature:

Example 36. Reduction of long distance agreements to $\mathbf{Man}(\mathbf{k}^2\mathbb{H})$. We consider again sentences viewed in §10.5.1 such as *she would want to try to be an actress*. In English the underlined words must agree in gender. The pattern involved was $(Sb, Pd)\cdot(1\oplus\Delta^*)\cdot(\zeta)_2^* \in \mathbf{k}\mathbb{PH}$, where $\Delta = \{Ax, Ob_S\}$; see Fig. (a).

We can obtain the same agreement effect with a manifold in $\mathbf{Man}(\mathbf{k}^2\mathbb{H})$. We pick the letters F and M for the gender cases *feminine* and *masculine* (we model only the gender features, but the other agreement features are obtained similarly). Given the original set of functions $\zeta = \{Sb, Ax, Ob_S, Pd, Dt\}$, we create a new set:

$$\overline{\zeta} = \{Sb, Dt\} \cup \{F, M\} \times \{Ax, Ob_S, Pd\}.$$

For clarity we notate the new functions $(F, Ax), (M, Ax), (F, Ob_S), (M, Ob_S), \ldots$ as $Ax^F, Ax^M, Pd^F, Pd^M, \ldots$. We group functions in*feminine functions* $\zeta^F = \{Ax^F, Ob_S^F, Pd^F\}$ and *masculine functions* $\zeta^M = \{Ax^M, Ob_S^M, Pd^M\}$.

We translate ordinary constraints about selection, valence and orthogonality to the new functions. For example the functions Ax^F and Ax^M select a verb as before regardless of its gender feature F or M. However the functions Pd^F and Pd^M must select respectively a feminine word and a masculine word. All these constraints are achieved in a manifold $\mathbf{Man}(\mathbf{k}\mathbb{H})$. For simplicity we work with a non-elliptic manifold.

The key to the reduction is as follows: if the subject takes a feminine (masculine) word then we block all the masculine (feminine) functions in the cluster. As a consequence at the end of the verbal cluster the predicate function inherits the gender and it take a word with the correct gender. To achieve this we make four constraints:

- for all $\varphi \in \zeta^*$, if $Sb\cdot\varphi$ takes a feminine word then for any masculine function $\alpha \in \zeta^M$, $\alpha\varphi$, is a null locus;
- for all $\varphi \in \zeta^*$, if $Sb\cdot\varphi$ takes a masculine word then for any feminine function $\alpha \in \zeta^F$, $\alpha\varphi$, is a null locus;
- for all $\varphi \in \zeta^*$, if $\beta\cdot\varphi$ is not null, where β is a feminine function, then for any masculine function α we have that $\alpha\beta\varphi$ is null;
- for all $\varphi \in \zeta^*$, if $\beta\cdot\varphi$ is not null, where β is a masculine function, then for any feminine function α we have that $\alpha\beta\varphi$ is null.

All these constraints are definable in $\mathbf{Man}(\mathbf{k}^2\mathbb{H})$ as follows:

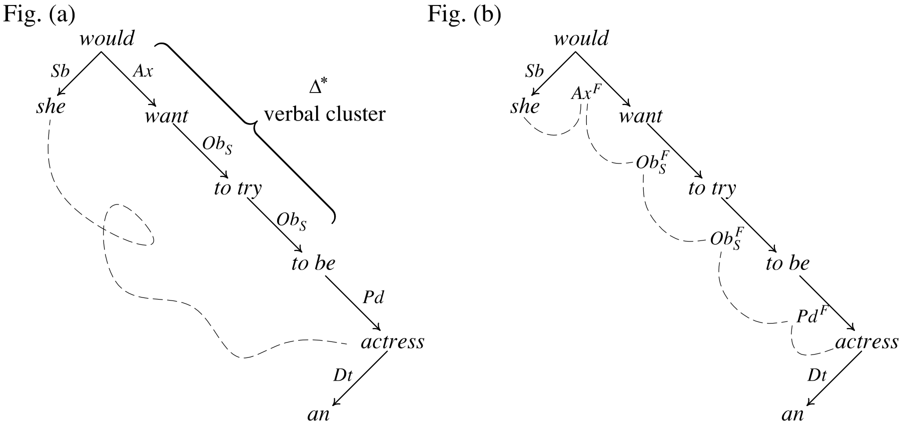
$$\bigcap_{\alpha\in\zeta^M} \mathbf{Synt}\begin{pmatrix} x \in \mathbf{feminine} \to y \approx 0 \\ (Sb, \alpha)(\zeta)_2^* \end{pmatrix},$$

$$\bigcap_{\alpha\in\zeta^F}\mathbf{Synt}\begin{pmatrix}x\in\textbf{masculine}\longrightarrow y\approx 0\\ (Sb,\alpha)(\zeta)_2^*\end{pmatrix},$$

$$\bigcap_{\alpha\in\zeta^M}\bigcap_{\beta\in\zeta^F}\mathbf{Synt}\begin{pmatrix}x\not\approx 0\longrightarrow y\approx 0\\ (\beta,\alpha\beta)(\zeta)_2^*\end{pmatrix},$$

$$\bigcap_{\alpha\in\zeta^F}\bigcap_{\beta\in\zeta^M}\mathbf{Synt}\begin{pmatrix}x\not\approx 0\longrightarrow y\approx 0\\ (\beta,\alpha\beta)(\zeta)_2^*\end{pmatrix}.$$

Intuitively the gender case is transmitted from the subject to the predicate through the verbal cluster, see Fig. (b). Thus a non-local constraint is transformed into a local constraint.



By interposing more clusters, this can be generalized to other long-distance agreements, such as coordination and pied-piping. If we want to subordinate a substructure with this type of agreement inside another of the same kind, then we have to clear the “memory” of the functions, which also can be achieved by the above techniques.

By similar means one can reduce a constraint in $\mathbf{k}^p\mathbb{H}$ to a constraint in $\mathbf{k}^2\mathbb{H}$. In fact we proved that constants $\mathbf{k}^p$ with $p \geq 2$ do not contribute more expressibility than context-freeness, see Corollary 9.6. But such constructions are, under our understanding, structurally inadequate because we believe that syntactic functions must be carriers of syntactic or even argumental information, but not morphological.

Now we reconsider linearizations and see how several constructions can be reduced to the projective case.

Example 37. Reduction of some linearizations to $\mathbf{Lin}(\mathbb{H}\mathbf{k})$. We saw that considering verb clusters leads to interposition of a monoid in the patterns of the arrangements. However we can try to reanalyze the chain of auxiliary verbs as follows. Fig.(a) depicts a wh-fronting construction attending structural facts, while Fig.(b) shows the projective alternative with the arrangement $\zeta^*In_? + 1 + \zeta^*Sb + \zeta^*Ax + \zeta^*Ob$.

Fig. (a)

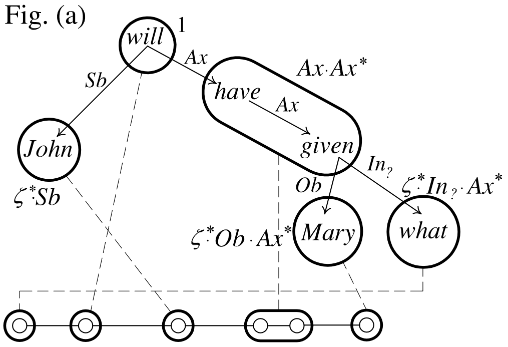
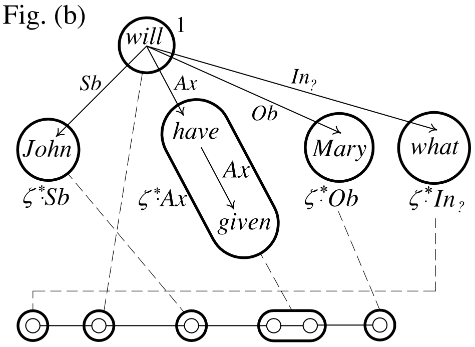


Fig. (b)



Now we only need to enforce a rule making sure that auxiliary functions can just govern other auxiliary functions, while the other verbal functions $Ob, In, At, Ob_?, Ob_!, \ldots$ must be immediate descendants of the root. This is accomplished in a manifold with type $\mathbf{k}^2\mathbb{H}$.

Extraposition constructions can also be reduced to the projective case. We introduce a new function $Nc_\uparrow$ which hangs directly from the verb. We also make the adjunct of time At hang from the root. Fig.(c) shows the linearization obeying structural facts, while Fig.(d) shows the conversion to the projective case, with the arrangement $\zeta^*Sb + 1 + \zeta^*Ax + +\zeta^*Nc_\uparrow + \zeta^*At$.

Fig. (c)

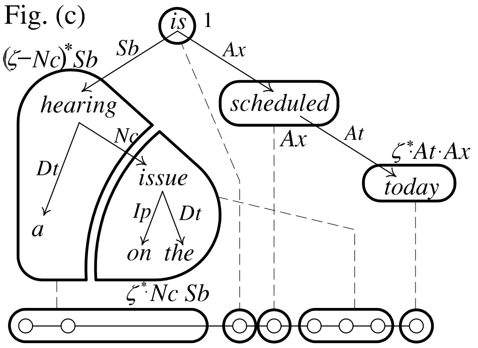
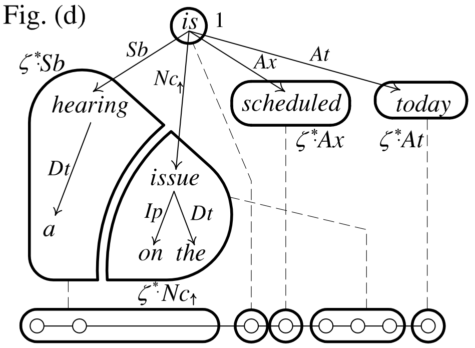


Fig. (d)



Now the constraint on the manifold must enforce that the functions $Nc_\uparrow$ and Nc (which is ordinarily at depth two) must be orthogonal. This is achieved in a manifold with type $\mathbf{k}^2\mathbb{H}$. When the extracted part is at a deeper position p, the rule adopts a type in $\mathbf{k}^p\mathbb{H}$. However by means as in the previous example, this can be reduced to the case $\mathbf{k}^2\mathbb{H}$.

In this last example projectivity is achieved by raising the material from the queue of the chain of the verbs, or other difficult positions, to the level of root children. In fact, some dependency grammarians advocate such options, although they are not the general accepted analyses; see arguments already discussed in §1.5, 6.1 and §11.1.

Summing up, by reanalyzing constructions, by structurally (in a linguistic sense) dubious means, one can reduce manifolds for *non-local* constraints to manifolds in $\mathbf{Man}(\mathbf{k}^2\mathbb{H})$ and linearizations for *movements* and *extrapositions* to projective linearizations in $\mathbf{Lin}(\mathbb{H}\mathbf{k})$. Combining both components we obtain the class $\mathbf{k}^2\mathbb{H}/\mathbb{H}\mathbf{k}$, or in more familiar words, the combinations all these phenomena are reducible to context-free languages.

However as is well-known, certain phenomena cannot be reduced to the context-free case. Firstly, non-semi-linear constructions are not reducible for obvious reasons. Secondly, *cross-serial dependencies* and *respectively* constructions cannot be linearized projectively without changing the type $\mathbf{k}^2\mathbb{H}$ of the manifold.[3]

Overall this leads us to a final conclusion:

Claim 6. Weak capacity. Assuming only semi-linear constructions, natural languages should be described in terms of weak capacity by the class in the bi-hierarhcy $\mathcal{BH}(\mathbf{k}, \mathbb{H}, \mathbb{G})$:

$$\mathsf{NT} \subseteq {}^{\mathbf{k}^2\mathbb{H}}\!/_{\mathbb{G}\mathbf{k}\mathbb{G}}.$$

There are good reasons to believe that this class could be mildly context-sensitive, or at least include mild context-sensitivity. Using the properties proved in Chapter 9 we can observe that the following hold:

- *The class contains all the context-free languages,* $\mathsf{CF} \subseteq \mathbf{k}^2\mathbb{H}/\mathbb{G}\mathbf{k}\mathbb{G}$.

  We only have to see that $\mathsf{CF} = \mathbf{k}^2\mathbb{H}/\mathbb{H}\mathbf{k} = \mathbf{k}^2\mathbb{H}/\mathbb{G}\mathbf{k} \subseteq \mathbf{k}^2\mathbb{H}/\mathbb{G}\mathbf{k}\mathbb{G}$.

- *The class contains all the anti-context-free languages,* $-\mathsf{CF} \subseteq \mathbf{k}^2\mathbb{H}/\mathbb{G}\mathbf{k}\mathbb{G}$.

  Similarly $-\mathsf{CF} = \mathbf{k}^2\mathbb{H}/\mathbf{k}\mathbb{H} = \mathbf{k}^2\mathbb{H}/\mathbf{k}\mathbb{G} \subseteq \mathbf{k}^2\mathbb{H}/\mathbb{G}\mathbf{k}\mathbb{G}$. This means that cross-serial dependencies are possible, and in particular $L_{q\text{-copy}}, L_{q\text{-count}} \in \mathbf{k}^2\mathbb{H}/\mathbb{G}\mathbf{k}\mathbb{G}$, for any $q \geq 0$.

- *The languages in* $\mathbf{k}^2\mathbb{H}/\mathbb{G}\mathbf{k}\mathbb{G}$ *are semilinear.*

  We learned that semi-linearity only depends on the semi-linearity of the manifold, cf. Theorem 9.12.

- $\mathbf{k}^2\mathbb{H}/\mathbb{G}\mathbf{k}\mathbb{G}$ *is not generable by* TAG, *p*-LCFRS *nor p*-MCFG, *for some fixed p.*

  This is a consequence of the fact that $L_{q\text{-copy}}, L_{q\text{-count}} \in \mathbf{k}^2\mathbb{H}/\mathbb{G}\mathbf{k}\mathbb{G}$, for any $q \geq 0$ and the pumping lemma (Seki et al., 1991) for multiple context-free grammars. There is the possibility that languages given by general MCFG be in this class. We recall that these formalisms are linguistically relevant since they are equivalent to MG.

[3] We saw in §8.2 that the copy language can be obtained projectively in $\mathbb{H}\mathbf{k}^2/\mathbb{H}\mathbf{k}$ but this construct implies reversing the type of the manifold. Indeed the bi-simmetry theorem states that $\mathbb{H}\mathbf{k}^2/\mathbb{H}\mathbf{k} = \mathbf{k}^2\mathbb{H}/\mathbf{k}\mathbb{H}$, but $\mathsf{CF} = \mathbf{k}^2\mathbb{H}/\mathbb{H}\mathbf{k} \neq \mathbb{H}\mathbf{k}^2/\mathbb{H}\mathbf{k} = -\mathsf{CF}$.

- *Syntactic structures of* $\mathbf{k}^2\mathbb{H}/\mathbb{G}\mathbf{k}\mathbb{G}$ *are local and linear-time decidable.*

  We saw this in Theorem 4.10 and Theorem 4.13. This is linguistically relevant because it is believed that syntactic structures of a formalism must be regular (as in MSOL based grammars, MTS and MCS formalisms). Locality is an even more restrictive condition than regularity.

- *Languages in* $\mathbf{k}^2\mathbb{H}/\mathbb{G}\mathbf{k}\mathbb{G}$ *are decidable.*

  By Theorem 7.8 and the above point the class is decidable.

- $\mathbf{k}^2\mathbb{H}/\mathbb{G}\mathbf{k}\mathbb{G}$ *is strictly less expressive than other formalisms like unrestricted* HPSG *or* LFG *which are Turing complete.*

  This is a direct consequence of the decidability. Indeed the class $\mathbf{k}^2\mathbb{H}/\mathbb{G}\mathbf{k}\mathbb{G}$ is semi-linear.

These points mark a clear agenda for future work. If we can show a poly-time parsing algorithm for this class of grammars, then it would be known to be entirely mildly context-sensitive, and we could make further comparisons with other well-known formalisms.

Finally, what about non-semi-linear constructions? In this case we would have to expand the numerator to incorporate the types $\mathbf{k}^2\mathbb{G}^2\mathbf{k}$ and $\mathbb{H}\mathbf{k}^2\mathbb{H}$. It remains a question for the future which is the most adequate class and what is its weak capacity.

12.2 Towards Semantics: Transposition Arrangements

There is a second future direction to develop the ideas shown up until here. Consider the following phrases where the parentheses indicate how the order of the complements must be taken in a semantic reading.

(68) a. last (Picasso sold).
b. (last Picasso) sold.

The first reading is the most common. The lexical piece *last* can be regarded as an operator which selects the last element of a list of elements chronologically ordered. So this noun phrase is selecting the last artwork from the list of the Picasso's sold. In contrast, the second noun phrase is identifying the last artwork of Picasso, that was sold.[4]

Nevertheless dependency trees are not able to differentiate the two possibilities, because there is only one dependency tree for both phrases. Nor can the linearization by arrangements distinguish the two cases, because arrangements write all of the elements of the phrase simultaneously.

[4] Picasso painted his last work in 1973; in 2015 an earlier Picasso was sold.

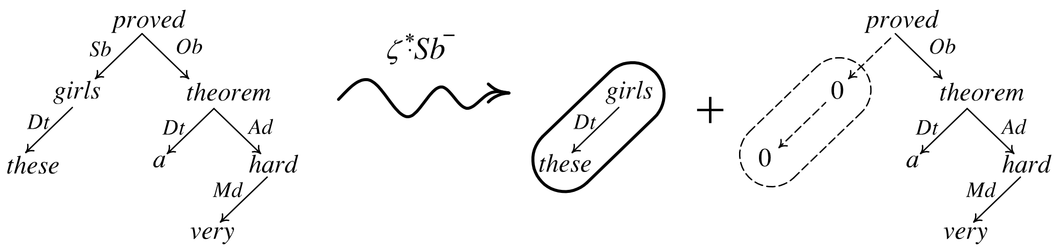


Figure 12.1: an example of left transposition given by $\zeta^* \cdot Sb^-$.

In this relation we can try to linearize substructures one-by-one instead of linearizing the whole phrase in one go. Thus we define:

Definition 12.1. Given a pattern Γ, we call a *right transposition* the expression in **Synt***:

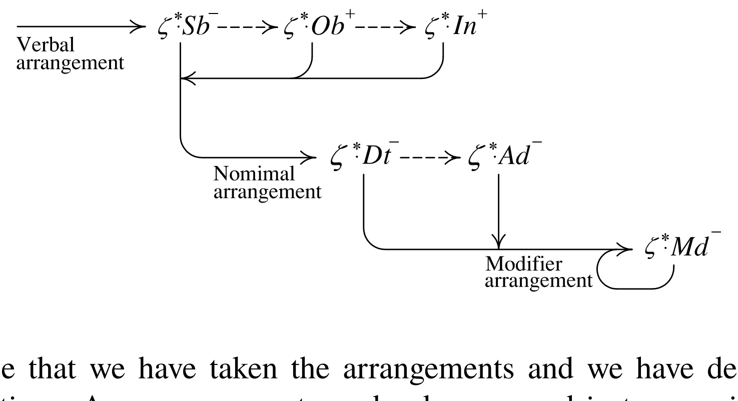
$$S\Gamma^{\mathsf{C}} + S_\Gamma,$$

where we recall that S_Γ is the induced syntagma, $\cdot^{\mathsf{C}}$ is the complement $\Gamma^{\mathsf{C}} = \zeta^* \setminus \Gamma$. So $S\Gamma^{\mathsf{C}}$ is the syntagma in which we make null all the loci in Γ. Similarly we define a *left transposition*:

$$S_\Gamma + S\Gamma^{\mathsf{C}}.$$

See an example of left transposition $(\zeta^* \cdot Sb)^-$ in Fig. 12.1. For comfort we omit parentheses, $\zeta^* \cdot Sb^- = (\zeta^* \cdot Sb)^-$.

Now we can translate the notion of system of arrangements to *system of transpositions*. The underlying idea is the same. We apply a transposition which gives a string of two syntagmata (instead of a string of several syntagma as in the case of arrangements). Then we have to indicate how the linearization must be continued for each one of both syntagmata. We use a continuous line for the induced syntagmata S_Γ and a dashed line for the complement $S\Gamma^{\mathsf{C}}$. Now we can rewrite the canonical system of arrangements which we reviewed in the last chapter, §11.3, as a system of transpositions:



Notice that we have taken the arrangements and we have decomposed them in transpositions. Any arrangement can be decomposed in transpositions, although the

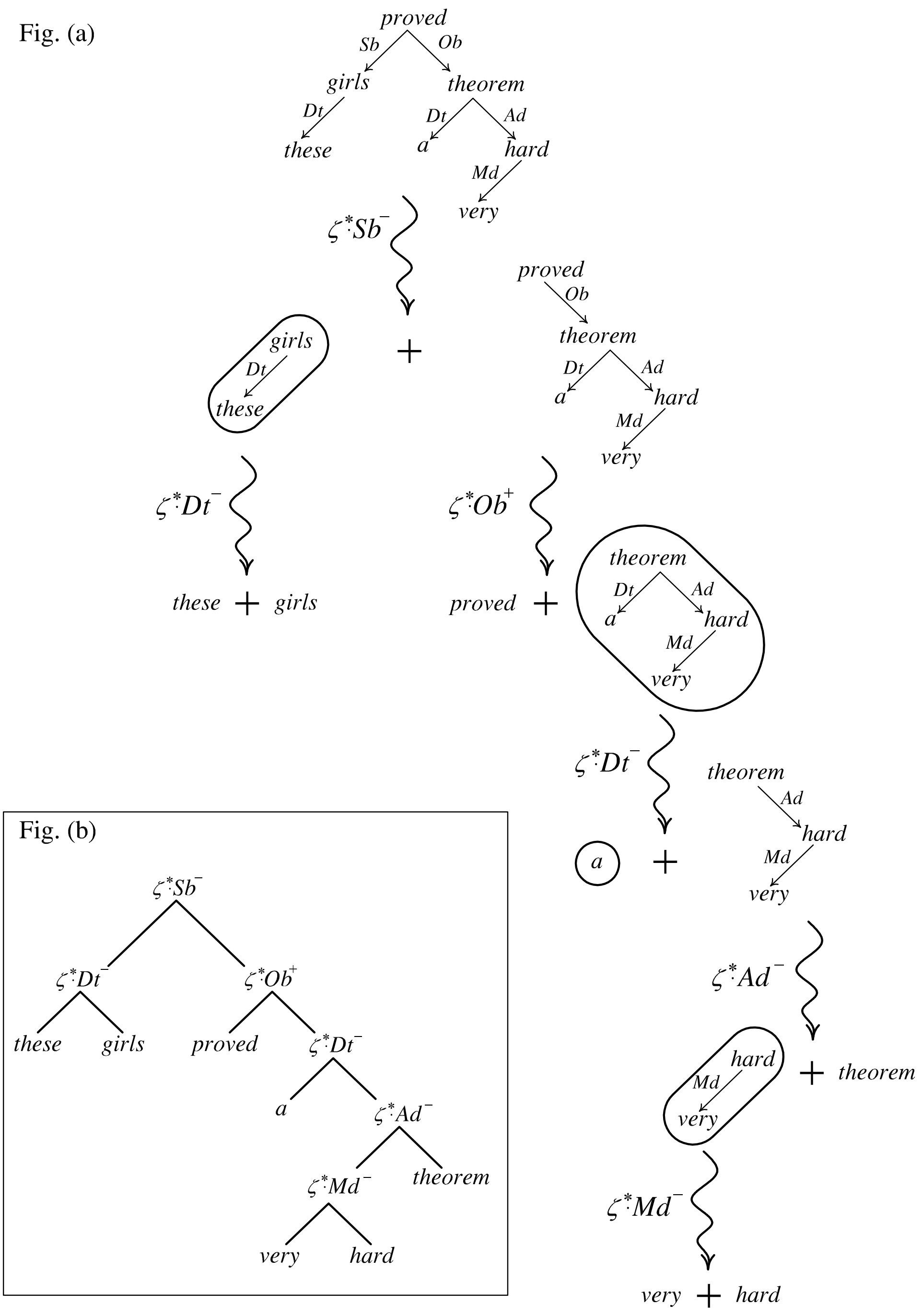


Figure 12.2: (a) linearization by transpositions. (b) schematization in a binary tree of the same linearization.

decomposition is not unique. For example, when signs are different we can shift transpositions; we can first transpose $\zeta^* \cdot Sb^-$ and then $\zeta^* \cdot Ob^+$, or first we can transpose $\zeta^* \cdot Ob^+$ and then $\zeta^* \cdot Sb^-$; both yields the same result, but in general two transpositions do not commute.

Let us show an example of linearization by transpositions; see Fig. 12.2(a). Interestingly the process can be summarized in a binary tree, see Fig. 12.2(b). This points towards a bridge between an analysis of dependencies and other formalisms based in analysis in constituents which assume a binary branching such as the minimalist program, or categorial grammar.

As we have commented, in general two transpositions do not commute, whereby the order is important. The application of a transposition places the material S_Γ in the periphery. Thus a succession of transpositions (following dashed lines) performs a linearization by layers; from the most external to the most internal which yields an scheme of nested constituents:

$$(\bullet_1(\bullet_2(\bullet_3 \cdots (\bullet_{n-1}(\bullet_n)\bullet_{n-1}) \cdots \bullet_3)\bullet_2)\bullet_1)$$

where every $\bullet_i$ is realized only in one of the two sides, as for example in the noun phrase:

$$\begin{array}{l} (\ \circ\ (\ \text{an}\ (\ \text{old}\ (\ \text{lady}\ )\ \circ\ )\ \circ\ )\ \text{from Berlin}\ ) \\ (\ \circ\ (\ \ \bullet\ \ (\ \ \bullet\ \ (\ \ \bullet\ \ )\ \circ\ )\ \circ\ )\ \quad \bullet \quad\ ) \end{array}$$

No doubt a well studied system of transpositions for natural language would be more intricate that above (with multiplicity of paths for a same linearization) and it would reveal that some constructions can be linearized in different ways, whereby a sentence could exhibit a non-unique parenthetic structure.[5]

There is a controversy regarding binary branching and n-branching. On the one hand, Binary branching agrees with the intuition that words are combined semantically pairwise and is the minimal combinatorial strategy. On the other hand the traditional grammarians decomposed the sentence in classical grammatical categories (verb phrase, noun phrase and others) which assumes n-branching.

In our framework linearizations by arrangements conform with the intuition of n-branching. In fact we have planned systems of arrangements through basic arrangements (verbal, nominal, modifier, etc.) which correspond to the classical phrases. As we said above transpositions would seem to agree with a binary branching. In both cases

[5] Evidence in favor of transpositions is the distribution by layers of the verbal morphemes as a universal principle, (Mairal and Gil, 2004). These morphemes can be situated at the left or right hand (depending on the language) of the root, but always distributed in a certain universal order of layers from the verbal lexeme to the peripheral. This can be implemented by a system of transpositions with functions introducing the several morphemes. What is universal is the order of the transpositions, while the sign of each transposition varies between languages.

branching appears as the trace of the linearizations, i.e. we do not need to introduce parenthetical structures over and above the syntagmata.[6]

Reexamining the system of transpositions above we observe that the structure of the arrangements are still visible (in fact we have still marked the old denominations of the arrangements: verbal, nominal and modifier). This could signify a convergent solution which combines naturally both perspectives. However all this constitutes material for future work.

12.3 Closing

Algebraic dependency grammar, we believe, proportions a new form of understanding natural languages and formal languages. In particular we have seen how certain natural constructions are responsible for the distribution of constants and patterns in their structural description. We have argued that this algebraic complexity is bounded for natural languages.

Nevertheless in our opinion this bound does not arise from a mathematical limitation of expressibility, rather an empirical limitation of the brain. Perhaps this bound is not completely fixed. Perhaps primitive languages lacked certain sophisticated syntactic constructions such as pied-piping, auxiliary verbs, and others. If we could recover data from ancient languages, we might observe that new constructions have been added to these in the course of time. We envisage the possibility of seeing language evolution as a succession of ascending stages in a bi-hierarchy. Of course this prospect belongs to the field of paleo-linguistics which is far from our competence and the goals of this thesis.

Regarding the organization of formal languages, we have introduced the notion of anti-languages. In relation to this let us quote the definition of *chirality* which was given by the physician W.T. Kelvin: “I call any geometrical figure, or group of points, chiral, and say it has chirality, if its image in a plane mirror, ideally realized, cannot be brought to coincide with itself”, (Kelvin, 1904). Chirality is a very useful concept in chemistry, physics and biology.[7] Each chiral object has a corresponding symmetric object. These two symmetric realizations are called *enantiomeric* forms. By contrast, *achiral* objects do not enjoy these two forms. So, a shoe is a chiral object which has two enantiomeric forms: the right shoe and the left shoe, while a sock is an achiral object!

Under this geometrical denomination, up to metaphors, we have shown that the class of context-free languages is indeed a chiral object, with the enantiomeric forms CF and $-\mathsf{CF}$. Then, a final interesting, or at least curious, question arises when we consider the corpus of linguistic phenomena. We know that the vast majority of constructions

[6] See (Kahane, 1997) which introduces “bubbles” in order to obtain a finer structure in the dependency tree. In our case, the trace of the linearization (by arrangements or by transpositions) can be represented also by bubbles.

[7] See for example (Cronin and Reisse, 2005).

are context-free. One can ask why there is this prevalence of a handedness, or in other words, why this abundance of context-freeness in natural language?

There seems to be a rupture of symmetry in favor of one of the enantiomeric classes of languages. But this is not exclusive to natural languages. Other physical and biological systems exhibits of this kind of rupture. Consider for example, amongst many others, the prevalence of right handedness in humans or the prevalence of a clockwise handedness in the spiral of the shells in the majority of the species of mollusks.

APPENDICES

A

Aspects of Coordination in Dependency Grammar

A.1 Criteria of Adequacy

We devote the following pages to some approaches to coordination in dependency grammar, and to providing our own proposal. Before reviewing an inventory of proposals for coordination we want to present the possible features and discuss the validity of criteria which coordination should satisfy.[1]

> **Iterativity and recursivity.** Kahane (1997) highlights two features of coordination constructions: *iterativity*, the number of conjuncts elements is not bounded; and *recursivity*, different coordinators can be combined; as in the sentences (a) and (b) respectively:
>
> (69) a. John, Mary, . . . , Joseph and Bill bought the same car.
> b. Peter invited either John and Mary or Bill.
>
> Accordingly Kahane (1997) says that "the recursivity is linguistically limited to one step and must be well marked (by special words, such as either, or prosody)." An example of necessity of prosody is:
>
> (70) For the next game we need a couple. The possibilities are: John and Mary, Lucy and Bill, or George and Sam.
>
> Adequate pauses and stresses suggest the reading (*John and Mary*), (*Lucy and Bill*), *or* (*George and Sam*).[2]

[1] In the sequel parentheses accompanying an ellipsis ∅, as in ∅(x), indicate that x is the gapped element.

[2] We do not know how is interpreted the limitation "to one step" of Kahane, although we agree that some constraints must be imposed. For instance, if we suppose that there is no prosodic marking and that iterativity and recursivity are not limited then the number of readings of a sentence as: *John and Mary*

Symmetric coordination vs. non-symmetric coordination. Coordination can exhibit directionality. Consider for example the sentence:[3]

(71) Hans slipped into his jacket and left. / $^{??}$ Hans left and slipped into his jacket.

In this case the coordinator implies a time order: "*A* and *B*" means that *A* occurs before *B*, and on some occasions, there is even a logical dependency:

(72) She slept and had sweet dreams. / $^{??}$ She had sweet dreams and slept.

However there are plenty of cases where the commutation is possible:

(73) John and Mary went camping = Mary and John went camping.

In several approaches symmetry is emphasized over asymmetry. However natural language exhibit both possibilities. Thus, on the one hand approaches that only allow symmetric coordination are insufficient. On the other hand, we could think similarly that approaches that only exhibit asymmetry are equally inadequate.

However let us consider the following analogy extracted from propositional calculus. The expression of propositional logic, $p \wedge q$, is in fact a string in $\{p, q, \wedge\}^*$ and, strictly speaking, we have different strings $p \wedge q \neq q \wedge p$. When we interpret the symbols p, q as certain propositions which are truth-valued then the symbol $\wedge$ becomes an operator which involves the value of the propositions and yields a result according its operands. Thus the symmetry of the formula arises from the interpretation $\wedge$ as an operator of truth-values which is usually expressed through equivalence $p \wedge q \equiv q \wedge p$.[4] In short, also in dependency grammar symmetry could be an interpretative rather than structural feature.

Pure dependency vs. extended structures. Some approaches claim that coordination is not possible to capture with dependency trees only. So the structure must be generalized or enriched. However we consider this a last resort solution. Since our framework comprises dependency trees which work well for a lot of natural constructions we should preserve the discipline of tree-shape dependency as far as we can go.

or Lucy and Peter, or more in general a_1 *and* b_1 *or* a_2 *and* b_2 *or* $\cdots$ a_n *or* b_n blows up. The number of parenthesizations is given by the catalan numbers C_{2n-1} and recall that $C_n = \frac{(2n)!}{(n+1)!n!} \sim \frac{4^n}{\sqrt{\pi n^3}}$, see for example (Davis, 2006). Some of the configurations of parentheses yield the same interpretation, but even with this, one can prove that the number of different interpretations is at least 2^{n-1}.

[3] Examples are from (Lison, 2006).

[4] The symbol ≡ here means equal for any values on p and q.

Tree-shape vs. non-tree-shape. Supposing pure dependencies based formalism we can distinguish tree-shaped vs. non-tree-shaped structures. Co-headed (or co-governed) words are often used in a semantic representation which in addition permits represent the coordination, *fred peeled and ate it*, as two-rooted (and hence non-tree-shape) dependency structure.[5] We will show that a purely tree-shape based representation also can be adequate for coordination.

Functionality vs. non-functionality. A functional (or deterministic) tree precludes the possibility of two arrows labeled with the same syntactic function governed by a same node.

Some approaches deny functionality, since one can then take advantage of the possibility of non-deterministic configurations in order to represent groups of conjuncts. We will show, however, other approaches where functionality is capable of representing coordination.

Explaining gapping. As far as we know all the modalities accept the possibility of gapping. Gapping arises in sentences as:

(74) a. Mary loves Bill and Ann loves John.
b. Mary loves Bill and Ann, John.

The coma is supplanting the verb: *Mary loves Bill and Ann* ∅(*loves*) *John*. However the following sentence is ungrammatical.

(75) *Mary ∅(loves) Bill and Ann loves John.

An adequate system of representation of coordination should permit explaining what elements can be gapped.[6] We call this feature *explaining gapping*.

Others: equal status of the conjuncts, the position of the coordinators . . . It is well-known that in general the conjuncts must be similar. This explains the possible ungrammaticality of the sentence:

(76) ??John and driving cars are nice.

This is already established in traditional grammars: “Since coordination is a relation between elements of equal status, they must be syntactically alike”, *The Cambridge Grammar of the English Language*, (Huddleston et al., 2002, pg. 1290).

The first desideratum in placing the coordinator is preserving projectivity. Popel et al. (2013) classify the possible coordinator positions comprehending several variants and subvariants.

[5] Example and details in (Temperley, 2005).

[6] In our opinion this feature is related with the above point *asymmetry*. We see later the details.

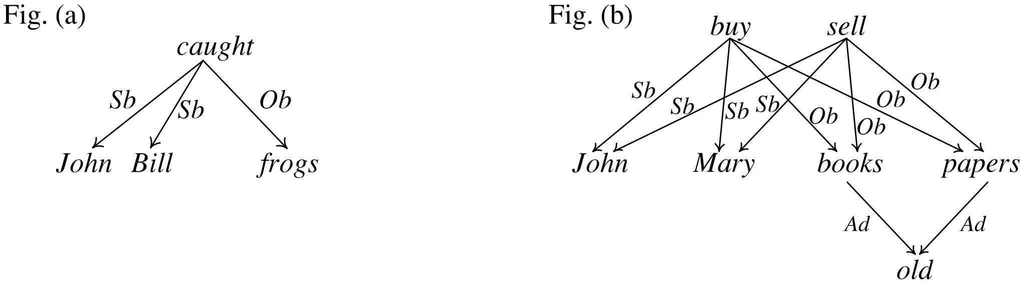


Figure A.1: (a) and (b) bare style of coordination.

A.2 A short Inventory of Coordination Styles

We list a representative though not exhaustive inventory including the style used in the thesis. Table A.1 at the end collects the features of each modality.

The bare style. Suppose that we allow non-functional analysis, having a configuration as in Fig. A.1(a). We call such a situation a *non-deterministic configuration*. One is tempted to interpret this as that the words *John* and *Bill* form a coordinated group since they are governed by the same function, *Sb*. Since the most prominent coordination in natural language is mediated by the *and* coordinator we can think of the nodes as forming a conjunctive group. So this style of coordination is a rather early linguistic application of non-deterministic configurations. Another feature of this style is that we can allow co-governed nodes to represent situations such in Fig. A.1(b).[7] But this style lacks the possibility of using different coordinators.

Analyses *à la* Tesnière. The modality commented just above has the inconvenience of capturing only one kind of coordination. Tesnière's solution was grouping and labeling the non-deterministic configurations in order to differentiate them by adding a *transversal* relation on the tree, see Fig. A.2(a) and (b). This permits diversity of coordinators, but, since Tesnière did not make formal this extra relation, we have multiple interpretations.

We deduce from the pictures in the *Eléments de syntaxe structurale* that the relation is symmetric, anti-reflexive and even anti-transitive. More even, it seems that the transitive closure of this relation must coincide with a whole non-deterministic configuration. We do not know if Tesnière would have tolerated combining recursively different kinds of coordinators.

Extended bare style. Another different way to solve the problem of which kind of coordination operates in a non-deterministic configuration consists in considering the

[7] This feature is also called *muli-head*; see (Hudson, 1990; Temperley, 2005).

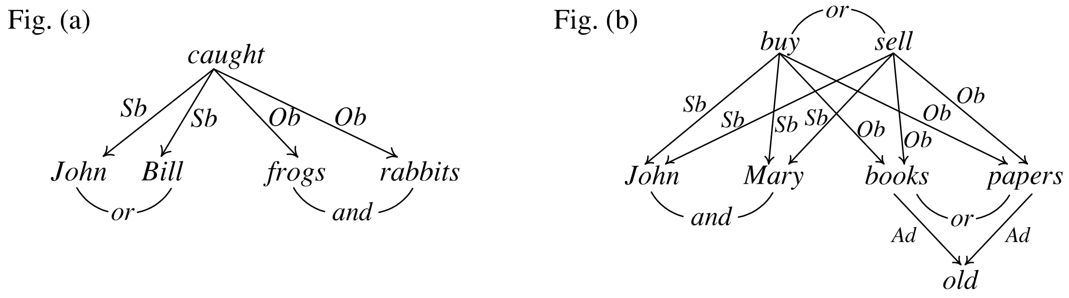


Figure A.2: (a) and (b) Tesnière's style of coordination.

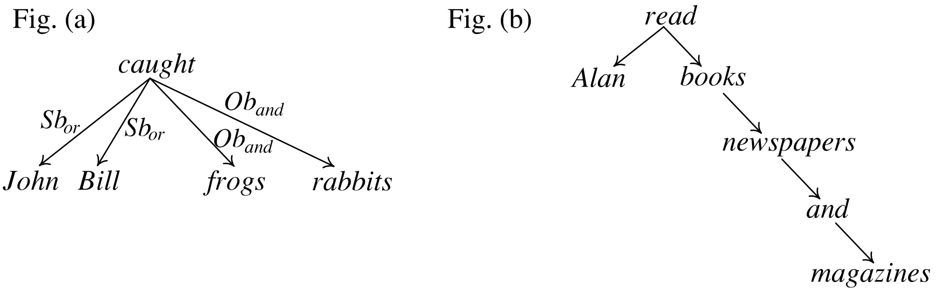


Figure A.3: (a) extended bare style and (b) Mel'čuk style of coordination.

Cartesian product of the set of syntactic roles by the set of coordinators. For example we notate $Sb_{and} = (Sb, and)$, see Fig. A.3(a). Now the second component of the syntactic role informs as to the kind of coordinator. This style works very similarly to that of Tesnière, but no extra relation is needed: it is purely dependency based. This style is illustrative; we do not know any approach which uses it.

Analysis *à la* Mel'čuk. The approach of Mel'čuk is purist: trees should be sufficient. No extra relation is needed. Its solution consists in concatenating the conjuncts (as many as we want) by a function, *coord*. Fig. A.3(b) shows an example adapted from Melčuk (2009).

The Mel'čuk style allows several variants. Regarding the position of the coordinators we enjoy different options: Fig. A.4(a), (b) and (c). In the cases (b) and (c) we introduce the coordinator through a syntactic function (*Ic = Introduction of coordinator*). These options are amongst the most reasonable in order to preserve projectivity.

However the original style of Mel'čuk interposes the coordinator between conjuncts which could violate the criterion of equality of status, see Fig. A.4(a). For this reason we have chosen the option (c) with the coordinator at the bottom (we call this **style of this thesis**, see last point).

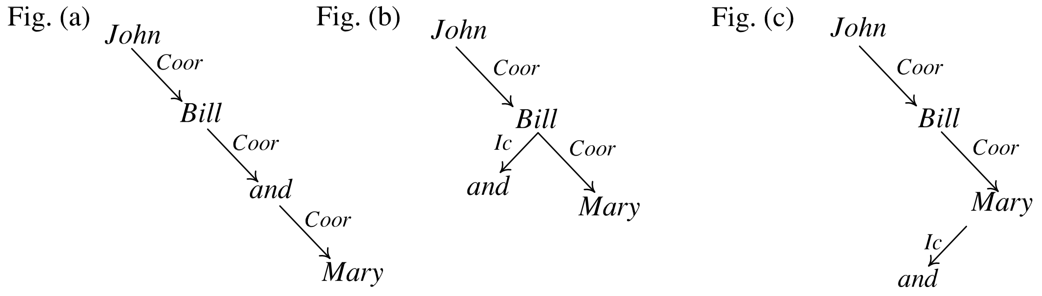


Figure A.4: (a) and (b) variants of analyses *a la* Mel'čuk according to the position of the coordinator; (c) the style chosen in this thesis.

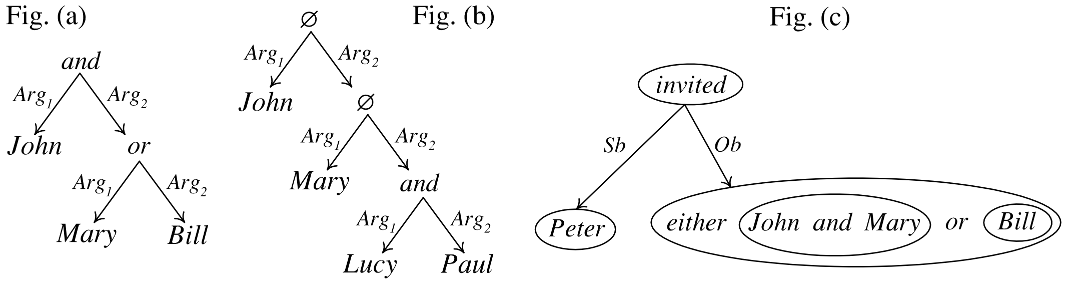


Figure A.5: (a) and (b) arithmetical style; (c) a bubble tree.

Analyses arithmetical-tree style. Another, very different, style of coordination is based in understanding coordinators as arithmetical operations, as $+(x, y)$ or $\times(x, y)$. So using operations **and**(x, y), **or**(x, y) we can construct a representation for the phrase *John and Mary or Bill* as:

$$\textbf{and}(\textit{John}, \textbf{or}(\textit{Mary, Bill})).$$

This can be translated to dependency trees by using two functions, say Arg_1, Arg_2, see Fig. A.5(a). If we want to iterate several elements, e.g. *Bill loves John, Mary, Lucy and Paul*, then a gap is needed to avoid the repetition of the coordinator, see Fig. A.5(b). The main properties of this style are that it is functional and it allows asymmetric coordination since the functions Arg_1, Arg_2 are different although they are placed at the same level in the tree.

Bubble trees. Very informally "(...) bubble tree are trees whose nodes are bubbles which in turn contain sub-bubbles linked to other bubbles and so on", as Kahane (1997) presented them. Given the rich interplay of trees and bubbles, this mechanism subsumes several kinds of structures over and above coordination. Regarding coordination, bubble trees allow its capture in a robust and unambiguously way. A coordination is a bubble enveloping the conjuncts which can be in turn other coordinating bubbles, as in Fig. A.5(c).

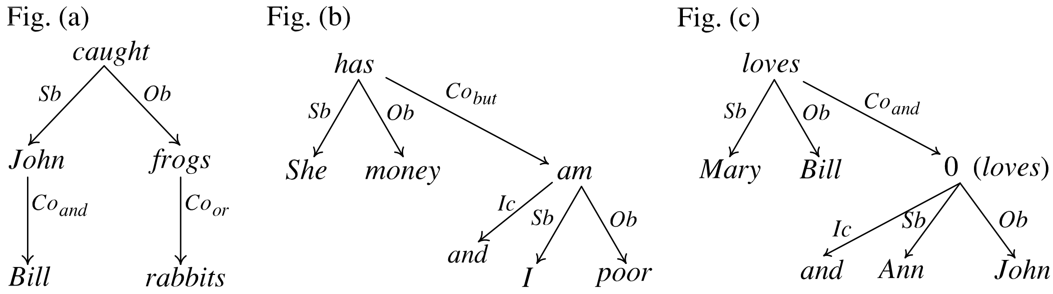


Figure A.6: the style of this thesis as a Mel'čuk variant; (a) and (b) show constructions which take advantage of differentiation of coordination by the functions; (c) gapping.

These structures are symmetric and not pure dependency structures. Although bubble trees allow gapping, which permit representation of several tricky constructions,[8] it is not clear that this style can account for an explanation of gapping. The reason is the symmetry: since the conjuncts are symmetric, one cannot differentiate which of the conjuncts must be gapped.

The Style of this thesis. We examine the Mel'čuk variant with the coordinator at the bottom. It is functional (and thus tree-shaped and purely dependency based) and it is asymmetric. Of course it enjoys iterativity. In addition since we place the coordinator at the bottom this preserve the equality of status of the conjuncts (in the sense that the coordination function chains always conjuncts, not other elements), and, in particular, the coordinator does not dominate conjuncts.

Instead of a general function for coordination, *coord*, we can differentiate the kind of coordination according the function such as *copulative coordination, disjunctive coordination, adversative coordination*, which we notate Co_{and}, Co_{or} or Co_{but}. See Fig. A.6(a). Differentiating the syntactic function of the coordinator in itself permits us to use, for example, the coordinator *and* in an adversative meaning as in the sentence: *Mary has all the money and*(but) *I am poor*, see Fig. A.6(b).

This last shows that diversity of coordinators is possible. Fig. A.7 shows that indeed recursivity is also possible. Fig. A.7(a) depicts the semantic representation of the phrase: *John and Mary, Lucy and Bill, or George and Sam.* We simply represent the syntactic functions Co_{and}, Co_{or}; in the syntactic representation, Fig. A.7(b), we introduce the coordinators.

We notice that this allows gapping, see Fig. A.6(c) which shows the analysis of the sentence: *Mary loves Bill and Ann, John.* Our interpretation is that in a semantic analysis this sentence must specify the elliptic verb (*loves*). Then we can assume a rudimentary principle on the interface semantic-syntax:

[8] see (Kahane, 1997; Lison, 2006) for more details.

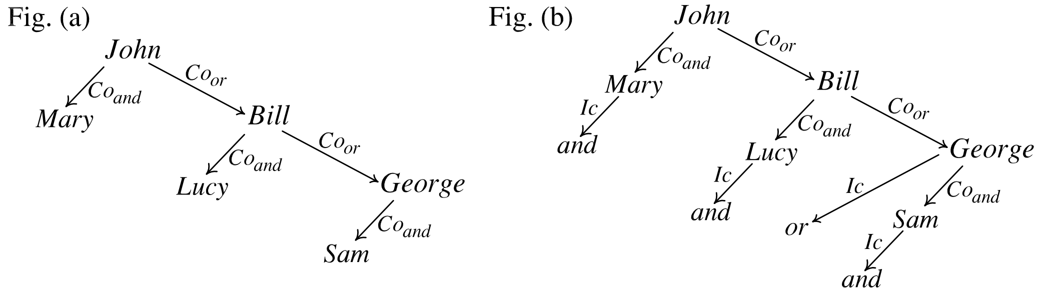


Figure A.7: recursivity in the style of this thesis; (a) semantic representation and (b) syntactic representation.

Claim 7. Syntactic representation can preclude the redundancy of words whereby the repeated words are gapped. Let us establish this with a bit more precision; in fact this principle seem to fit well in terms of patterns. Let *Co* be a syntactic function of coordination, for example $Co_{and}, Co_{or}, \ldots$. The principle states that the locus which can be gapped is of the form $\varphi\cdot Co^n\cdot\psi$ with $n > 1$ and $\varphi, \psi \in \zeta^*$, i.e. the pattern $\zeta^*\cdot Co\cdot Co^*\cdot\zeta^*$.

This question is closely connected with co-reference and anaphora, whereby the above principle is affected by similar phenomenology.

Using this principle of non-redundancy we can exhibit ambiguity in a natural fashion. Consider the phrase: *old books and papers*. Supposing that a speaker has in mind *old books and (old) papers*. In a semantic analysis we propose analysis as in Fig. A.8(a.i). By the above principle the word *old* is gapped at the level of syntax, see Fig. A.8(b.i). Now we suppose that a speaker has in mind *(old books) and papers*, i.e. *papers* has no adjective. Its semantic representation would be Fig. A.8(b.ii). Since now there is no redundancy, there is nothing to delete at the syntactic level. Both situations yields the same syntactic representation. Thus when a hearer tries to reverse the process, he cannot determine whether the speaker wanted to say *old books and old papers* or simply *(old books) and papers*. Finally, if the speaker has in mind *books and old papers*, Fig. A.8(a.iii), the analysis becomes as in A.8(a.ii) and here there is no ambiguity.

We can justify more complex analyses in the same way. Fig. A.9(a) shows an interpretation at the semantic level of the sentence: *John and Mary buy and sell old books and papers*. Fig. A.9(b) shows the gapping and Fig. A.9(d) shows the syntactic level with the introduction of the coordinators. A couple of more of tricky cases follow in Fig. A.10 and Fig. A.11.

Although we claim that the grammaticality of coordination constructions can be explained just with a syntactic component and a word-order component (that is, with an algebraic dependency grammar), we see that a semantic component helps to understand the why of reviewed cases. However the incorporation of the semantic module constitutes matter for future research.

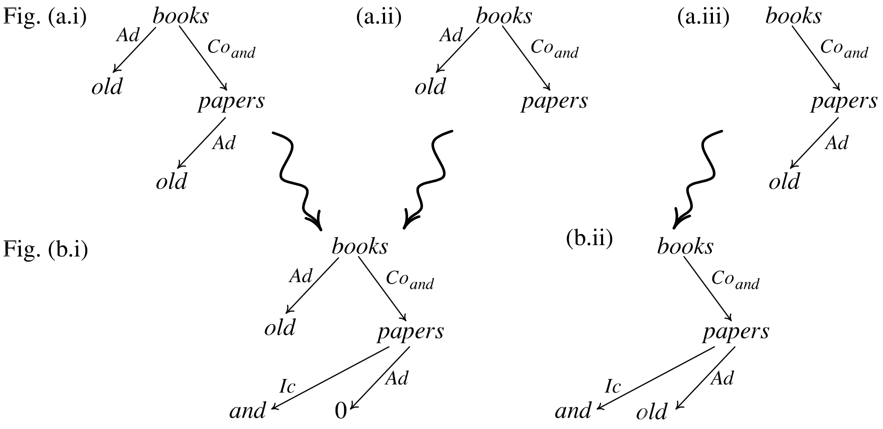


Figure A.8: (a.i) and (a.ii) semantic analyses of *old books and papers*. (a.iii) semantic analysis of *books and old papers*. (b.i) syntactic analysis of *old books and papers*. (b.ii) syntactic analysis of *books and old papers*.

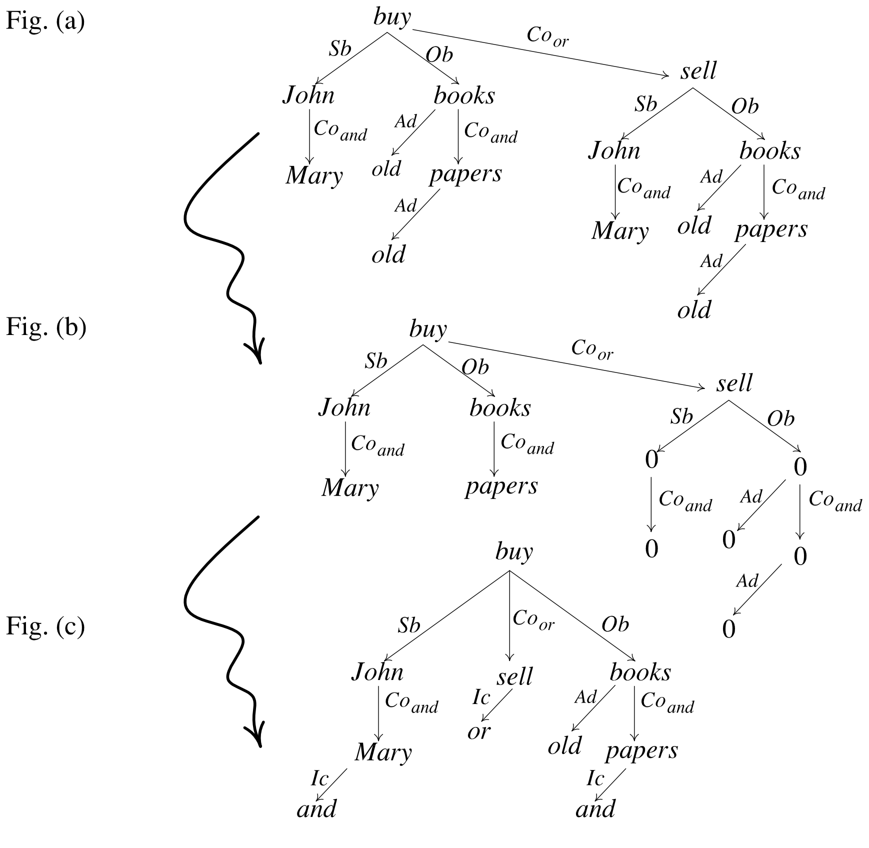


Figure A.9: semantic and syntactic analyses in the style of this thesis for the sentence *John and Mary buy or sell old books and papers*: (a) semantic analysis; (b) gapping; (c) syntactic analysis.

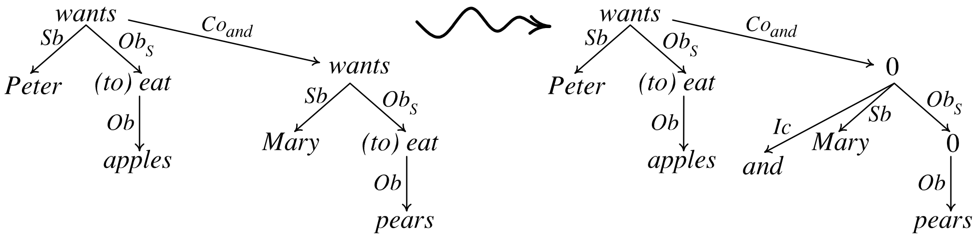


Figure A.10: semantic and syntactic analyses in the style of this thesis of the sentence *Peter wants to eat apples and Mary, pears.*

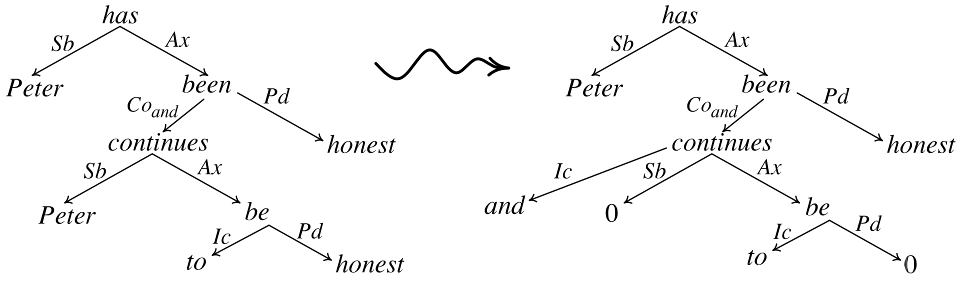


Figure A.11: semantic and syntactic analyses in the style of this thesis of the sentence *Peter has been and continues to be honest.*

Coordination Style	**Iterativity**	**Recursivity**	**Asymmetry**	**Pure Depend. based**	**Tree-shape**	**Functionality**	**Explaining gapping**
The bare style	✓	×	×	✓	×	×	×
a la Tesnière	✓	−	×	×	×	×	×
Extended bare style	✓	−	×	✓	×	×	×
a la Mel'čuk	✓	−	✓	✓	✓	✓	−
Arithmetical trees	✓	✓	✓	✓	✓	✓	−
Bubble trees	✓	✓	×	×	×	×	×
Style of this thesis	✓	✓	✓	✓	✓	✓	✓

Table A.1: styles of coordination and their features. The symbol − means not applied, not described or unknown.

B

Syntactic Functions and Isotopes

B.1 Notation

We present a more complete table of syntactic functions. For each function we have proposed at least one example in English, Catalan and Spanish, provided they exist. Some functions only exist in some of these languages. Not all the functions have corresponding isotopes; this varies according to the language as well. Tables at the end summarize functions for each language, see §B.8.

Regarding notation, when the syntactic function does not exist (or we could not find an example) we mark — —. However when a whole group of functions does not exist we have removed it.

When the syntactic function exists for certain language we indicate first the language, **(Eng)** , **(Cat)** or **(Spa)** , followed by the example or examples. The continuous underline word corresponds to the word governed by the function in question and the dashed underline corresponds to the governor (or head). The brackets indicate the limits of the domination of the function, which correspond to the traditional constituent. For example, for the function *Ob* we display the sentence: *John gave Mary [some money].* In this case, the main verb *gave* governs *money*, and thus the example establishes the dependence:

$$\textit{gave} \xrightarrow{Ob} \textit{money}$$

The scope of the domination of the object is *[some money]*. In some occasions the constituent is discontinuous and then we mark the segments: *[Where] do you come [from]?*. Of course this depends on the style of analysis adapted (see §1.5).

B.2 Primitive Functions

Verbal Functions

Unit Function

1 — *Unit function*. This is not a proper function algebraically, i.e.: 1 is the identity element of the monoid ζ^*, but it is not a generator, $1 \notin \zeta$. However it corresponds to the root of the syntagma and it has linguistic meaning. It always take a finite verb. For example: **(Eng)** *[the bird cracked the window].* **(Cat)** *[L'ocell trencà la finestra].* **(Spa)** *[El pajaro rompió la ventana].*

Canonical Functions

Sb — *Subject function*. This identifies the main argument in the sentence. The subject function always selects a noun. It is who/what makes the action given by the verb, e.g. **(Eng)** *[The black bird] cracked the window. [John] gave Mary some money.* **(Cat)** *[l'ocell negre] trencà la finestra. [El Joan] donà diners a la Maria.* **(Spa)** *[El pajaro negro] rompió la ventana. [Juan] dio dinero a Maria.*

Ob — *Object*. This identifies the second main argument.[1] The object function always selects a noun. **(Eng)** *The black bird smashed [the window]. John gave Mary [some money].* **(Cat)** *[l'ocell negre trencà [la finestra] El joan donà [diners] a la Maria.* **(Spa)** *El pajaro negro rompió [la ventana]. Juan dio [dinero] a Maria.*

In — *Indirect object*. This identifies the third main argument.[2] The indirect object function always selects a noun. **(Eng)** *John gave [Mary] some money.* **(Cat)** *El Joan donà diners [a la Maria].* **(Spa)** *Juan dio dinero [a Maria].*

Pd — *Predicate*. The object of a copulative verb like *to be* or *to look*. It is also called *attribute*. **(Eng)** *Mary looks [tired].* **(Cat)** *La Maria sembla [cansada].* **(Spa)** *Maria parece [cansada].*

Rg — *Verbal regimen* or simply *Regimen*. This is an extra argument (not subject, nor object nor indirect object, although sometimes it is very similar to some of them) for some verbs. It is frequently introduced by a preposition.[3] There are many regimen functions; it would be convenient to differentiate them, but we have condensed them all into one. **(Eng)** *John works [as a musician].* **(Cat)** *El Joan fa [de músic]. La Maria s'interessa [per la música clàssica].* **(Spa)** *Juan trabaja [de músico]. Maria se interesa [por la música clásica].*

[1] For some linguists, who do not accept the subject as an argument, it is the first argument.

[2] This is also called the second object in di-transitive verbs.

[3] The name is taken from Romance grammarians.

Adjunct Functions

Ap — *Adjunct of place*. Function which announces where the action occurs.[4, 5] **(Eng)** *John buys fish [here].* **(Cat)** *La Maria treballa [a l'oficina del costat].* **(Spa)** *Maria trabaja [en la oficina de al lado].*

At — *Adjunct of time*. Function which announces when the action occurs.[6] **(Eng)** *Mary works tomorrow .* **(Cat)** *La Maria treballa [demà].* **(Spa)** *María trabaja [mañana].*

Am — *Adjunct of mode*. Function which announces how the action occurs.[7] **(Eng)** *John buys fish [fortunately].* **(Cat)** *[Per sort] demà no treballem . Ho enllestirem [molt ràpidament].* **(Spa)** *Afortunadamente mañana no se trabaja . Lo acabaremos [muy rápidamente].*

Ab — *Adjunct because*. Indicates the cause or reason. **(Eng)** *I like sunny days [because of the light].* **(Cat)** *M' agraden els dies de sol [per la llum].* **(Spa)** *Me gustan los dias de sol [por la luz].*

··· — Others. We only have shown the more basic adjuncts. However other are often recognized: instrument, company, finality etc.

Nominal Functions

Qn — *Quantifier*. This establishes a quantity or cardinality (*one, two, some, all,* . . .). The quantifier function always selects a lexical element in the category of quantifier. **(Eng)** *The [two] black cats scratch the sofa*; *[almost one-hundred] penguins are jumping together*. **(Cat)** *[gairebé tots] els nois hi eren.* **(Spa)** *Estaban [Casi todos] los chicos .*

Dt — *Determiner*. This function is very similar to a quantifier. However determiners give some extra information which concerns the speaker, such as uniqueness, context, proximity or prominence. The determiner function always selects a lexical element from the category of determiners. **(Eng)** *[the] two black cats scratch the sofa.* **(Cat)** *[els] dos gats negres d'en Joan esgarrapen el sofà.* **(Spa)** *los dos gatos negros de Juan arañan el sofá.*

[4] Generally it is an optional argument, however some verbs in Catalan always take an adjunct: **(Cat)** *Posa-hi més sucre/*Posa més sucre*.

[5] If we wanted a detailed grammar, surely we would need to distinguish several functions like, for example, an adjunct of place for origin (*from here*), another for destination (*to there*), and others.

[6] We could also distinguish an adjunct of frequency, **(Eng)** *sometimes*, an adjunct of time location, **(Eng)** *yesterday*, or adjunct of time interval, **(Eng)** *during two days*, amongst others.

[7] This function is related to modal logic, so for a detailed analysis it would be convenient to distinguish cases. **(Eng)** *[necessarily] dogs are animals*.

Ps — *Possesive*. This function is very similar to a determiner. Possessives give information about the noun and the owner. The possessive function always selects a lexical element from the category of possessives. *My two black cats scratch the sofa.* **(Cat)** *Els [meus] dos gats negres esgarrapen el sofà.* **(Spa)** *Mis dos gatos negros arañan el sofá.*

Ad — *Adjective*. This function generally establishes a constraint on the noun. The adjective function always selects a lexical element in the category of adjectives. **(Eng)** Two [black] cats scratch the sofa. **(Cat)** *Dos gats negres esgarrapen el sofà.* **(Spa)** *Dos gatos negros arañan el sofá.*

Nc — *Noun complement*. This function is very similar to *Ad*, but *Nc* is usually introduced by a preposition and it allows construction of a new syntagma. The Noun complement function always selects a lexical element in the category of nouns. **(Eng)** *the two black cats [from the street] scratch the sofa.* **(Cat)** *Els dos gats negres [del carrer] esgarrapen el sofà.* **(Spa)** *Los dos gatos negros [de la calle] arañan el sofá.*

··· — Others.

Modifier Functions

Md — *Modifier*. Function which modifies the governor in mode or intensity *[very] expensive, [too] expensive, ...* Modifier function always selects a lexical element in the category of modifiers. **(Eng)** *It will be a [very] long trip.* **(Cat)** *Ha resultat ser una festa [ben] lluïda .* **(Spa)** *Será un viaje [muy] largo .*

Ma — *Adjective Modifier Complement*. Function which introduces an argument for the adjective: **(Eng)** *She felt very sure [of herself].* **(Cat)** *Ella va sentir-se molt segura [de si mateixa].* **(Spa)** *Ella se sintió muy segura [de si misma].*

··· — Others.

Introductory Functions

Ip — *Introduction of a preposition.* Introduces materially a preposition. This function always selects a lexical element in the category of prepositions. **(Eng)** *the two black cats [from] the street .* **(Cat)** *Els dos gats negres [de] 'l carrer .* **(Spa)** *Los dos gatos negros [de] la calle .*

Ic — *Introduction of a coordinator.* Introduces materially a coordinator. This function always selects a lexical element in the category of coordinators. **(Eng)** *a dog [and] a cat . Un gos [i] un gat .* **(Spa)** *Un perro [y] un gato .*

Coordination and Connecter Functions

Co_{and} — *Coordination And.* **(Eng)** *John [and Mary] come to supper.* **(Cat)** *El Joan [i la Maria] venen a sopar.* **(Spa)** *Juan [y María] vienen a cenar.*

Co_{or} — *Coordination Or.* Similar to the coordination function and.

Co_{if} — *Conditional.* **(Eng)** *If I come to supper, [I will prepare desserts].* **(Cat)** *Si vinc a sopar, [jo faré les postres].* **(Spa)** *Si vengo a cenar, yo prepararé los postres.*

··· — Others like *while*, *however*, etc.

B.3 Subordinate Isotope Functions

Subordinate Verbal Functions

Subordinate Unit Functions

Ax — *Auxiliar function.* This introduces a verb governed by an auxiliary verb. **(Eng)** *We have [won again].* **(Cat)** *hem [guanyat un altre cop].* **(Spa)** *Hemos [ganado otra vez*].

Subordinate Canonical Functions

Sb_S — *Sentential subordinate subject.* The subject is not a noun, but a subordinate clause. **(Eng)** *[Whoever gets the best mark], will receive a medal.* **(Cat)** *[Qui menja sopes] se les pensa totes.* **(Spa)** *[Quien haya robado las galletas] lo pagará caro.*

Ob_S — *Sentential subordinate object.* **(Eng)** *Yesterday I discovered [John hates vegetables].* **(Cat)** *El Joan diu [que no vindrà a sopar].* **(Spa)** *Juan dice [que no vendra a cenar].*

In_S — *Sentential subordinate indirect object.* **(Eng)** *She gives money to [whoever does not need it].* **(Cat)** *Li ha donat els diners [a qui menys ho necessita].* **(Spa)** *Le dió el dinero [a quién menos lo necesita].*

Pd_S — *Sentential subordinate predicate.* **(Eng)** *She appeared [to hide the key]* **(Cat)** *Semblava [que amagués les claus].* **(Spa)** *Parecía [que escondiera las llaves].*

Rg_S — *Sentential subordinate regimen.* **(Eng)** *I'm interested [in what nobody considers relevant].* **(Cat)** *M' interesso [pel que ningú creu important].* **(Spa)** *Me intereso [por lo que nadie cree importante].*

··· — Others.

Subordinate Adjunct Functions

Ap_S — *Sentential subordinate adjunct of place.* **(Eng)** *We met [where she wanted].* **(Cat)** *[D'on vinc jo] les muntanyes són altes. Hem quedat [on ella ha volgut].* **(Spa)** *[De donde vengo] las montañas son altas.*

At_S — *Sentential subordinate adjunct of time* **(Eng)** *We met [when she wanted].* **(Cat)** *Hem quedat [quan ella ha volgut].* **(Spa)** *Quedamos [cuando ella quiso].*

Am_S — *Sentential subordinate adjunct of mode.* **(Eng)** *We met [how she wanted].* **(Cat)** *Hem quedat [com ella ha volgut].* **(Spa)** *Quedamos [como ella quiso].*

Ab_S — *Sentential subordinate because.* **(Eng)** *She won't come [because she is busy].* **(Cat)** *No vindrà [perquè té feina].* **(Spa)** *No vendrá [ya que está ocupada].*

… — Others.

Subordinate Nominal Functions

Qn_S — *Sentential subordinate quantifier.*[8] **(Cat)** *N'atraparé [tantes com pugui].* **(Spa)** *Atraparé [tantas] ranas [como pueda].*

Dt_S — —

Ps_S — —

Ad_S — *Sentential subordinate adjective.* **(Eng)** *The man [who wears yellow shoes] will come tomorrow.* **(Cat)** *L' home [que duu les sabates grogues] vindrà demà.* **(Spa)** *El hombre [que lleva zapatos amarillos] vendrá mañana.*

Nc_S — *Sentential subordinate noun complement.* **(Eng)** *I don't find the novel [with which John won the prize].* **(Cat)** *No trobo la novel·la [amb la qual guanyà el premi el Joan].* **(Spa)** *No encuentro la novela [con la que Juan ganó el premio].*

… — Others.

Subordinate Modifier Functions

Md_S — *Sentential subordinate modifier* **(Eng)** *I will catch [as many frogs as I can]. On the hill was erected a house more luxurious [than you could ever imagine].* **(Cat)** *Una casa luxosa [com no us imaginareu mai] s'alçava damunt del turó.* **(Spa)** *En la colina se erigía una casa lujosa [como no podrias imaginar].*

Ma_S — —

… — —

B.4 Interrogative Isotope Functions

[8] This function does not exist in English, which makes the subordination through the modifier subordination isotope (see Md_S later).

Interrogative Verbal Functions

Interrogative Canonical Functions

Sb$_?$ — *Interrogative subject.* **(Eng)** *[Who] bought flowers for Mary*? **(Cat)** *[Qui] ha comprat flors a la Maria?* **(Spa)** *¿[Quien] compró flores a Maria?*

Ob$_?$ — *Interrogative object.* **(Eng)** *[What] did John buy for Mary]*? **(Cat)** *[Què] ha comprat el Joan a la Maria?* **(Spa)** *¿[Qué] compró Juan a Maria*?

In$_?$ — *Interrogative indirect object.* **(Eng)** *[who did John buy flowers?* **(Cat)** *[A qui] ha comprat el Joan flors?* **(Spa)** *¿ [A quién] compró flores Juan?*

Pd$_?$ — *Interrogative predicate.* **(Eng)** *[What] is Mary?* **(Cat)** *[Què] sembla la Maria amb aquest barret?* **(Spa)** *¿[Qué] parece María con éste sombrero?*

Rg$_?$ — *Interrogative regimen.* **(Eng)** *[What] does John work as?* **(Cat)** *[De què] fa el Joan?* **(Spa)** *¿[De qué] trabaja Juan?*

· · · — Others.

Interrogative Adjunct Functions

Ap$_?$ — *Interrogative adjunct of place.* **(Eng)** *[Where] do you come [from]?* **(Cat)** *[D'on] vens? [On] hem quedat?* **(Spa)** *¿[De donde] vienes ?*

At$_?$ — *Interrogative adjunct of time.* **(Eng)** *[When] do you come?* **(Cat)** *[Quan] vindràs ?* **(Spa)** *¿[Cuándo] vendrás ?*

Am$_?$ — *Interrogative adjunct of mode.* **(Eng)** *[How] do things work here?* **(Cat)** *[Com] van les coses aquí?* **(Spa)** *¿[Cómo] van las cosas aquí?*

Ab$_?$ — *Interrogative because.* **(Eng)** *[Why] does John not have a job?* **(Cat)** *[perquè] no té feina el Joan?* **(Spa)** *¿[Porqué] no tiene trabajo Juan?*

· · · — Others.

Interrogative Nominal Functions

Qn$_?$ — *Interrogative quantifier.*[9] **(Cat)** *[Quants] peixos has comprat?* **(Spa)** *¿[Cuántos] peces has comprado?*

Dt$_?$ — —

Ps$_?$ — —

Ad$_?$ — —[10]

Nc$_?$ — *Interrogative noun complement.* **(Eng)** *[Whose] car have you stolen?* **(Cat)** *El cotxe [de qui] has robat?* **(Spa)** *El coche [de quién] has robado?*

[9] This interrogative pronoun does not exist in English, which makes the interrogation of quantity through the modifier interrogative pronoun (see *Md*$_?$ later).

[10] This function seems not to exist. Usually in order to make a question about an adjective we use the interrogative predicate. **(Eng)** *What was the stolen car like?* **(Cat)** *Com era el cotxe robat?* **(Spa)** *¿Cómo era el coche robado?*

Interrogative Modifier Functions

$Md_?$ — *Interrogative modifier.* **(Eng)** *[How] many attempts do you need? [How] long is the paper?* In Catalan the interrogative modifier is used to ask for a non-countable amount, not for a quantifier. **(Cat)** *[Com] de llargs vols els pantalons? (els vull molt/poc/mig llargs).* Similarly in Spanish. **(Spa)** *¿[Como] de largo es el artículo.*

$Ma_?$ — —

· · · — —

B.5 Relative Pronoun Isotope Functions

Relative Pronoun Verbal Functions

Relative Pronoun Unit Functions

Re — *Relative pronoun unit.* **(Eng)** *John says [that] he likes jazz.* **(Cat)** *El Joan diu [que] li agrada el jazz.* **(Spa)** *Juan dice [que] le gusta el jazz.*

Relative Pronoun Canonical Functions

Sb_R — *Relative pronoun subject.* **(Eng)** *She gave the money to a man [who] does not need it.* **(Cat)** *Li ha donat els diners a [qui] menys ho necessita .* **(Spa)** *Le dió el dinero a [quien] menos lo necesita .*

Ob_R — *Relative pronoun object.* **(Eng)** *I do not know [what] kids want .* **(Cat)** *No sé [què] volen els nois.* **(Spa)** *No sé [qué] quieren los chicos.*

In_R — *Relative pronoun indirect object.* **(Eng)** *I met the girl [who] John gives flowers.* **(Cat)** *He conegut la noia [a qui] el Joan regala flors.* **(Spa)** *He conocido la chica a [quién] Juan regala flores.*

Pd_R — *Relative pronoun predicate.* **(Eng)** *I'm not sure [how] John is .* **(Cat)** *El [què] això sembli no m'interessa.* **(Spa)** *Lo [qué] eso parezca no me interesa.*

Rg_R — *Relative pronoun regimen.* **(Cat)** *Del [que] treballi el Joan no és rellevant.* **(Spa)** *De lo [que] trabaje Juan no es relevante.*

· · · — Others.

Relative Pronoun Adjunct Functions

Ap_R — *Relative pronoun adjunct of place.* **(Eng)** *I don't want to know [where] you come from.* **(Cat)** *No vull saber [d'on] vens .* **(Spa)** *No quiero saber de [donde] vienes .*

At_R — *Relative pronoun adjunct of time.* **(Eng)** *Tell me [when] you arrive .* **(Cat)** *digue'm [quan] arribaràs .* **(Spa)** *dime [*cuando*] llegarás .*

Am_R — *Relative pronoun adjunct of mode.* **(Eng)** *Tell me [how] the tale finishes .* **(Cat)** *Explica'm [com] acaba el conte.* **(Spa)** *Cuentame [como] termina el cuento.*

Ab_R — *Relative pronoun adjunct because.* **(Eng)** *Tell me [why] John bought the house.* **(Cat)** *Explica'm [perquè] el Joan va comprar la casa.* **(Spa)** *Cuentame [porqué] Juan compró la casa.*

··· — Others.

Relative Pronoun Nominal Functions

Qn_R — *Relative pronoun quantifier.*[11] **(Cat)** *No sabrem mai [quants] grans de sorra hi ha a la platja.* **(Spa)** *No sabremos nunca [cuantos] granos de arena hay en la playa.*

Dt_R — —

Ps_R — —

Ad_R — —

Nc_R — *Relative pronoun noun complement.* **(Eng)** *The book the title [of which] I can't remember will be reedited.* **(Cat)** *El llibre el títol [del qual] no recordo serà reeditat.* **(Spa)** *El libro el título [del cual] no recuerdo será reeditado.*

··· — Others.

Relative Pronoun Modifier Functions

Md_R — *Relative pronoun adjunct of modifier.* **(Eng)** *We don't know [how] long the novel is.* **(Cat)** *No sabem [com] de llarga és la novel·la.* **(Spa)** *No sabemos [cuan] larga es la novela.*

Ma_R — —

··· — Others.

B.6 Topicalization Isotope Functions[12]

Topicalization Verbal Functions

Topicalization Canonical Functions

$Sb_!$ — *Topicalized subject.*[13] **(Cat)** *Hi aniré [jo mateix].* **(Spa)** *Iré [yo mismo].*

[11] As in the case of the interrogative pronoun this isotope does not exist in English.

[12] The following examples are usually with emphatic tone.

[13] This function either does not exists or its effect is invisible in SVO languages which only topicalize to the left margin. However in Romance languages there is the possibility of topicalizing the subjects to the right margin.

$Ob_{!}$ — *Topicalized object.* **(Eng)** *[Vegetables], kids hate .* **(Cat)** *[Flors], ha comprat el Joan per la Maria.* **(Spa)** *[Flores], ha comprado Juan para María.*

$In_{!}$ — *Topicalized indirect object.* **(Eng)** *[Mary], John buys flowers.* **(Cat)** *[Per la Maria], el Joan ha comprat flors.* **(Spa)** *[Para Maria], Juan ha comprado flores.*

$Pd_{!}$ — *Topicalized predicate.* **(Eng)** *[Very nice], she is .* **(Cat)** *[ben bonica] que és .* **(Spa)** [Bien bonita] que es .

$Rg_{!}$ — *Topicalized regimen.* **(Eng)** *[As musician], John works .* **(Cat)** *[De music], fa el Joan.* **(Spa)** *[De músico], trabaja Juan.*

$\cdots$ — Others.

Topicalization Adjunct Functions

$Ap_{!}$ — *Topicalized adjunct of place.*[14] **(Cat)** *[A la muntanya], no m'hi veuràs mai.*

$At_{!}$ — *Topicalized adjunct of time.* Similar to $Ap_{!}$.

$Am_{!}$ — *Topicalized adjunct of mode.* Similar to $Ap_{!}$.

$Ab_{!}$ — *Topicalized adjunct of cause.* **(Eng)** *[For this reason], I don't want to pay the bill.* **(Cat)** *[Per aquesta raó] no vull pagar el compte.* **(Spa)** *[Por esta razón] no quiero pagar la cuenta.*

$\cdots$ — Others.

Topicalized Nominal Functions

$Qn_{!}$ — *Topicalized quantifier.*[15] **(Cat)** *[tres], n'he comprat, de peixos .*

$Dt_{!}$ — —

$Ps_{!}$ — —

$Ad_{!}$ — *Topicalized adjective.*[16]: **(Cat)** *[Groc], se l'ha comprat, el cotxe , el Joan.*

$Nc_{!}$ — —[17]

Topicalized Modifier Functions

[14] In English this isotope becomes invisible since adjuncts can already be placed at the margins. The same is true in Spanish. However in Catalan we have a proof of this isotope because of the presence of a redundant pronoun called *pronom de represa* (resumptive pronoun) as in the example, where *hi* is the resumptive pronoun which duplicates the adjunct.

[15] English and Spanish disallow this. In Catalan it is allowed to topicalize the quantifier provided we insert a duplicate pronoun, as in the example. In Spanish this topicalization is not possible. Notice that in the sentence, for example, *tres peces he comprado* what is topicalized is the full object, not the quantifier.

[16] In Catalan this is similar to the case of the quantifier. In Spanish this topicalization seems unclear.

[17] A possibility, but not clear, is **(Cat)** *De la Maria, és fill el Joan!*.

$Md_!$ — —

$Ma_!$ — —

· · · — —

B.7 Pronoun Isotope Functions

Pronominal Verbal Functions

Pronominal Canonical Functions

Sb_P — *Pronominal subject.* **(Eng)** *[She] bought the flowers.* **(Cat)** *[Ella] ha comprat les flors.* **(Spa)** *[Ella] ha comprado las flores.*

Ob_P — *Pronominal object.* **(Eng)** *Mary bought [them].* **(Cat)** *La Maria [les] ha comprades.* **(Spa)** *Maria [las] compró.*

In_P — *Pronominal indirect object.* **(Eng)** *John gave [her] some money.* **(Cat)** *El Joan va donar-[li] diners.* **(Spa)** *Juan [le] dió dinero.*

Pd_P — *Pronominal predicate.* **(Cat)** *El Joan [ho] sembla.* Catalan can use a more specific pronoun (*en*) for the predicate. **(Cat)** *El Joan [n'] és.* **(Spa)** *Juan [lo] es.*

Rg_P — *Pronominal regimen.* **(Cat)** *No [hi] estic interessat ara mateix.*

· · · — Others.

Pronominal Adjunct Functions

Ap_P — *Pronoun adjunct of place.* Catalan uses specific pronouns for adjuncts of place. **(Cat)** *Ara [hi] vaig.* English and Spanish use demonstratives as pronouns. **(Eng)** *We go [there].* **(Spa)** *Vamos* [allá].

At_P — —

Am_P — *Pronominal adjunct of mode.* Catalan and Spanish use demonstratives as pronouns. **(Cat)** *Es fa [així].* **(Spa)** *Se hace [así].*

Ab_P — *Pronominal because.* Catalan and Spanish uses a demonstrative pronoun **(Cat)** *[Per això] no vindré.* **(Spa)** *Por [eso] no iré.*

· · · — Others.

Pronominal Modifier Functions

Md_P — —

Ma_P — —

· · · — —

B.8 Tables of Functions

		UNITS	VERBAL											NOMINAL						MODIFIER		
			CANONICAL						ADJUNCT													
	Primitive	1	*Sb*	*Ob*	*In*	*Pd*	*Rg*	. . .	*Ap*	*At*	*Am*	*Ab*	. . .	*Qn*	*Dt*	*Ps*	*Ad*	▨ *Nc*	. . .	▨ *Md*	*Ma*	. . .
Isotopes	Subord.	▨ *Ax*	▨ Sb_S	▨ Ob_S	▨ In_S	▨ Pd_S	▨ Rg_S	▨ . . .	▨ Ap_S	▨ At_S	▨ Am_S	▨ Ab_S	▨ . . .				Ad_S	Nc_S	. . .	Md_S	Ma_S	. . .
	Interrog.		$Sb_?$	$Ob_?$	$In_?$	$Pd_?$	$Rg_?$	. . .	$Ap_?$	$At_?$	$Am_?$	$Ab_?$	. . .				$Ad_?$	$Nc_?$	. . .	$Md_?$	$Ma_?$	. . .
	Relative Pronoun	*Re*	Sb_R	Ob_R	In_R	Pd_R			Ap_R	At_R	Am_R	Ab_R	. . .					Nc_R	. . .	Md_R	Ma_R	. . .
	Topical.			$Ob_!$	$In_!$	$Ap_!$	$Rg_!$	. . .														
	Pronoun		Sb_P	Ob_P	In_P																	

INTRODUCTORY	
Ip	*Ic*

COORDINATION & CONNECTER			
▨ Co_{and}	▨ Co_{or}	▨ Co_{if}	▨ . . .

▨ Cluster functions

. . . Other functions of the same kind

Table B.7: table of functions for English

VERBAL												NOMINAL						MODIFIER				
UNITS	CANONICAL						ADJUNCT															
1	Sb	Ob	In	Pd	Rg	· · ·	Ap	At	Am	Ab	· · ·	Qn	Dt	Ps	Ad	Nc	· · ·	Md	Ma	· · ·	Primitive	
Ax	Sb_S	Ob_S	In_S	Pd_S	Rg_S	· · ·	Ap_S	At_S	Am_S	Ab_S	· · ·	Qn_S			Ad_S	Nc_S	· · ·	Md_S	Ma_S	· · ·	Subord.	Isotopes
	$Sb_?$	$Ob_?$	$In_?$	$Pd_?$	$Rg_?$	· · ·	$Ap_?$	$At_?$	$Am_?$	$Ab_?$	· · ·	$Qn_?$			$Ad_?$	$Nc_?$	· · ·	$Md_?$	$Ma_?$	· · ·	Interrog.	
Re	Sb_R	Ob_R	In_R	Pd_R	Rg_R	· · ·	Ap_R	At_R	Am_R	Ab_R	· · ·	Qn_R				Nc_R	· · ·	Md_R	Ma_R	· · ·	Relative Pronoun	
	$Sb_!$	$Ob_!$	$In_!$	$Pd_!$	$Rg_!$	· · ·	$Ap_!$	$At_!$	$Am_!$	$Ab_!$	· · ·										Topical.	
	Sb_P	Ob_P	In_P	Pd_P	Rg_P	· · ·	Ap_P	At_P	Am_P												Pronoun	

INTRODUCTORY		COORDINATION & CONNECTER			
Ip	Ic	Co_{and}	Co_{or}	Co_{if}	· · ·

▨ Cluster functions · · · Other functions of the same kind

Table B.8: table of functions for Catalan

VERBAL												NOMINAL						MODIFIER				
UNITS	CANONICAL						ADJUNCT															
1	Sb	Ob	In	Pd	Rg	· · ·	Ap	At	Am	Ab	· · ·	Qn	Dt	Ps	Ad	Nc	· · ·	Md	Ma	· · ·	Primitive	
Ax	Sb_S	Ob_S	In_S	Pd_S	Rg_S	· · ·	Ap_S	At_S	Am_S	Ab_S	· · ·	Qn_S			Ad_S	Nc_S	· · ·	Md_S	Ma_S	· · ·	Subord.	Isotopes
	$Sb_?$	$Ob_?$	$In_?$	$Pd_?$	$Rg_?$	· · ·	$Ap_?$	$At_?$	$Am_?$	$Ab_?$	· · ·	$Qn_?$			$Ad_?$	$Nc_?$	· · ·	$Md_?$	$Ma_?$	· · ·	Interrog.	
Re	Sb_R	Ob_R	In_R	Pd_R	Rg_R	· · ·	Ap_R	At_R	Am_R	Ab_R	· · ·	Qn_R				Nc_R	· · ·	Md_R	Ma_R	· · ·	Relative Pronoun	
	$Sb_!$	$Ob_!$	$In_!$	$Pd_!$	$Rg_!$	· · ·				$Ab_!$	· · ·										Topical.	
	Sb_P	Ob_P	In_P	Pd_P																	Pronoun	

INTRODUCTORY	
Ip	Ic

COORDINATION & CONNECTER			
Co_{and}	Co_{or}	Co_{if}	· · ·

▨ Cluster functions

· · · Other functions of the same kind

Table B.9: table of functions for Spanish

C

Aspects of Time Complexity of Manifolds

This appendix is devoted to studying decidability of manifolds in terms that were defined in Definition 4.12. We saw that manifolds in $\mathbf{Man}(\mathbf{k}^p\mathbb{H})$ and manifolds in $\mathbf{Man}(\mathbb{H}\mathbf{k}^p)$ are linear time decidable for any $p \geq 0$; Theorems 4.13 and §5.2. Here we generalize these results for more complex patterns.

The time for deciding an intersection/union of manifolds is the sum of the time of deciding the manifolds. Since a manifold is a finite $\cup\cap$-combination of simple manifolds, in what follows we only treat the case of simple manifolds. We restrict ourselves to those patterns in $\{\mathbf{k}, \mathbb{H}\}^*$ or $\{\mathbf{k}, \mathbb{H}, \mathbb{P}\}^*$ which are the patterns related to natural constructions. First we have to address the auxiliary problem of deciding patterns.

C.1 Time for Patterns

C.1.1 Automata over Monoids for Rational Sets

Let us introduce some notation. We call the elements $x \in \zeta^{*a}$ *string vectors*. Recall that we extended length to string vectors by $|(x_1, \ldots, x_a)| = \max\{|x_1|, \ldots, |x_a|\}$. We introduce the norm $\|x\| = \|(x_1, \ldots, x_a)\| = \sum_{i=1}^{a} |x_i|$, where $|x_i|$ is the length of $x_i \in \zeta^*$. Note that $\|x \cdot x'\| = \|x\| + \|x'\|$. Notice also that a comparison between two strings, $x \overset{?}{=} y$, $x, y \in \zeta^*$ can be made in $\min\{|x|, |y|\}$ steps, because if $x = x_1 \cdots x_n$, $x_i \in \zeta$ and $y = y_1 \cdots y_m$, $y_i \in \zeta$, then we have to check whether $x_1 \overset{?}{=} y_1$, $x_2 \overset{?}{=} y_2, \ldots$ until one of the strings finishes. Equally the comparison of two string vectors $x, y \in \zeta^{*a}$ can be made in $\min\{\|x\|, \|y\|\}$. The notions of prefix and suffix generalize for string vectors: consider $x, y \in \zeta^{*a}$; we say that y is a *prefix* of x if there is a $z \in \zeta^{*a}$ such that $x = yz$, and y is a *suffix* of x iff $x = zy$. However, and importantly, differently to free monoids, a vector string can have several prefixes and suffixes z with $|z| = 1$. For example the vector (α, β) has three (non-trivial) prefixes of length 1: $(\alpha, 1), (1, \beta), (\alpha, \beta)$.

In order to determine the running time of the problem $S \in^? W$ for a given syntagma S and a given manifold W, which will be done in the next sections, we need to know before the running time for deciding the problem $x \in^? \Gamma$ in terms of the norm $\|x\|$. We will suppose that patterns are given decomposed in basic patterns. The length of the pattern is the length of its decomposition.

We saw that patterns are a particular case of rational sets, see Remark 3.4. [1] Similarly to regular languages, the class of rational sets coincides with the class of sets which are accepted by a (non-deterministic) finite automaton over M. These automata are a direct generalization of automata over a free monoid, but now the transitions are labeled by elements of the monoid M.

Theorem C.1. *A subset of a monoid M is rational if and only if it is accepted by a finite automaton over M.*

Proof. See (Sakarovitch, 2009). □

We can construct an automaton accepting a pattern in $\mathrm{Rat}(\zeta^{*a})$. In fact the general construction is very simple. Given the pattern $\Gamma = \prod_{i=1}^{k} \Gamma_i$ we take the automaton with states $\{1, \ldots, k+1\}$. If $\Gamma_i \in \mathbb{M}$, with $\Gamma = \{\varphi_{i,1}, \ldots, \varphi_{i,n}\}^*$, the state i has n loop transitions labeled with $\varphi_{i,j}$, see Fig. C.1(a), and we add the trivial or identity transition $(1)_a$ from i to $i+1$. If $\Gamma_i \in \mathbf{k}$ with $\Gamma_i = \{\varphi_i\}$, we add the transition φ_i from i to $i+1$, see Fig. C.1(b). 1 is the initial state and $k+1$ is the state of acceptance.

General non-deterministic automata can also be determinized as in the case of automata over free monoids. However, differently to standard automata, even when the automaton is deterministic, this does not guarantee a deterministic computation (also called unambiguous computation). Consider the pattern $\{(\alpha, \alpha), (\beta, \beta)\}^*(1, \alpha)$. The automaton of Fig. C.1(c) accepts this pattern and it is deterministic, however the computation is in general not deterministic. This is because, for example, the string vector (α, α) has two possible prefixes of length 1: (α, α) itself and $(1, \alpha)$.

In general the problem of deciding whether a vector string belongs to a given pattern is not harder than NP. We cannot rely on automata in order to decide in polynomial time the problem $x \in^? \Gamma$. However for patterns involved in natural language constructions the problem turns out to be polynomial. We devote the rest of the appendix to this.

C.1.2 Time for Patterns in $\{\mathbf{k}, \mathbb{H}, \mathbb{P}, \mathbb{G}\}^*$

We are going to show an algorithm running in polynomial time for patterns in $\{\mathbf{k}, \mathbb{H}, \mathbb{P}, \mathbb{G}\}^*$, i.e. the monoids are homogeneous, generalized homogeneous or pivoting, see definitions in Example 3. Recall that $\mathbb{H} \subseteq \mathbb{G}$.

[1] The class of rational sets over a monoid M, $\mathrm{Rat}(M)$, is the least class of subsets of M such that it is closed by union, product and the star operator.

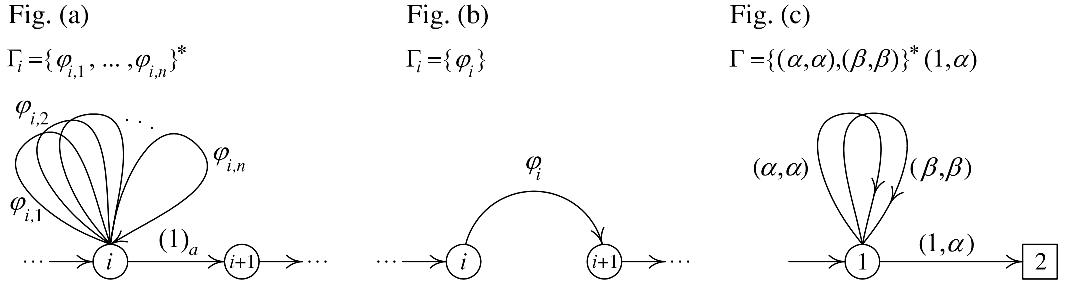


Figure C.1: (a) and (b) transitions of an automaton which accepts a pattern; (c) deterministic automaton on ζ^{*2} for a pattern which exhibits non-deterministic computations.

Lemma C.2. *If Γ is a basic pattern $\Gamma \in \mathbf{k} \cup \mathbb{G} \cup \mathbb{P}$, then the problem $x \in^? \Gamma$ can be solved in linear time $O(n)$, where $n = \|x\|$.*

Proof. Let us first consider the easiest case $\Gamma \in \mathbf{k}$. There is a φ such that $\Gamma = \{\varphi\}$ and then $x \in \Gamma \iff x = \varphi$. The problem $x =^? \varphi$ can be solved in constant time $\|\varphi\|$.

Now consider the case $\Gamma \in \mathbb{G}$ and let $\Gamma = \{\varphi_1, \ldots, \varphi_g\}^*$. Notice, first, that when the arity is $a = 1$ the problem is clearly linear. We factorize x in prime factors $x = x_1 \cdots x_n$ and then we check that each $x_i \in \{\varphi_1, \ldots, \varphi_g\}$. So we need $|x|$ steps. However monoids in $\mathbb{G}$ behave very similarly to patterns with arity 1, since they have all the components equal. Thanks to this we only have to check the first component. For a homogeneous vector we have that $\|(x)_a\| = a|x|$, and the time is $\|x\|/a$.

Finally consider the case $\Gamma \in \mathbb{P}$. Then $\Gamma = 1 \oplus \xi^*$ Since the first component is 1 we can ignore it, and then the problem is trivial. □

Now we see the general problem:

Theorem C.3. *Given a pattern $\Gamma \in \{\mathbf{k}, \mathbb{H}, \mathbb{G}, \mathbb{P}\}^*$, the problem $x \in^? \Gamma$ can be solved in polynomial time $O(n^{a(k-1)+1})$, where $n = \|x\|$, a is the arity, and k is the length of the given decomposition of Γ.*

Proof. First consider the problem of splitting a string. Given $x \in \zeta^*$ with length $|x| = n$, the number of ways of splitting the string x in k substrings, $x = x_1 \cdots x_k$, is given by the number $P_k(n)$ in basic combinatoric theory which is equal to the number of partitions of a number n in k ordered non-negative integers.[2] For example $P_2(3) = 4$ because $3 = 0 + 3 = 1 + 2 = 2 + 1 = 3 + 0$. The following expression counts and bounds it:

$$P_k(n) = \binom{n+k-1}{k-1} = \frac{(n+k-1)!}{(k-1)!n!} = \frac{1}{(k-1)!} \cdot \underbrace{(n+k-1)\cdots(n+1)}_{k-1 \text{ products}}$$

$$\le \frac{(n+k-1)^k}{(k-1)!} \le (n+k)^{k-1}.$$

[2] See for example (Comellas et al., 1996).

Notice that we allow that some substrings be empty (i.e. the identity element $1 \in \zeta^*$).

Now consider the same problem but with the vector string $x \in \zeta^{*a}$, which means splitting every component in k substrings. Since we have a components the number now is $P_k(n_1)\cdots P_k(n_a)$ where n_i is the length of the i-component of x. Notice that $n_i \leq n_1 + \cdots + n_a = \|x\| = n$ and then we have:

$$P_k(n_1)\cdots P_k(n_a) \leq P_k(n)\cdots P_k(n) = P_k(n)^a \leq (n+k)^{(k-1)a}.$$

Now we consider the question in hand, $x \in^? \Gamma$. We assume that $\Gamma = \Gamma_1 \cdots \Gamma_k$ is given already decomposed in k basic patterns. Then we split x in k sub-vector-strings $x = x_1 \ldots x_k$. For each decomposition and for each factor x_s, $s = 1, \ldots, k$ we check if $x_s \in \Gamma_s$. If we have that $x_1 \in \Gamma_1, \ldots, x_s \in \Gamma_s$ for at least one of these decompositions, then $x \in \Gamma$. If not $x \notin \Gamma$.

Now we have only to calculate the time. We known from the above lemma that the problem $x \in^? \Gamma$ when Γ is a basic pattern can be solved in $O(\|x\|)$. So checking that $x_s \in \Gamma_s$ for all $s = 1, \ldots, k$, consumes $\|x_1\| + \cdots + \|x_k\| = \|x\| = n$, so the time complexity is $O(n)$.

Since this must be done for all the decompositions, we have to perform $(n+k)^{(k-1)a} \times O(n)$ steps. That is $O((n+k)^{a(k-1)} \cdot n) = O(n^{a(k-1)+1})$. □

Notice in addition that:

Corollary C.4. *The proof of the above theorem defines an algorithm which gives in fact all the possible decompositions of* $x = x_1 \cdots x_k$ *with* $x_i \in \Gamma_i$*, where the* Γ_i *are the basic patterns of the decomposition of the pattern,* $\Gamma = \Gamma_1 \cdots \Gamma_k$.

Proof. Trivial. □

C.2 General Algorithm for Manifolds

First we are going to see an algorithm for the general problem $S \in^? W$ which will serve next sections. Let $W = \mathbf{Synt}\binom{B}{\Gamma}$ be a manifold with arity a and let $S \in \mathbf{Synt}$ be a syntagma. Consider the pseudo-code in Fig. C.2. We make some comments. We have used the set $(\mathrm{Spt}(S) \sqcup \{\blacktriangle\})^a$, where $\blacktriangle$ is an auxiliary symbol not in $\mathrm{Spt}(S)$. This symbol represents any possible null locus.

The inputs are the syntagma S and the manifold W given by the valuation B and the pattern Γ. The syntagma is given just on the support $S : \mathrm{Spt}(S) \longrightarrow \Sigma$, (for example as a table). The pattern $\Gamma = \prod_{i=1}^{k} \Gamma_i$ is given already decomposed in a list $\Gamma = (\Gamma_1, \ldots, \Gamma_k)$.

The x is a vector $x = (x_1, \ldots, x_a) \in (\mathrm{Spt}(S) \sqcup \{\blacktriangle\})^a$. The command `for each` is a loop which cycles through all the possible $(\mathrm{Spt}(S) \sqcup \{\blacktriangle\})^a$, so the loop cycles $(|S| + 1)^a$ times.

Within this loop we differentiate two possible cases: $x \in (\mathrm{Spt}(S))^a$ and $x \notin (\mathrm{Spt}(S))^a$, i.e. x contains some $\blacktriangle$ in its components or not. The second case invokes the function

$\mathtt{OUTSIDE}_{S,\Gamma}(x)$ which we define later. Finally the $S_\blacktriangle : \mathrm{Spt}(S) \sqcup \{\blacktriangle\} \longrightarrow \Sigma_+$ is the mapping defined as $S_\blacktriangle(x) = S(x)$ provided $x \neq \blacktriangle$, and $S_\blacktriangle(\blacktriangle) = 0$. We use the abbreviation for the vector $\overline{S_\blacktriangle}(x) = \big(S_\blacktriangle(x_1), \dots, S_\blacktriangle(x_a)\big)$.

Definition C.5. Suppose that $x_{i_1}, \dots, x_{i_t}$ are all the $t > 0$ components of $x \in (\mathrm{Spt}(S) \sqcup \{\blacktriangle\})^a$ which are $\blacktriangle$, in particular $x \notin (\mathrm{Spt}(S))^a$. Given a subset $Y = \{y_{i_1}, \dots, y_{i_t}\} \subseteq \zeta^*$ with the same indexation as x_{i_j}, we define the vector $(x|Y)$ which consists in substituting all the $x_{i_s} = \blacktriangle$ by the corresponding $y_{i_s} \in Y$ for each $s = 1, \dots, t$.

The function $\mathtt{OUTSIDE}_{S,\Gamma}(x)$ returns 1 if there is a $Y = \{y_{i_1}, \dots, y_{i_t}\} \subseteq \mathrm{Spt}(S)$ such that $S(y_{i_1}) = \ldots = S(y_{i_t}) = 0$ and such that $(x|Y) \in \Gamma$, otherwise $\mathtt{OUTSIDE}_{S,\Gamma}(x) = 0$.[3]

Theorem C.6. *The above algorithm,* `GENERAL_ALGORITHM_FOR_RECOGNITION_OF_MANIFOLD` *indeed works provided that* $\mathtt{OUTSIDE}_{S,\Gamma}(x)$ *can be computed.*

Proof. A first attempt to solve the problem consists in considering all the vectors $x \in \mathrm{Spt}(S)^a$. If $x \in \Gamma$ then we have to check if the syntagma satisfies the valuation B in the locus x. This is indeed the first `if` inside the loop of the algorithm.

However this does not consider the possible null loci. Furthermore, there are infinite null loci. However, we notice that we do not need to take account each null locus. We only need to know if there is some vector in Γ for which some of the components are null in S. In fact, we do not need to know these null loci, but just the existence. This is because the valuation only evaluates elements of $(\Sigma_+)^a$, not loci.

The second `if` in the loop considers this possibility. So we represent with a $\blacktriangle$ a possible null locus. Then we check if given $x \in (\mathrm{Spt}(S) \sqcup \{\blacktriangle\})^a$, we can substitute the $\blacktriangle$'s by adequate null loci for S, i.e. $(x|Y)$ which in addition satisfies that $(x|Y) \in \Gamma$. This is done by the function $\mathtt{OUTSIDE}_{S,\Gamma}(x)$. When this is the case, we can evaluate whether the syntagma satisfies the valuation, i.e. whether $B\big(\overline{S_\blacktriangle}(x)\big) = 1$. This vector $\overline{S_\blacktriangle}(x)$ makes null all the $\blacktriangle$'s. □

C.3 Time for Manifolds with Type in $\{\mathbf{k}, \mathbb{H}, \mathbb{G}\}^*$

When the monoids are in $\Gamma \in \{\mathbf{k}, \mathbb{H}, \mathbb{G}\}^*$ we can develop an algorithm and calculate the running time for the function `OUTSIDE`. Since the calculations can be a bit involved we consider first that patterns are unambiguous and that syntagmata are non-elliptic. In the end we will incorporate the general features.

First we see the case $\mathtt{OUTSIDE}_{S,\Gamma}(x)$ when $x \neq (\blacktriangle)_a$ and then we will see the other case $x \neq (\blacktriangle)_a$.

We will need a technical trick. Patterns in $\{\mathbf{k}, \mathbb{H}, \mathbb{G}\}^*$ have the property that one can reconstruct all the components of an element from a given component. Consider for example the pattern of arity 2, $(\alpha, \beta)(\alpha, \alpha)^* \in \mathbf{k}\mathbb{G}$. If the first component is α^5 then the second component is $\beta\alpha^4$.

[3] We named it "outside" because it consider loci outside the syntagma.

```
program GENERAL_ALGORITHM_FOR_RECOGNITION_OF_MANIFOLD
      read $S, B, \Gamma$;
      for each $x \in (\mathrm{Spt}(S) \sqcup \{\blacktriangle\})^a$ do
            evaluate $x \in^? (\mathrm{Spt}(S))^a$;
            if $x \in (\mathrm{Spt}(S))^a$ then
                  evaluate $x \in^? \Gamma$;
                  if $x \in \Gamma$ then
                        evaluate $B\big(\overline{S}(x)\big) =^? 1$;
                              if $B\big(\overline{S}(x)\big) = 1$ then
                                    continue;
                              else print $S \notin W$; haltprogram;
                              endif;
                  else continue;
                  endif;
            else evaluate OUTSIDE$_{S,\Gamma}(x) =^? 1$;
                  if OUTSIDE$_{S,\Gamma}(x) = 1$ then
                        if $B\big(\overline{S_\blacktriangle}(x)\big) = 1$ then continue;
                        else print $S \notin W$; haltprogram;
                        endif;
                  else continue;
                  endif;
            endif;
      endfor;
      print $S \in W$;
endprogram;
```

Figure C.2: general algorithm for recognition of a manifold $W = \mathbf{Synt}\binom{B}{\Gamma}$ which calls the function OUTSIDE$_{S,\Gamma}$.

We recall that the (algebraic) j-th *projection* is the mapping: $\pi_j : \zeta^{*a} \longrightarrow \zeta^*$ $\pi_j(x_1, \dots, x_a) = x_j$. We also recall that a decomposition of a pattern $\Gamma = \Gamma_1 \cdots \Gamma_k$ is *unambiguous* iff every $x \in \Gamma$ factorizes in a unique form with k factors in Γ_i.

Lemma C.7. *Consider $\Gamma \in \{\mathbf{k}, \mathbb{H}, \mathbb{G}\}^*$ with arity a and an unambiguous factorization $\Gamma = \Gamma_1 \cdots \Gamma_k$. Suppose that $x \in \Gamma$. Given a component of x, say x_j, we can reconstruct the rest of the components in time $O(n^{a(k-1)+1})$, where $n = \|x\|$.*

Proof. We suppose $x_j \in \zeta^*$ given. We take the pattern given by the projection $\pi_j(\Gamma) = \pi_j(\Gamma_1) \cdots \pi_j(\Gamma_k)$. This pattern has arity 1, but, in any case, we can apply the algorithm of the proof of Theorem C.3 which in addition yields a factorization of this component $x_j = x_{j,1} \cdots x_{j,k}$ in terms of the basic patterns, i.e.: $x_{j,1} \in \pi_j(\Gamma_1), \dots, x_{j,k} \in \pi_j(\Gamma_k)$. All this take $O(n^{a(k-1)+1})$ steps, see Corollary C.4. This factorization is unique since the decomposition of $\Gamma = \Gamma_1 \cdots \Gamma_k$ is unambiguous.[4]

Now we consider any other component of x, say x_i. This component factorizes as $x_i = x_{i,1} \cdots x_{i,k}$ in terms of the basic patterns, i.e.: $x_{i,1} \in \pi_i(\Gamma_1), \dots, x_{i,k} \in \pi_i(\Gamma_k)$. Our goal is to reconstruct $x_{i,1}, \dots, x_{i,k}$ from the $x_{j,1}, \dots, x_{j,k}$ which we have obtained in the above paragraph. With these $x_{i,1}, \dots, x_{i,k}$ we can multiply and then obtain $x_i = x_{i,1} \cdots x_{i,k}$.

Recall that $\Gamma \in \{\mathbf{k}, \mathbb{H}, \mathbb{G}\}^*$ by assumption. For each $s = 1, \dots, k$ we distinguishes two cases:

- Suppose that $\Gamma_s \in \mathbf{k}$, say $\Gamma_s = \{\varphi_s\}$. Then we can reconstruct immediately $x_{i,s} = \pi_i(\varphi_s)$. This takes constant time, $O(1)$.
- Suppose that $\Gamma_s \in \mathbb{G}$ (which contains the case $\mathbb{H} \subseteq \mathbb{G}$). Since Γ_s is homogeneous all the components are equal. An then $x_{i,s} = x_{j,s}$. This also takes constant time, $O(1)$.

If we do this for each $s = 1, \dots, k$ we can reconstruct $x_i = x_{i,1} \cdots x_{i,k}$ which takes time $O(k)$. Thus the total time is $O(n^{a(k-1)+1}) + O(k) = O(n^{a(k-1)+1})$. The two cases above reconstruct unequivocally x_i from the decomposition of x_j which, we recall, is unique. This proves that the reconstruction $(x_1, \dots, x_k)$ is really x. □

We note that the condition of unambiguity is not really necessary to obtain a polynomial time. However, allowing ambiguous patterns increases the time since we have to repeat calculations for each possible factorization. Notice that the number of possible factorizations of an $x \in \Gamma$ is bounded by $O(\|x\|^{a(k-1)+1})$. So in the case of ambiguity the algorithm would take time: $O(n^{a(k-1)+1}) + O(n^{a(k-1)+1}) \times \Big(O(n^{a(k-1)+1}) + O(n)\Big) = O(n^{2a(k-1)+2})$.

[4] Let us see this in more detail. Since the decomposition of Γ is unambiguous, for each $x \in \Gamma$ it decomposes in a unique way as $x = x^1 \cdots x^k$ (here the superscripts are indexes, not exponents). Then the j-th component of x, i.e. x_j, decompose in a unique way as $x_j = \pi_j(x^1) \cdots \pi_j(x^k) = x_{j,1} \cdots x_{j,k}$, where the $x_{j,s}$ are the j-th component of x^s.

Lemma C.8. *Consider a non-elliptic syntagma S with $|S| = n$, a pattern $\Gamma \in \{\mathbf{k}, \mathbb{H}, \mathbb{G}\}^*$ with arity a, length k, and with an unambiguous decomposition. Consider $x \in (\mathrm{Spt}(S) \sqcup \{\blacktriangle\})^a$, $x \notin (\mathrm{Spt}(S))^a$ such that $x \neq (\blacktriangle)_a$. Finally let $x_{i_1}, \dots, x_{i_t}$ be the components of x which are $\blacktriangle$'s. Then:*

(i) there is a unique possible set $Y = \{y_{i_1}, \dots, y_{i_t}\} \subseteq \zeta^*$ such that $(x|Y) \in \Gamma$;

(ii) the function $\mathtt{OUTSIDE}_{S,\Gamma}(x)$ with $x \neq (\blacktriangle)_a$ can be computed in time $O(n^{a(k-1)+1})$.

Proof. The condition $x \neq (\blacktriangle)_a$ implies that x contains at least one component $x_j \neq \blacktriangle$. So by the last Lemma C.7 we can reconstruct all the components from x_j, and this is the unique possibility. So we have already proved (i). Doing this takes $O(\|x_j\|^{a(k-1)+1})$ steps. However $x_j \in \mathrm{Spt}(S)$, and then $\|x_j\| \leq d < n$, where d is the depth of S and n is the size of S. So the time is bounded by $O(n^{a(k-1)+1})$.

Let us see (ii). In order to calculate the time of $\mathtt{OUTSIDE}_{S,\Gamma}(x)$ we have to obtain the set Y and then we have to check that (a) $(x|Y) \in^? \Gamma$ (which takes $O(n^{a(k-1)+1})$ by the above paragraph) and that (b) $S(y) =^? 0$ for each $y \in Y$ (which is done in time $t{\cdot}n \leq (a-1){\cdot}n = O(n)$, since every checking $S(y)=^?0$ consumes at most n steps).[5] If both are true then $\mathtt{OUTSIDE}_{S,\Gamma}(x) = 1$, otherwise $\mathtt{OUTSIDE}_{S,\Gamma}(x) = 0$. So we have to sum the time to obtain Y (which takes $O(n^{a(k-1)+1})$) with the time to check (a) (which takes $O(n^{a(k-1)+1})$) and (b) (which takes $O(n)$). That is: $O(n^{a(k-1)+1}) + O(n^{a(k-1)+1}) + O(n) = O(n^{a(k-1)+1})$. □

Lemma C.9. *Consider a syntagma S with $|S| = n$, a pattern $\Gamma \in \{\mathbf{k}, \mathbb{H}, \mathbb{G}\}^*$, and $x = (\blacktriangle)_a$. Then we have:*

(i) if $\Gamma \notin \mathbf{k}^p$ for any $p \geq 0$, then $\mathtt{OUTSIDE}_{S,\Gamma}(x) = 1$;

(ii) if $\Gamma \in \mathbf{k}^p$ for some $p \geq 0$ then:

$$\mathtt{OUTSIDE}_{S,\Gamma}(x) = 1 \iff S(\varphi_1) = \cdots = S(\varphi_a) = 0,$$

where $\Gamma = \{(\varphi_1, \dots, \varphi_a)\} \in \mathbf{k}^p$.

So the function $\mathtt{OUTSIDE}_{S,\Gamma}(x)$ with $x = (\blacktriangle)_a$ can be calculated in linear time $O(n)$.

Proof. Let the decomposition of Γ be $\Gamma = \Gamma_1 \cdots \Gamma_k$ and a its arity. Consider (i). We suppose that $\Gamma \notin \mathbf{k}^p$ for any $p \geq 0$; then there is a non-trivial monoid $\Theta \in \mathbb{G}$ such that $\Gamma = \Gamma'\Theta\Gamma$ where Γ', Γ'' are patterns. Since $\Theta \in \mathbb{G}$ is generalized homogeneous and non-trivial, there is an element $(\phi)_a = (\phi, \dots, \phi) \in \Theta$, such that $(\phi)_a \neq (1)_a$. Now we take any $\varphi' \in \Gamma'$ and $\varphi'' \in \Gamma''$. Consider the vector $y = \varphi' \cdot \big((\phi)_a\big)^{d+1} \cdot \varphi'' \in \Gamma'\Theta\Gamma'' = \Gamma$. It easy to check that any component of y has length greater than $d + 1$. If we take d

[5] Checking (a) is necessary because we have reconstructed only the $\blacktriangle$ components. However non-$\blacktriangle$ components could make $(x|Y)$ not be in Γ.

the depth of S, then for every component of $y = (y_1, \dots, y_a)$ we have that $S(y_i) = 0$. So we have proved that we can substitute all the ▲'s in x (that is, all the components) by $Y = \{y_1, \dots, y_a\}$ in such a way that $(x|Y) \in \Gamma$ and $S(y_1) = \cdots = S(y_a) = 0$. So $\mathtt{OUTSIDE}_{S,\Gamma}(x) = 1$.

Let us see (ii). We suppose $\Gamma \in \mathbf{k}^p$ for some $p \geq 0$. Then Γ is a constant, say $\Gamma = \{(\varphi_1, \dots, \varphi_a)\} \in \mathbf{k}^p$. The only possible substitution of the ▲ components is this vector $\Gamma = \{(\varphi_1, \dots, \varphi_a)\}$. So in this case: $\mathtt{OUTSIDE}_{S,\Gamma}(x) = 1 \iff S(\varphi_1) = \cdots = S(\varphi_a) = 0$.

Now using (i) and (ii) we can build an algorithm for $\mathtt{OUTSIDE}_{S,\Gamma}(x)$ when $x = (▲)_a$ and $\Gamma \in \{\mathbf{k}, \mathbb{H}, \mathbb{G}\}^*$. First we solve $\Gamma \overset{?}{\in} \mathbf{k}^p$ for some p. This can be done in k steps. In fact this p must be k, the length of Γ. Since we have Γ decomposed in its factors, we only have to check if all the factors are constant or not. If $\Gamma \notin \mathbf{k}^p$ we are done and $\mathtt{OUTSIDE}_{S,\Gamma}(x) = 1$. If $\Gamma \in \mathbf{k}^p$ then we have to check whether $S(\varphi_1) \overset{?}{=} 0, \dots, S(\varphi_a) \overset{?}{=} 0$, which can be calculated in $|\varphi_1|n + \cdots + |\varphi_a|n = \|\varphi\|n$ steps. So the total time is, in the worst case, $k + \|\varphi\|n$. □

Corollary C.10. *Consider a non-elliptic syntagma S with $|S| = n$, a pattern $\Gamma \in \{\mathbf{k}, \mathbb{H}, \mathbb{G}\}^*$ with arity a, length k and with an unambiguous decomposition. Consider $x \in (\mathrm{Spt}(S) \sqcup \{▲\})^a$ but $x \notin \mathrm{Spt}(S)$. Then $\mathtt{OUTSIDE}_{S,\Gamma}(x)$ can be computed in time $O(n^{a(k-1)+1})$.*

Proof. We use the previous lemmas. Fig.C.3 shows pseudo-code. First we check $x \overset{?}{=} (▲)_a$. This can be solved in a steps. If $x \neq (▲)_a$ then we apply the first Lemma C.8 where the time is $O(n^{a(k-1)+1})$. If not, we apply the second Lemma C.9 where the time is linear $O(n)$. So, in either case, the worst time is $O(n^{a(k-1)+1})$. □

Joining the previous lemmas we are equipped to come back to the problem $S \overset{?}{\in} W$.

Theorem C.11. *Let $W = \mathbf{Synt}\binom{B}{\Gamma}$ be a manifold with $\Gamma \in \{\mathbf{k}, \mathbb{H}, \mathbb{G}\}^*$ with arity a, length k and with an unambiguous decomposition. The problem $S \overset{?}{\in} W$ for a non-elliptic syntagma S can be computed in time $O(n^{ak+1})$.*

Proof. We take the general algorithm from Fig.C.2. The central loop cycles $(n+1)^a$ times. We have two basic computations in this loop. We evaluate $x \overset{?}{\in} \mathrm{Spt}(S)^a$ which can be done in a steps. When $x \in \mathrm{Spt}(S)^a$ in the worst case in each loop we have to evaluate $x \overset{?}{\in} \Gamma$ and $B(\overline{S}(x)) \overset{?}{=} 1$.

By assumption $\Gamma \in \{\mathbf{k}, \mathbb{H}, \mathbb{G}\}^*$; then by Theorem C.3 the evaluation $x \overset{?}{\in} \Gamma$ takes time $O(\|x\|^{a(k-1)+1})$. We can bound $\|x\|$ in terms of the size of S, i.e. n. Consider $x = (x_1, \dots, x_a)$, where $x_1, \dots, x_a \in \mathrm{Spt}(S)$, $|x_i| \leq d$, and where d is the depth of the syntagma. Since $d < n$, where $n = |S|$, $\|x\| = \sum_{i=1}^{a} \|x_i\| \leq ad < an$. So the time is less than $O\big((an)^{a(k-1)+1}\big) = O\big(n^{a(k-1)+1}\big)$.

Regarding the second evaluation $B(\overline{S}(x)) \overset{?}{=} 1$, this is the same as searching for the vector $\overline{S}(x) = (S(x_1), \dots, S(x_n))$ in a finite and fixed set $B^{-1}(1)$. Since $B^{-1}(1) \subseteq \Sigma^a$, the

```
function OUTSIDE
    input $S, \Gamma, x$;
    evaluate $x =^? (\blacktriangle)_a$
    if $x = (\blacktriangle)_a$ then
        calculate $Y$;
        calculate $(x|Y)$;
        if $\forall y \in Y,\ S(y) = 0$ and $(x|Y) \in \Gamma$ then
            return $1$;
        else
            return $0$;
        endif
    else
        $p$ := *length of* $\Gamma$;
        evaluate $\Gamma \in^? \mathbf{k}^p$;
        if $\Gamma \notin \mathbf{k}^p$ then
            return $1$;
        else
            $(\varphi_1, \ldots, \varphi_a) := \Gamma$;
            if $S(\varphi_1) = \cdots = S(\varphi_a) = 0$ then
                return $1$;
            else
                return $0$;
            endif
        endif
    endif
endfunction;
```

Figure C.3: algorithm for the function $\mathtt{OUTSIDE}_{S,\Gamma}(x)$; the set Y is the unique possible set such that $(x|Y) \in \Gamma$.

time is bounded by v^a steps where $v = |\Sigma|$, i.e. constant time.[6] Thus when $x \in \mathrm{Spt}(S)^a$ the computation takes time bounded by $O\big(n^{a(k-1)+1}\big)$.

Consider now $x \notin \mathrm{Spt}(S)^a$; first we have to calculate the function $\mathtt{OUTSIDE}_{S,\Gamma}(x)$ which takes time $O(n^{a(k-1)+1})$ by the last corollary, and second we have to calculate $B\big(\overline{S_\blacktriangle}(x)\big)$ which runs in constant time.[7] So the loop consumes $O(n^{a(k-1)+1})$ steps.

So in both cases the time is $O(n^{a(k-1)+1})$. Since the general loop cycles $(n+1)^a$ times, the total time is $(n+1)^a \times O(n^{a(k-1)+1})$, that is $O(n^{ak+1})$. □

There remains the general problem where syntagmata can be elliptic and patterns can be ambiguous:

Theorem C.12. *Manifolds with type in* $\{\mathbf{k}, \mathbb{H}, \mathbb{G}\}^*$ *and with effectively bounded ellipticity* ε *are decidable. In particular, if* ε *is a polynomial, the time is polynomial.*

Proof. First we generalize the problem for possibly elliptic syntagmata S, with $|S| = n$, depth $d = \mathrm{depth}(S)$, and $e = |\mathrm{Ell}(S)|$ ellipses. By assumption the ellipticity of each syntagma is bounded by a computable function $\varepsilon(n)$, $e \le \varepsilon(n)$. In the previous theorems and lemmas we have used the inequality $d < n$. When the syntagma is elliptic this inequality generalizes as $d < n + e \le n + \varepsilon(n)$ (Lemma 2.6). So all the bounds can be generalized and the bound for the last theorem becomes $O((n+\varepsilon(n))^{ak+1})$. We commented just after Lemma C.7 that it also works for ambiguous patterns, but with time $O(n^{2a(k-1)+2})$, or assuming elliptic syntagmata, $O((n+\varepsilon(n))^{2a(k-1)+2})$. We can remake the calculations of the last proof to obtain the bound $O(n^a(n+\varepsilon(n))^{2a(k-1)+2})$. When ε is a polynomial, this bound is also a polynomial. □

C.4 Time for Manifolds with Type in $\{\mathbf{k}, \mathbb{P}, \mathbb{H}, \mathbb{G}\}^*$

We solve the problem for manifolds with homogeneous ($\mathbb{H}, \mathbb{G}$) and pivoting ($\mathbb{P}$) types with the condition that the valuation is neutral:

Definition C.13. We say that a boolean function is *neutral* iff

$$B(x_1, \dots, x_i, 0, x_{i+1}, \dots, x_a) = 1,$$

for every $i = 1, \dots, a$ and for any $x_1, \dots, x_i, x_{i+1}, \dots, x_a \in \Sigma_+$. That is, if whenever some input is null, B returns the true value.

Theorem C.14. *Let* Γ *be a pattern in* $\{\mathbf{k}, \mathbb{H}, \mathbb{G}, \mathbb{P}\}^*$ *and let* $W = \mathbf{Synt}\binom{B}{\Gamma}$ *be a manifold such that the valuation B is neutral. The problem* $S \in^? W$ *can be solved in time* $O(n^{ak+1})$, *where n is the size of S, a is the arity, and k is the length of the given decomposition of the pattern* Γ.

[6] The value v is constant and thus v^a is constant although large.

[7] We construct the vector with components $S(x_j)$ when $x_j \neq \blacktriangle$ and 0 otherwise, i.e. $\overline{S_\blacktriangle}(x)$, and then evaluate B which is done in constant time v^a.

```
program RECOGNITION_OF_MANIFOLD_WITH_NEUTRAL_VALUATION
    read $S, B, \Gamma$;
    for each $x \in (\mathrm{Spt}(S))^a$ do
        evaluate $x \in^? \Gamma$;
        if $x \in \Gamma$ then
            evaluate $B(\overline{S}(x)) =^? 1$;
            if $B(\overline{S}(x)) = 1$ then continue;
            else print $S \notin W$; haltprogram;
            endif;
        else continue;
        endif;
    endfor;
    print $S \in W$;
endprogram;
```

Figure C.4: algorithm for recognition of a manifold $W = \mathbf{Synt}\binom{B}{\Gamma}$ with B neutral valuation.

Proof. By definition a neutral valuation always returns 1 when some component is null. If we consider the second part in the loop of the GENERAL_ALGORITHM_FOR_ RECOGNITION_OF _MANIFOLD we realize that we do not need to evaluate it for these valuations, because $B\big(\overline{S_\blacktriangle}(x)\big) = 1$ always. So we do not need the function OUTSIDE$_{S,\Gamma}(x)$. Furthermore we can rewrite the general algorithm without the ▲ symbol, see the pseudocode in Fig. C.4. Regarding the time, the loop cycles a times the evaluations $x \in^? \Gamma$ and $B(\overline{S}(x)) =^? 1$. We see in the proof of Theorem C.11 that this takes time $O\big(n^{a(k-1)+1}\big)$, so the total time is $O\big(n^{ak+1}\big)$. □

Example 38. Neutral valuation. We have seen several examples in which the valuation is neutral. In general all the morphological agreements can be taken to be neutral. Consider the gender agreement of a noun and its adjective in Romance languages, given by a valuation $B : \Sigma^2 \longrightarrow \Sigma$. When there is no adjective, the valuation must consider the pair $(noun, 0)$. Since adjectives are optional we must take $B(noun, 0) = 1$, otherwise the syntagma is not valid in the manifold. However, even if the adjective was obligatory one could take $B(noun, 0) = 1$ because it would never be the case. Similar considerations allow us to take $B(0, adjective) = 1$.

However the constraints in a grammar which control valence, or in general the flow of functions in a syntagma, must consider null loci in their valuations. For example a transitive verb disallows a null locus *Ob*, and in consequence the valuation returns the false value when one of the components is null.

List of Figures

List of Tables

List of Formal Languages

$\Sigma \subset \Sigma^*$	alphabet language
$\{a\}$	singleton lng.
Σ^*	trivial or full lng.
$L_{\text{squa}} = \{a_1^2 \cdots a_n^2 \mid a_1, \ldots, a_n \in \Sigma, n \in \mathbb{N}_+\}$	squares lng.
$L_{\text{copy}} = \{xx \mid x \in \Sigma^*\}$	copy lng.
$L_{\text{mirr}} = \{xx^R \mid x \in \Sigma^*\}$	mirror lng.
$L_{\text{mult}} = \{(abc)^n \mid n \in \mathbb{N}_+\}$	multiple *abc* lng.
$L_{\text{resp}} = \{a^n b^n c^n \mid n \in \mathbb{N}_+\}$	respectively *abc* lng.
$L_{q\text{-copy}} = \{x^q \mid x \in \Sigma^*\}$	generalized copy lng.
$L_{2\text{-resp}} = \{a^n b^n \mid n \in \mathbb{N}_+\}$	2-respectivey lng.
$L_{3\text{-resp}} = L_{\text{resp}}$	respectivey lng.
$L_{q\text{-resp}} = \{a_1^n \cdots a_q^n \mid n \in \mathbb{N}_+\}$	generalized respectivey lng.
$L_{q\text{-count}} = \{a_1^n \cdots a_q^n \mid n \in \mathbb{N}_+, a_1, \ldots, a_q \in \Sigma\}$	q-counting lng.
$L_{\text{cros}} = \{a^n b^m c^n d^m \mid n, m \in \mathbb{N}_+\}$	crossed lng.$^{(*)}$
$L_{\text{expo}} = \{a^{2^n - 1} \mid n \in \mathbb{N}_+\}$	exponential growth lng.
$L_{n^2\text{-poly}} = \{a^{n^2} \mid n \in \mathbb{N}_+\}$	quadratic growth lng.
$L_{n^3\text{-poly}} = \{a^{n^3} \mid n \in \mathbb{N}_+\}$	cubic growth lng.
$L_{p(n)\text{-poly}} = \{a^{p(n)} \mid n \in \mathbb{N}_+\}$	polynomial growth lng.
$L_{\text{noin}} = \{(ba^n)^n \mid n \in \mathbb{N}_+\}$	example of non-indexed lng.
$\widetilde{L_{\text{squa}}} = \{a_1 \overline{a}_1 \cdots a_n \overline{a}_n \mid a_i \in \Sigma, \overline{a}_i \in \overline{\Sigma}, n \in \mathbb{N}_+\}$	tilde squares lng.
$\widetilde{L_{\text{copy}}} = \{a_1 \cdots a_n \overline{a}_1 \cdots \overline{a}_n \mid a_i \in \Sigma, \overline{a}_i \in \overline{\Sigma}, n \in \mathbb{N}_+\}$	tilde copy lng.
$L_{\text{Dyck}} =$ well-balanced parentheses $\subseteq (\Sigma \sqcup \overline{\Sigma})^*$	Dick lng.
$L_{\text{Chin}} = \{ab^{k_1} \cdots ab^{k_p} \mid k_1 > \cdots > k_p > 0, p \in \mathbb{N}\}$	Chinese number lng.
$L_{\text{OldG}} = \{\prod_{k=0}^{n} ab^k \mid n \in \mathbb{N}_+\}$	*suffixaufnahme* Old Georgian

$^{(*)}$Sometimes this language is called the *respectively language*, however we have reserved this name for the language $L_{\text{resp}} = \{a^n b^n c^n \mid n \in \mathbb{N}_+\}$.

List of Classes of Languages and Acronyms

Classes of Languages

AB	class of	alphabet languages
TR		trivial lngs or full lngs.
FN		finite lngs.
FN$_0$		finite stressed lngs.
RG		regular lngs.
CF		context-free lngs.
CS		context-sensitive lngs.
NT		natural lngs.
AD		algebraic dependency lngs.
AN		Angluin lngs.
AN$_0$		free-constants Angluin lngs.
PG$_p$		p-polynomial growth unary lngs.
PG		polynomial growth unary lngs.
IN		indexed lngs.
PTIME		P-problems or poly-time decidable lngs.
NPTIME		NP-problems.
Rec		recursive lngs.
RecEnum		recursive enumerable lngs.

List of Acronyms

ADG	algebraic dependency grammars
AMR	abstract meaning representation
CFG	context-free grms.
CSG	context-sensitive grms.
D	displacement calculus
FOL	first order logic
FGD	functional generative grm.
GB	government and binding
GPSG	generalized phrase structure grms.
HPSG	head-driven phrase structure grms.
ING	indexed grms.
L	Lambek calculus
LING	linear indexed grms.
p-LCFRG	p-linear context-free rewriting systems
LCFRG	union of p-linear context-free rewriting systems
LFG	lexical functional grms.
MCS	mildly-context-sensitivity
p-MCFG	p-multiple context-free grms.
MCFG	union of p-multiple context-free grms.
MG	minimalist grms.
MTT	meaning text theory
MP	minimalist program
MSOL	monadic second order logic
MTS	model theoretic syntax
PMCFG	parallel multiple context-free grms.
RCG	range concatenation grms.
RGG	regular grms.
TAG	tree adjoining grms.
TDG	topological dependency grm.
TG	transformational grm.
XDG	extensible dependency grm.

Bibliography

Adámek, J., Herrlich, H. and Strecker, G. E. (2004), 'Abstract and concrete categories: The joy of cats'. `http://katmat.math.uni-bremen.de/acc`[Accessed: 05/6/2017].

Adjukiewicz, K. (1935), 'Die syntaktische konnexität', *Studia Philosophica* **1**, 1–27.

Aho, A. V. (1968), 'Indexed grammars, an extension of context-free grammars', *Journal of the ACM (JACM)* **15**(4), 647–671.

Alfonseca Cubero, E., Alfonseca Moreno, M. and Moriyón Salomón, R. (2007), *Teoría de autómatas y lenguajes formales*, Mc-Graw-Hill/Interamericana de España, Madrid.

Angluin, D. (1980*a*), 'Finding patterns common to a set of strings', *Journal of Computer and System Sciences* **21**(1), 46–62.

Angluin, D. (1980*b*), 'Inductive inference of formal languages from positive data', *Information and control* **45**(2), 117–135.

Arora, S. and Barak, B. (2009), *Computational complexity: a modern approach*, Cambridge University Press, New York.

Autebert, J.-M., Berstel, J. and Boasson, L. (1997), Context-free languages and pushdown automata, *in* G. Rozenberg and A. Salomaa, eds, 'Handbook of formal languages: Word, Language, Grammar', Vol. 1, Springer-Verlag, Berlin, Heildelberg, pp. 111–174.

Bach, E. (1981), Discontinuous constituents in generalized categorial grammar, *in* 'Proceedings of the 11th Annual Meeting of the North Eastern Linguistics Society', GLSA Publications, Department of Linguistics, University of Massachussets at Amherst, New York, pp. 1–12.

Bach, E. (1988), Categorial grammars as theories of language, *in* R. T. Oehrle, E. Bach and D. Wheeler, eds, 'Categorial grammars and natural language structures', Vol. 32 of *Studies in Linguistics and Philosophy*, Springer, Netherlands, pp. 17–34.

Bach, E. and Miller, P. (2003), Generative capacity, *in* 'The International Encyclopedia of Linguistics', Vol. 3, Oxford University Press, Oxford, pp. 20–21.

Bar-Hillel, Y. (1953), 'A quasi-arithmetical notation for syntactic description', *Language* **29**, 47–58.

Bar-Hillel, Y. and Shamir, E. (1960), 'Finite-state languages: formal representations and adequacy problems', *Bull. of Res. Council of Israel* **8F**, 155–166.

Barton, G. E., Berwick, R. C. and Ristad, E. S. (1987), *Computational complexity and natural language*, MIT press, Massachusetts.

Basart, J. M. (2003), *Grafs: fonaments i algorismes*, Vol. 13 of *Manuals de la Universitat Autònoma de Barcelona*, Universitat Autònoma de Barcelona, Barcelona.

Becker, T., Rambow, O. and Niv, M. (1992), The derivational generative power of formal systems or scrambling is beyond LCFRS, Technical Report IRCS-92-38, University of Pennsylvania, Institute for Research in Cognitive Science.

Berstel, J. (1979), *Transductions and context-free languages*, Vol. 38 of *Leitfäden der angewandten Mathematik und Mechanik, Teubner Studienbücher: Informatik*, Teubner, Stuttgart.

Birkhoff, G. (1940), *Lattice theory*, Vol. 25 of *Colloquium publications*, American Mathematical Soc., Providence.

Blum, N. and Koch, R. (1999), 'Greibach normal form transformation revisited', *Information and Computation* **150**(1), 112–118.

Bordihn, H., Holzer, M. and Kutrib, M. (2009), 'Determination of finite automata accepting subregular languages', *Theoretical Computer Science* **410**(35), 3209–3222.

Boullier, P. (1999), Chinese numbers, mix, scrambling, and range concatenation grammars, *in* 'Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics (EACL'99), Bergen, Norway, June 1999', pp. 53–60.

Boullier, P. (2004), Range concatenation grammars, *in* H. Bunt, J. Carroll and G. Satta, eds, 'New Developments in Parsing Technology', Vol. 23 of *Text, Speech and Language Technology*, Springer, Dordrecht, pp. 269–280.

Bresnan, J., Kaplan, R. M., Peters, S. and Zaenen, A. (1987), Cross-serial dependencies in Dutch, *in* 'The formal complexity of natural language', D. Reidel, Dordrecht, pp. 286–319.

Bröker, N. (1998), Separating surface order and syntactic relations in a dependency grammar, *in* 'Proceedings of the 17th International Conference on Computational Linguistics (COLING'98), Montreal, Quebec, Canada, August 1998', Vol. 1, pp. 174–180.

Bröker, N. (2000), 'Unordered and non-projective dependency grammars', *TAL. Traitement automatique des langues* **41**(1), 245–272.

Cardó, C. (2016), Algebraic governance and symmetry in dependency grammars, *in* A. Foret, G. Morrill, R. Muskens, R. Osswald and S. Pogodalla, eds, '20th and 21st International Conferences on Formal Grammar, (FG'15), revised Selected Papers, Barcelona, Catalunya, August 2015, (FG'16), proceedings, Bozen, Italy, August 2016', Vol. 9804 of *LNCS, FoLLI Publications in Logic, Language and Information*, Springer, Berlin, Heidelberg, pp. 60–76.

Charles University, Czech Republic Faculty of Mathematics and Physics (2017), '*Institute of Formal and Applied Linguistics*', `https://ufal.mff.cuni.cz/data/semantics-pragmatics` [Accessed: 21/3/2017].

Chomsky, N. (1956), 'Three models for the description of language', *IRE Transactions on information theory* **2**(3), 113–124.

Chomsky, N. (1957), *Syntactic structures*, Vol. 4 of *Janua linguarum, Series minor*, Mouton & Co.

Chomsky, N. (1965), *Aspects of the Theory of Syntax*, Vol. 11, MIT press, Massachusetts.

Chomsky, N. (1995), *The minimalist program*, Vol. 28 of *Current studies in linguistic series*, MIT Press, Massachusetts.

Chomsky, N. and Schützenberger, M. P. (1963), 'The algebraic theory of context-free languages', *Studies in Logic and the Foundations of Mathematics* **35**, 118–161.

Clark, A. and Yoshinaka, R. (2012), Beyond semilinearity: Distributional learning of parallel multiple context-free grammars, *in* J. Heinz, C. Higuera and T. Oates, eds, 'Proceedings of the 11th International Conference on Grammatical Inference (ICGI'12), Washington, D.C., USA, September 2012', Vol. 21 of *Proceedings of Machine Learning Research*, PMLR, University of Maryland, College Park, MD, USA, pp. 84–96.

Comellas, F., Fabrega, J., Lladó, A. and Serra, O. (1996), *Matemàtica discreta*, Vol. 26 of *Politext*, Edicions UPC, Barcelona.

Comon, H., Dauchet, M., Gilleron, R., Löding, C., Jacquemard, F., Lugiez, D., Tison, S. and Tommasi, M. (2007), 'Tree automata techniques and applications'. `http://tata.gforge.inria.fr/` [Accessed: 05/6/2017].

Cronin, J. and Reisse, J. (2005), 'Chirality and the origin of homochirality', *Lectures in astrobiology* pp. 473–515.

Davey, B. A. and Priestley, H. A. (2002), *Introduction to lattices and order*, Cambridge university press, New York.

Davis, T. (2006), 'Catalan numbers'. `http://www.geometer.org/mathcircles/` [Accessed: 05/6/2017].

de Cornulier, B. (1973), 'But if "respectively" meant something?', *Research on Language & Social Interaction* **6**(1-4), 131–134.

Debusmann, R. (2000), 'An introduction to dependency grammar'. Hausarbeit fur das Hauptseminar Dependenzgrammatik SoSe 99, Saarbrücken: Universitat des Saarlandes.

Debusmann, R. (2007*a*), The complexity of first-order extensible dependency grammar, Technical report, Saarland University.

Debusmann, R. (2007*b*), Extensible Dependency Grammar: a modular grammar formalism based on multigraph description, PhD thesis, Universität des Saarlandes.

Debusmann, R. and Duchier, D. (2002*a*), Topological dependency analysis of the Dutch verb cluster, Technical Report P0315, Saarland University.

Debusmann, R. and Duchier, D. (2002*b*), 'Towards a syntax-semantics interface for topological dependency grammar'. `https://www.univ-orleans.fr/lifo/membres/duchier/papers/duchier-dg-synsem.pdf`[Accessed: 12/6/2017].

Debusmann, R., Duchier, D., Koller, A., Kuhlmann, M., Smolka, G. and Thater, S. (2004), A relational syntax-semantics interface based on dependency grammar, *in* 'Proceedings of the 20th international conference on Computational Linguistics (COLING'04), Geneva, Switzerland, August 2004', pp. 176–182.

Debusmann, R., Duchier, D. and Kruijff, G.-J. M. (2004), Extensible dependency grammar: A new methodology, *in* 'Proceedings of the Workshop on Recent Advances in Dependency Grammar (COLING'04), Geneva, Switzerland, August 2004', pp. 78–85.

Debusmann, R. and Kuhlmann, M. (2010), Dependency grammar: Classification and exploration, *in* M. Crocker and J. Siekmann, eds, 'Resource-adaptive cognitive processes', Cognitive Technologies, Springer, Berlin, Heilderberg, pp. 365–388.

Debusmann, R., Postolache, O. and Traat, M. (2005), A modular account of information structure in extensible dependency grammar, *in* A. Gelbukh, ed., 'Computational Linguistics and Intelligent Text Processing (CICLing'05), Mexico City, Mexico, February 2005', Vol. 3406 of *LNCS*, Springer, Berlin, Heilderberg, pp. 25–36.

Duchier, D. (2004), 'Lexicalized syntax and topology for non-projective dependency grammar', *Electronic Notes in Theoretical Computer Science* **53**, 70–80.

Duchier, D. and Debusmann, R. (2001), Topological dependency trees: A constraint-based account of linear precedence, *in* 'Proceedings of the 39th Annual Meeting on Association for Computational Linguistics (ACL'01), Toulouse, France, July 2001', pp. 180–187.

Edlefsen, M., Leeman, D., Myers, N., Smith, N., Visscher, M. and Wellcome, D. (2008), Deciding strictly local (SL) languages, *in* 'Proceedings of the midstates conference for undergraduate research in computer science and mathematics (MCURCSM'08)', Vol. 6, pp. 6–75.

Engelfriet, J. (2015), 'Tree automata and tree grammars', arXiv preprint arXiv:1510.02036.

Eroms, H.-W. and Heringer, H. J. (2003), 'Dependenz und lineare ordnung', *Dependenz und Valenz/Dependency and valency: Ein internationales Handbuch der zeitgenössischen Forschung/An international handbook of contemporary research* **25**, 247–263.

Flanigan, J., Thomson, S., Carbonell, J. G., Dyer, C. and Smith, N. A. (2014), A discriminative graph-based parser for the abstract meaning representation, *in* 'Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL'14), Baltimore, USA', Vol. 1: Long Papers, pp. 1426–1436.

Gaifman, H. (1965), 'Dependency systems and phrase-structure systems', *Information and control* **8**(3), 304–337.

Gazdar, G. (1988), Applicability of indexed grammars to natural languages, *in* U. Reyle and C. Rohrer, eds, 'Natural language parsing and linguistic theories', Vol. 35 of *Studies in Linguistics and Philosophy*, Springer, Netherlands, pp. 69–94.

Gerdes, K. and Kahane, S. (2001), Word order in German: A formal dependency grammar using a topological hierarchy, *in* 'Proceedings of the 39th annual meeting on association for computational linguistics (ACL'01), Toulouse, France, July 2001', pp. 220–227.

Gerdes, K. and Kahane, S. (2015), Non-constituent coordination and other coordinative constructions as dependency graphs, *in* ‘Proceedings of the third international conference on dependency linguistics (DEPLING’15), Uppsala, Sweden, August 2015’, pp. 101–110.

Gilman, R. H. (1996), ‘A Shrinking Lemma for Indexed Languages’, *Theoretical Computer Science* **163**, 277–281.

Gladkij, A. and Mel’čuk, I. (1975), ‘Tree grammars. I. A formalism for syntactic transformations in natural languages’, *Linguistics* **13**(150), 47–82.

Gold, E. M. (1967), ‘Language identification in the limit’, *Information and control* **10**(5), 447–474.

Grätzer, G. (1968), *Universal algebra*, The university series in higher mathematics, D. Van Nostrand Co., New York.

Gross, M. (1964), ‘On the equivalence of models of language used in the fields of mechanical translation and information retrieval’, *Information storage and retrieval* **2**(1), 43–57.

Groß, T. (2011), Transformational grammarians and other paradoxes, *in* ‘5th International Conference on Meaning-Text Theory (ICMTT’11), Barcelona, Spain, September 2011’, pp. 88–97.

Groß, T. and Osborne, T. (2009), ‘Toward a practical dependency grammar theory of discontinuities’, *SKY Journal of Linguistics* **22**, 43–90.

Groß, T. and Osborne, T. (2015), The dependency status of function words: Auxiliaries, *in* ‘Proceedings of the third international conference on dependency linguistics (DEPLING’15), Uppsala, Sweden, August 2015’, pp. 111–120.

Hayashi, T. (1973), ‘On derivation trees of indexed grammars: an extension of the $uvwxy$-theorem’, *Publications of The Research Institute for Mathematical Sciences* **9**, 61–92.

Hays, D. G. (1958), Grouping and dependency theories, *in* H. Edmunson, ed., ‘Proceedings of the national symposium on machine translation, Los Angeles, California’, University of California Los Angeles, UCLA, Prentice-Hall, London, pp. 258–266.

Hays, D. G. (1964), ‘Dependency theory: A formalism and some observations’, *Language* pp. 511–525.

Heinz, J., Rawal, C. and Tanner, H. G. (2011), Tier-based strictly local constraints for phonology, *in* 'Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (HLT'11), Portland, Oregon, June 2011', Vol. 2, pp. 58–64.

Hopcroft, J. E., Raajeev, M. and Ullman, J. D. (2001), *Introduction to automata theory, languages, and computation*, Pearson education, 2nd edn, Addison Wesley.

Hopcroft, J. E. and Ullman, J. D. (1979), *Introduction to automata theory, languages, and computation*, Series in computer science, Addison Wesley.

Huddleston, R., Pullum, G. K. et al. (2002), *The Cambridge grammar of the English language*, Cambridge University Press, New York.

Hudson, R. (2000), 'Discontinuity', *Traitement automatique des langues* **41**(1), 15–56.

Hudson, R. A. (1990), *English word grammar*, B. Blackwell, Oxford.

Joshi, A. K. (1985), Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions?, *in* D. Dowty, L. Karttunen and A. Zwicky, eds, 'Natural Language Parsing', Cambridge University Press, pp. 206–250.

Kac, M. B., Manaster-Ramer, A. and Rounds, W. C. (1987), 'Simultaneous-distributive coordination and context-freeness', *Computational Linguistics* **13**(1-2), 25–30.

Kahane, S. (1997), Bubble trees and syntactic representations, *in* 'Proceedings of the 5th Meeting of the Mathematics of Language (MOL'97), Saarbrücken, Germany', pp. 70–76.

Kahane, S. and Mazziotta, N. (2015), Dependency-based analyses for function words – introducing the polygraphic approach, *in* 'Proceedings of the third international conference on dependency linguistics (DEPLING'15), Uppsala, Sweden, August 2015', pp. 181–190.

Kahane, S., Nasr, A. and Rambow, O. (1998), Pseudo-projectivity: a polynomially parsable non-projective dependency grammar, *in* 'Proceedings of the 17th international conference on Computational linguistics (COLING'98), Montreal, Quebec, Canada, August 1998', Vol. 1, pp. 646–652.

Kallmeyer, L. (2010), *Parsing beyond context-free grammars*, Springer, Berlin, Heilderberg.

Kanazawa, M. (2017), 'Formal grammar: An introduction'. Lecture notes, course mathematical linguistics, Sokendai, `http://research.nii.ac.jp/~kanazawa/FormalGrammar/` [Accessed: 05/6/2017].

Kanazawa, M. and Salvati, S. (2012), MIX is not a tree-adjoining language, *in* 'Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL'12), Jeju, Korea, July 2012', Vol. 1, pp. 666–674.

Kaplan, R. M. and Bresnan, J. (1982), Lexical-functional grammar: A formal system for grammatical representation, *in* M. Dalrymple, R. M. Kaplan, J. T. Maxwell III and A. Zaenen, eds, 'Formal Issues in Lexical-Functional Grammar', Standford University, Standford, pp. 29–130.

Kelley, D. (1995), *Automata and Formal Languages: an introduction*, Prentice-Hall, Inc., New Jersey.

Kelvin, W. T. B. (1904), *Baltimore lectures on molecular dynamics and the wave theory of light*, CJ Clay and Sons, London.

Kessler, B. (1995), 'Discontinuous constituents in Latin'. `http://www.artsci.wustl.edu/~bkessler/latin-discontinuity/discontinuity.ps.`[Accessed: 05/6/2017].

Knuutila, T. (1993), Inference of k-testable tree languages, *in* H. Bunke, ed., 'Advances in Structural and Syntactic Pattern Recognition: proceedings of the international workshop (SSPR'92), Bern, Switzerland, August 1992', Vol. 5 of *Machine perception artificial intelligence*, World Scientific, Singapore, pp. 109–120.

Kobele, G. M. (2006), Generating Copies: An investigation into structural identity in language and grammar, PhD thesis, University of California, Los Angeles.

Koller, A. (2015), Semantic construction with graph grammars, *in* 'Proceedings of the 11th International Conference on Computational Semantics (IWCS'15), London, United Kingdom, April 2015', pp. 228–238.

Kruijff, G.-J. M. (2002), 'Formal and computational aspects of dependency grammar: Historical development of DG'. Lecture notes, computational linguistics, university of Saarland, Saarbrücken Germany, `https://www.infoamerica.org/documentos_pdf/bar03.pdf`[Accessed: 06/1/2018].

Kuhlmann, M. (2010), *Dependency Structures and Lexicalized Grammars: An Algebraic Approach*, Vol. 6270, Springer. Published PhD thesis.

Kuhlmann, M. (2013), 'Mildly non-projective dependency grammar', *Computational Linguistics* **39**(2), 355–387.

Kuhlmann, M. and Jonsson, P. (2015), 'Parsing to non-crossing dependency graphs', *Transactions of the Association for Computational Linguistics* **3**, 559–570.

Kuhlmann, M. and Möhl, M. (2007), Mildly context-sensitive dependency languages, *in* 'proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL'07), Prague, Czech Republic, June 2007', Vol. 45, pp. 160–167.

Kuhlmann, M. and Nivre, J. (2006), Mildly non-projective dependency structures, *in* 'Proceedings of the COLING/ACL on Main conference poster sessions (COLING/ACL'06), Sydney, Australia, July 2006', pp. 507–514.

Lambek, J. (1958), 'The mathematics of sentence structure', *American mathematical monthly* **65**, 154–170.

Lang, S. (2004), *Algebra*, Vol. 211 of *Graduate Texts in Mathematics*, Springer-Verlag, New York.

Lecerf, Y. (1961), 'Une représentation algébrique de la structure des phrases dans diverses langues naturelles', *Comptes rendus hebdomadaires des seances de l'academie des sciences de Paris* **252**(2), 232–234.

Lison, P. (2006), 'Bubble trees: A dependency grammar approach to coordination'. `http://www.dfki.de/~plison/pubs/talks`[Accessed: 05/6/2017].

Mairal, R. and Gil, J. (2004), *Entorno a los universales lingüísticos*, Ediciones Akal, Madrid.

Marcus, S. (1967), *Algebraic Linguistics; Analytical Models*, Vol. 29 of *Mathematics in Science and Engineering*, Academic Press, New York.

Matthews, P. (1981), *Syntax*, Cambridge University Press, New York.

Melčuk, I. A. (1988), *Dependency Syntax: Theory and practice*, SUNNY series in linguistics, State University of New York Press, New York.

Melčuk, I. A. (2009), *Dependency in linguistic description*, Vol. 111 of *Studies in language companion series*, John Benjamins Publishing Co., Amsterdam, pp. 1–101.

Michaelis, J. and Kracht, M. (1997), Semilinearity as a syntactic invariant, *in* 'Logical aspects of computational linguistics', Vol. 1328 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, pp. 329–345.

Morrill, G. (2007), 'A chronicle of type logical grammar: 1935–1994', *Research on Language and Computation* **5**(3), 359–386.

Morrill, G. (2011), *Categorial grammar: Logical syntax, semantics, and processing*, Oxford University Press, Oxford.

Morrill, G., Valentín, O. and Fadda, M. (2011), 'The displacement calculus', *Journal of Logic, Language and Information* **20**(1), 1–48.

Nivre, J. (2005), Dependency grammar and dependency parsing, Technical Report MSI 05133, Växjö University: School of Mathematics and Systems Engineering.

Obrebski, T. and Gralinski, F. (2004), Some notes on generative capacity of dependency grammar, *in* 'Workshop on Recent Advances in Dependency Grammar (COLING'04), Geneve, Switzerland, August 2004', pp. 65–71.

Orr, M. A. (1807), *Dante and the early astronomers*, A. Wingate, London.

Osborne, T. (2013), The distribution of floating quantifiers a dependency grammar analysis, *in* 'Proceedings of the second international conference on dependency linguistics (DEPLING'13), Prague, Czech Republic, August 2013', pp. 272–281.

Osborne, T. (2015), Diagnostics for constituents: Dependency, constituency, and the status of function words, *in* 'Proceedings of the third international conference on dependency linguistics (DEPLING'15), Uppsala, Sweden, August 2015', pp. 251–260.

Osborne, T. and Kahane, S. (2015), Translating Tesnière elements into English. In the third international conference on dependency linguistics (DEPLING'15), Uppsala, Sweden, August 2015, `http://depling.org/depling2015/`[Accessed: 05/6/2017].

Osborne, T., Putnam, M. and Groß, T. (2012), 'Catenae: Introducing a novel unit of syntactic analysis', *Syntax* **15**(4), 354–396.

Parikh, R. J. (1966), 'On context-free languages', *Journal of the ACM (JACM)* **13**(4), 570–581.

Phillips, C. (2003), 'Linear order and constituency', *Linguistic inquiry* **34**(1), 37–90.

Pin, J. É. (2016), 'Mathematical foundations of automata theory'. Lecture notes LIAFA, Université Paris, `https://www.irif.fr/~jep/PDF/MPRI/MPRI.pdf` [Accessed: 05/6/2017].

Popel, M., Marecek, D., Stepánek, J., Zeman, D. and Zabokrtskỳ, Z. (2013), Coordination structures in dependency treebanks., *in* 'Proceedings of the 13rd conferences of the association for computational linguistics (ACL'13), Sofia, Bulgaria, August 2013', pp. 517–527.

Pullum, G. K. (2007), 'The evolution of model-theoretic frameworks in linguistics', *Model-theoretic syntax at* **10**, 1–10.

Pullum, G. K. and Gazdar, G. (1982), ‘Natural languages and context-free languages’, *Linguistics and Philosophy* **4**(4), 471–504.

Pullum, G. K. and Scholz, B. C. (2001), On the distinction between model-theoretic and generative-enumerative syntactic frameworks, *in* P. de Groote, G. Morrill and C. Retoré, eds, ‘Proceedings of the Logical Aspects of Computational Linguistics: 4th International Conference (LACL’01), Le Croisic, France, June 2001’, Vol. 2099 of *LNCS*, Springer, Berlin, Heidelberg, pp. 17–43.

Radzinski, D. (1991), ‘Chinese number-names, tree adjoining languages, and mild context-sensitivity’, *Computational Linguistics* **17**(3), 277–299.

Rico-Juan, J. R., Calera-Rubio, J. and Carrasco, R. C. (2005), ‘Smoothing and compression with stochastic k-testable tree languages’, *Pattern Recognition* **38**(9), 1420–1430.

Robinson, J. J. (1970), ‘Dependency structures and transformational rules’, *Language* **46**, 259–285.

Rogers, J. (1996), A model-theoretic framework for theories of syntax, *in* ‘Proceedings of the 34th annual meeting on Association for Computational Linguistics (ACL’96), Santa Cruz, California, 1996’, pp. 10–16.

Rogers, J. (2003), ‘wMSO theories as grammar formalisms’, *Theoretical Computer Science* **293**(2), 291–320.

Rogers, J. and Hauser, M. D. (2010), ‘The use of formal language theory in studies of artificial language learning: A proposal for distinguishing the differences between human and nonhuman animal learners’, *Studies in Generative Grammar* **104**, 213–232.

Rogers, J., Heinz, J., Fero, M., Hurst, J., Lambert, D. and Wibel, S. (2013), Cognitive and sub-regular complexity, *in* G. Morrill and M. Nederhof, eds, ‘Proceedings of the 17th and 18th International Conferences, (FG’12), revised Selected Papers, Opole, Poland, August 2012, (FG’13), Düsseldorf, Germany, August 2013’, Vol. 8036 of *LNCS, FoLLI Publications in Logic, Language and Information*, Springer, Berlin, Heidelberg, pp. 90–108.

Rogers, J. and Nordlinger, R. (1998), *A descriptive approach to language-theoretic complexity*, MIT Press, Massachusetts.

Ruiz, J., España, S. and Garcia, P. (1998), Locally threshold testable languages in strict sense: Application to the inference problem, *in* ‘Grammatical Inference, proceedings of the 4th International Colloquium (ICGI’98), Ames, Iowa, USA, July 1998’, Vol. 1433 of *Lecture Notes in Artificial Intelligence*, Springer, Berlin, Heilderberg, pp. 150–161.

Sakarovitch, J. (2009), *Elements of Automata Theory*, Cambridge University Press, Cambridge.

Salvati, S. (2011), 'MIX is a 2-MCFL'. Lecture in Multiple Context-free Grammars, Course 5, INRIA Bordeaux Sud-Ouest, `http://www.labri.fr/perso/salvati/downloads/cours/esslli/course5.pdf`[Accessed: 05/6/2017].

Sankappanavar, H. P. and Burris, S. (1981), *A course in universal algebra*, Vol. 78 of *Graduate Texts Math*.

Searle, J. R. (1980), 'Minds, brains, and programs', *Behavioral and brain sciences* **3**(03), 417–424.

Seki, H., Matsumura, T., Fujii, M. and Kasami, T. (1991), 'On multiple context-free grammars', *Theoretical Computer Science* **88**(2), 191–229.

Sgall, P., Nebeský, L., Goralčíková, A. and Hajičová, E. (1969), *A functional approach to syntax in generative description of language*, Elsevier Science B. V., Elsevier, Amsterdam.

Shieber, S. M. (1985), 'Evidence against the context-freeness of natural language', *Linguistics and philosophy* **8**, 333–343.

Solà, J., Lloret, M. R., Mascaró, J. and Pérez Saldanya, M., eds (2010), *Gramàtica del català contemporani: Sintaxi (1-16)*, Vol. II, Empúries, Barcelona.

Stabler, E. (1997), Derivational minimalism, *in* C. Retoré, ed., 'Logical Aspects of Computational Linguistics. First International Conference (LACL'96), Nancy, France, September 1996', Vol. 1328 of *LNCS*, Springer, Berlin, Heidelberg, pp. 68–95.

Steedman, M. (1985), 'Dependency and coordination in the grammar of Dutch and English', *Language* **61**(3), 523–568.

Straka, M., Hajic, J. and Straková, J. (2016), UD-pipe: Trainable pipeline for processing CoNLL-U files performing tokenization, morphological analysis, POS tagging and parsing, *in* 'Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16), Portorož, Slovenia, May 2016', pp. 4290–4297.

Temperley, D. (2005), 'The dependency structure of coordinate phrases: A corpus approach', *Journal of psycholinguistic research* **34**(6), 577–601.

Temperley, D., Sleator, D. and Lafferty, J. (1993), Parsing English with a link grammar, *in* 'Proceedings of the third International Workshop on Parsing Technologies (IWPT'93), Tilburg, The Netherlands and Durbuy, Belgium, August 1993', pp. 91–105.

Tesnière, L. (1959), *Eléments de syntaxe structurale*, Klincksieck, Paris.

The Stanford NLP Group (2017), '*Stanford Dependencies*'. `https://nlp.stanford.edu/software/stanforddependencies.shtml` [Accessed: 21/3/2017].

Thomas, W. (1997), *Languages, Automata, and Logic*, Springer, Berlin, Heidelberg, pp. 389–455.

Tosik, M. (2015), 'Abstract meaning representation, a survey'. Seminar: Dependency Parsing, Department of Linguistics, University of Potsdam, `http://www.melanietosik.com/files/amr.pdf`[Accessed: 7/6/2017].

Vijay-Shanker, K. and Weir, D. J. (1994), 'The equivalence of four extensions of context-free grammars', *Theory of Computing Systems* **27**(6), 511–546.

Vijay-Shanker, K., Weir, D. J. and Joshi, A. K. (1987), Characterizing structural descriptions produced by various grammatical formalisms, *in* 'Proceedings of the 25th annual meeting on Association for Computational Linguistics (ACL'87), Stanford, California, July 1987', pp. 104–111.

Weir, D. J. (1988), Characterizing mildly context-sensitive grammar formalisms, PhD thesis, University of Pennsylvania.

Weir, D. J. and Joshi, A. K. (1988), Combinatory categorial grammars: Generative power and relationship to linear context-free rewriting systems, *in* 'Proceedings of the 26th annual meeting on Association for Computational Linguistics (ACL'88), Buffalo, New York, June 1988', pp. 278–285.

Yli-Jyrä, A. M. et al. (2003), Multiplanarity – a model for dependency structures in treebanks, *in* 'Proceedings of the 2nd Workshop on Treebanks and Linguistic Theories (TLT 2), Växjö, Sweden, November 2003', pp. 189–200.

Les paraules seran sempre nostres!
clap!, clap!, clap!, clap!, clap!
clap!, clap!, clap!, clap!