RECURSION IN COGNITION: A COMPUTATIONAL INVESTIGATION INTO THE REPRESENTATION AND PROCESSING OF LANGUAGE

David James Lobina Bona

Dipòsit Legal: T. 726-2012

RECURSION IN COGNITION: A Computational Investigation into the Representation and Processing of Language

DOCTORAL THESIS FOR THE
COGNITIVE SCIENCE AND LANGUAGE PROGRAMME

David James Lobina Bona
Department of Psychology
University Rovira i Virgili

February, 2012

David James Lobina Bona

Recursion in Cognition: a Computational Investigation into the Representation and Processing of Language

Doctoral Thesis

Supervised by José Eugenio García-Albea Ristol

and Manuel García-Carpintero Sánchez-Miguel

Department of Psychology

Tarragona

2012

UNIVERSITAT
ROVIRA I VIRGILI
DEPARTAMENT DE PSICOLOGIA
http://psico.fcep.urv.es

Carretera de Valls, s/n
43007 Tarragona
Tel. +34 977 55 80 75
Fax +34 977 55 80 88
a/e: sdpsico@urv.cat

HAGO CONSTAR que el presente trabajo, titulado "Recursion in cognition: a computational investigation into the representation and processing of language", que presenta David James Lobina Bona para la obtención del título de Doctor, ha sido realizado bajo mi dirección en el Departamento de Psicología de esta Universidad y que cumple los requisitos para poder optar a la Mención Europea.

Tarragona, 10 de febrero de 2012

El director de la tesis doctoral

[illegible]

Dr. José Eugenio García-Albea Ristol

Co-director de la tesis

[illegible]

Dr. Manuel García-Carpintero Sánchez-Miguel

To my parents, my sister, my cousins, my aunt, my Italian family; to Tilly; to Mark, Amy, Eric, Jo, Helen, Mario, Paloma, Manuel; to José Díaz, Raúl, Fran, Javier, Alberto, Alafont, Ricardo, Miguel Ángel, Elia, Carolina, Rosa, Elena; to José Eugenio, Josep, José Manuel, Teresa, Marc, Daniel, Roger, Felipe, Mari; and to Robin.

"I said just an instant ago that only Champollion would know how to crack such a conundrum", says Augustus sadly. "But now I doubt if Champollion could pull it off. A Chomsky might in a pinch, though".

Contents

List of Figures

List of Tables

Preface

The following material constitutes a thesis for a doctoral degree in *Cognitive Science and Language*, to be submitted at the psychology department of the University Rovira i Virgili (with the participation of the philosophy department at the University of Barcelona). It was financed, in part, by three AGAUR grants (2010 BE 00618, 2010 FI-B2 00013, 2009 SGR 401), a grant from the Spanish Ministry of Education (SEJ 2006 11955), and by an award from the Anglo-Spanish Society in London (for which Mr Albert Jones deserves a special mention). Some of this funding was employed to visit three overseas departments, where I was able to share my ideas and advance my studies; namely, the Rutgers University Center for Cognitive Science (Autumn-Winter 2009), the Research Center for English and Applied Linguistics at the University of Cambridge (Spring 2011), and the Center for General Linguistics (ZAS) in Berlin (Spring-Summer 2011).

The thesis incorporates material that I worked on several years ago (before the beginning of the PhD programme), as well as sections from recent publications that reported some of the early findings of the thesis. The bulk of this PhD, however, is the result of the study I have conducted in conjunction with my supervisors and departmental colleagues in the last four or so years. Naturally, I am indebted to the many individuals who discussed my work with me, including colleagues, conference participants, journal referees and acquaintances. In particular, my supervisor Dr José E. García-Albea and Mr Mark Brenchley deserve a special mention for their extensive comments, suggestions and discussions on the content and style of the entire dissertation. Finally, I have also drawn inspiration from the work of many other individuals, a list of which can be found in alphabetical order in the very last section of this thesis.

This manuscript was set by the author using the LaTeXtypesetting system.

DJL
London-Tarragona-Barcelona
2007-12

Introduction

The present manuscript defends a certain way of studying cognition, one that focuses on providing an account of a given mental phenomenon at the different levels of analysis that David Marr (1982) identified some 30 years ago; hence, the subtitle of this work. The investigation presented here, then, starts with the *computational* level —the study of the mapping of some structures into other structures and the formal properties that derive therefrom (an abstract analysis known as the "theory of the computation" in theoretical computer science)— and subsequently advances to a study of how this mapping is effected in the real-time implementation of an algorithm (that is, the *algorithmic* level, akin to "applied computer science").

Thus, a progression from an entirely theoretical matter to a theoretically-driven empirical investigation will be delineated herein in order to provide a unified account of the topic at hand: the role of recursion in cognitive science. To be more precise, I will mainly focus on the linguistic capacity, but many issues related to general cognition will also be discussed. The overall aims are the following: to delimit the place of recursion within the computational system underlying the *faculty of language*; to work out the role recursion has in the operations of the syntactic parser; and to describe the recursive nature of the structures the mind appears to have and use.

A study of these characteristics gives rise to what I will call "conceptual issues" in the study of cognition; that is, questions which typically involve the theoretical concepts proposed to explain a given phenomenon (including the axioms interconnecting them); a fortiori, such a study involves a purely theoretical problem whose resolution is paramount if we are to gain a coherent understanding of cognitive matters. These sort of issues have usually preoccupied the philosopher, which perhaps should put to rest the perennial debate of what philosophy can offer to cognitive science —Dupuy (2009), for instance, insists that the debate between classical and connectionist architectures will only ever be settled by philosophy (for some discussion of these issues, see the journal TOPICS in Cognitive Science, volume 1, issues 2 and 3). Indeed, the work presented here is mostly an exercise in the philosophy of cognitive science, but there will be enough material to engage both the linguist and the psychologist, and perhaps others.

In order to explain what I mean by "conceptual issues" in the study of cognition, I will in this introduction briefly focus on the relationship between language and thought, which will be of some relevance later on. To begin with, there is an intuitive sense in which thought must exist without language, for the latter seems to constantly underdetermine and misrepresent the former. Indeed, ambiguous sentences, paraphrases and deictic reference are some of the ways in which language underspecifies thought. This lack of transparency between language and thought, however, is only apparent for whoever interprets the linguistic output (a listener, a reader, ecc.), and can never be so for whoever is entertaining the corresponding thoughts. That is, while the linguistic vehicle employed to convey a specific thought may be ambiguous or not specific enough, the underlying thought does not suffer from these maladies.

This intuition appears to be supported by empirical evidence. Perceptual integration and categorisation (even in pre-verbal infants) are some of the phenomena that involve conceptualisation without language (see, for instance, some of the data reported in Gleitman & Papafragou 2005). Indeed, the ability to merge information form different modalities in the fixation of belief (the raison d'être of thought) seems to clearly call for an amodal representation system; a "language of thought", in the sense of Fodor (1975).

This is not to deny that language may be employed, and it probably is quite often, *for* (the aid of) thought. For instance, language can be used to make our thought processes explicit, as in the conscious decision-making so characteristic of considered action, something that is sometimes conducted in "inner speech" (cf. Jackendoff 1997, ch. 8). Nor is it to deny that many instances of considered action are carried out without explicit, linguistic vehicles; my point here is that the interrelation between language and thought involves a number of factors that ought to be taken into consideration when studying human behaviour, especially in regards to the cognitive machinery we theorists postulate.

There are, I believe, two relevant ways in which the relationship between language and thought can be framed. One is in terms of the acquisition of language. Many of the relations that linguistic structures express must be representable a priori if they are to be acquired, as very often these are not deducible from either the structures themselves or the context. Segal (2001) refers to this as a distinction between having a structure and putting it into use. Consider the two sentences he employs to make this point (p. 127):

(1) Mary's Ferrari is faster than any Lotus.

(2) If Mary's Ferrari is faster than any Lotus, Pete will eat his hat.

Note that the phrase *Mary's Ferrari is faster than any Lotus* appears in both sentences, but the word *any* can only mean EVERY in (1), while it can either mean EVERY or AT LEAST ONE in (2). This is a difference in grammaticality and judgement that cannot be derived from the surface structure; nevertheless, the language faculty derives the same structure in both cases. Rather, the right interpretation is a matter of mapping each sentence to the right conceptual structure.

Another relevant example comes from the study of pidgin and creole languages. In a classic study, Bickerton (1981) describes the difference between specific and non-specific readings of noun phrases, exemplified by the following pair: *you should see the doctor* and *call the doctor who treated Marge* (p. 207). As he then points out, pidgins usually lack such a distinction, but creoles —the language that the offspring of pidgin speakers eventually create— do exhibit it. Consequently, given that they are exposed to a language that lacks it, children must possess a prior system in which this distinction can somehow be represented —that is, they could not derive the difference in meaning between those two sentences from the input they receive.

These two cases point to the existence of a language of thought in which these relations and structures are representable, something that must be granted if we are to account for how a natural language can be acquired at all. A fortiori, this language of thought must be able to represent whatever relations *any* natural language expresses. Alternatively, it could be suggested that acquiring a specific language offers the possibility of entertaining thoughts than one would not otherwise be able to. However, it is not at all clear how an organism could create/learn/acquire a system/structure more powerful than the one it already has, as stressed in Fodor (1975, 1979).

Another way to relate language and thought is in terms of the two levels of explanation I outlined above, and how they relate to each other. The first level refers to the structural properties of a given mental capacity, what Chomsky (1965) calls *competence*, akin to Marr's computational level. The second level focuses on the actual operations in play during actual behaviour; that is, how the capacity is put to use —Chomsky's *performance*, akin (very roughly) to Marr's algorithmic level.

Regarding the first level, Chomsky (1980, p. 58) speculates that the linguistic capacity may be an assembly of different systems whose union results in a domain-specific mental faculty. M. D. Hauser, Chomsky & Fitch (2002) expand this point and offer a breakdown of the language faculty into the following elements: a computational system (CS), a finite set of elements (i.e., lexical items; very roughly, words) that are combined by the CS into more complex structures, and two external systems that "interpret" these generated representations, the conceptual-intentional (C/I; roughly, the "thought systems") and the sensori-motor (SM; less roughly, the apparatus that converts the linguistic structure into a physical signal).

Furthermore, these scholars suggest that perhaps most of the properties of the language faculty are the result of the conditions imposed by the external systems. Thus, linguistic expressions exhibit hierarchy because of an imposition by the thought systems, while the fact that the physical signal is linear and flat results from the constraints of the SM interface.

Note that if hierarchy is to be explained as a condition of the C/I interface, a rather rich structure is *ipso facto* ascribed to this component. It is, then, plausible to suppose that the language of thought has a *sui generis* organization, one that involves a proprietary vocabulary (i.e., concepts) that is manipulated by a CS whose operations are compositional and systematic. This CS might well be shared by both the language capacity and the thought systems, but the way it applies will differ across the two domains, given that concepts and lexical items *have* different (structural) properties.

Regarding the algorithmic level, it is to be expected that *parsing* operations will have particular properties, different from those of the grammar, even if the overall language comprehension system must make use of (or interacts with) the knowledge base of the language faculty if we are to explain why speakers interpret sentences one way and not another. Naturally, it is the grammar that establishes what principles make a sentence grammatical (including when they apply) but the way the parser operates need not be isomorphic to the corresponding grammatical derivations. After all, the parser proceeds in an incremental, left-to-right manner, which contrasts with the bottom-up derivations linguists postulate (Bever, 1970).[1]

Similarly for the thought systems, since the language of thought must somehow be connected to perceptual systems (including the linguistic parser); that is, there has to be an online process in which perceptual information is translated into the language of thought so that it can then be combined with other sources in the formation of beliefs (this is independent of those cases in which a thought is generated from general knowledge or long-term memory, which might well occur without the participation of the perceptual systems). This transformation could well be the result of principles that may have no direct connection to the structural properties of the language of thought itself.

There is also the question of the relation between the computational and algorithmic levels, not a simple matter. I will come back to this later on, but for now I will anticipate that, for language at least, I will settle on an "analysis-by-synthesis" nexus (Halle & Stevens, 1959). What is of interest at this moment is the resultant picture I am describing: one in which we study distinct mental *realities*. A competence-like study focuses on the elements underlying mental *faculties* (in the sense of Chomsky 1980), while an analysis of performance studies the online mechanisms of processing *modules* (in the sense of Fodor 1983). I will expand this point in chapter 1, but for now it will suffice to state that it is at both these levels (and their interconnection) that studies on the language/thought relationship ought to be focused. As an illustration, take the particular case of the role of language in spatial cognition, a topic that has generated much debate.

In a series of papers on spatial re-orientation tasks —tasks in which subjects are, firstly, disoriented inside a room, and then asked to find their away around by employing geometrical and non-geometrical (featural) information— Hermer-Vázquez and colleagues (Hermer-Vázquez, Spelke & Katsnelson, 1999; Hermer-Vázquez, Moffet & Munkholm, 2001) have argued that natural language plays a

[1] In the early stages of generative grammar, sequenced applications of rewriting rules were employed to generate structures, and the very first step involved opening a sentence (S) node. This can metaphorically be described as a top-down process, but current linguistic theory (i.e., the *minimalist program*) postulates a building operation (viz., *merge*) that is customarily described as applying from the bottom up. That is, the first step of a derivation involves the merging of the verb with its arguments, and subsequent steps construct the remaining structure until the top node is reached. I will come back to this eventually.

crucial role in combining these two sources of information, as exemplified in sentences like 'the ball is to the left of the short white wall'. Apparently, children under the age of 5 cannot conjoin these sources of information in certain experimental settings, perhaps because they are yet to acquire the relevant linguistic representations. Similarly, adults also fail to adjoin this information if they carry out a secondary linguistic task such as speech shadowing, but not when the concurrent task involves rhythm-clapping shadowing. This suggests, to Hermer-Vázquez et al. (1999) at least, that language mediates the union of these two non-verbal representations, even if they *can* separately be entertained in a language of thought. In this sense, language augments your representational power, as it conjoins representations that cannot be so merged in the language of thought (for some unspecified reason). In a perhaps stronger flavour, Carruthers (2002) suggests that language in fact links up the different (conceptual) modules of the mind; it is the inter-modular language of the mind.

There are many reasons to doubt this. Firstly, note that the three works just referenced allocate modular status to these bodies of information, perhaps an unwarranted assumption. After all, it is not clear that either of these two sources of information forms a self-contained body of structural properties mediated by domain-specific mechanisms, let alone that they need any special mechanism to connect them. Fodor (1983) described modules in terms of a rather restricted set of properties (such as speed, automaticity, encapsulation, ecc.), but this at least had the advantage of individuating modules as a phenomenon so narrow as to make explanation possible —indeed, it is because they are modular that they can be scientifically studied at all. Unsurprisingly, Fodor allocated modular status to peripheral systems only; that is, psychological mechanisms that operate regardless of general information (and many other factors).

Moreover, it turns out that languageless creatures manage to accomplish the merging of geometrical and non-geometrical information just fine (see references in Twyman & Newcombe 2010), and nonverbal infants should be expected to do so too. In fact, 18-month-olds achieve this when the size of the experimental setting is big enough (ibid., p. 1324). The different sources of information *can*, then, be combined in conceptual structure, which suggests that the results of Hermer-Vázquez et al. may have little to do with general properties of mental architecture.

Nevertheless, the experiments suggest possible *processing effects* due to language and these ought to be explained in the terms I have described before; that is, in terms of what elements language and thought share, and how they interact in real-time processing. In fact, the regressive study carried out by Hermer-Vázquez et al. (2001) points in this direction. Therein, they attempted to find correlations between possible factors and found that the only one that reached statistical significance was the linguistic production of phrases involving terms like *left* and *right* (the sample was rather small though, N=24; furthermore, this presupposes having the concepts LEFT and RIGHT, not a trivial matter). One ought to emphasise that it was linguistic *production* that was found significant; that is, a performance phenomenon, not a transparent fact about mental organisation.

The relevant question to ask, then, is: how is it that production (as in the speech shadowing condition) blocks merging of geometrical and non-geometrical information? It cannot be that verbal shadowing constitutes a secondary task that impedes the primary task of merging these two sources because they both share a specific component (productive language), as the evidence alluded to supra clearly shows that this integration can be done in thought. If it were not so, one could then wonder about the actual character of the sentences that subjects represent to themselves. The example I mentioned earlier on —viz., 'the ball is to the left of the short white wall'— is of course only one way to codify this information, but there are equivalent ways to do this: 'the short white wall is to the right of the ball'; 'the wall is short and white', 'the ball is right next to it'; 'the ball is to the west of the non-black, little wall', *e così via*. Are we to ascribe/project any of these representations to the participants? Would this have any effect on the interpretation of the data? If we are being serious about linguistic production being the necessary link between these "modules", this would surely be relevant.

Actually, the only plausible interpretation is that adult subjects may well be representing the task to themselves in inner speech, and speech shadowing may consequently be interfering with this. That is, these tasks may be probing the effect of inner speech and speech shadowing on attentional mechanisms, including the conscious reflection on how to solve a task.

Samuels (2002) is almost right when he points out that speech shadowing is a language production task that, according to standard accounts, involves, inter alia, the integration of communicative intentions. He then points out that there are in fact two sort of integrations to carry out in the speech shadowing version of the experiment: on the one hand, the integration of geometrical and non-geometrical information; and on the other, integrating the communicative intentions behind the linguistic message. It is plausible to claim that both integrations take place in the "central system" (in the language of thought), which would surely cause a significant memory load; or, at least one greater than in the rhythm-clapping condition; hence, the different results.

To be more precise, speech shadowing is a *type* of linguistic production, as it does not incur the same integration load as normal production. It certainly *does* involve some load in this sense, but it also engages whatever cost comprehending the sentences to be shadowed imposes —i.e., speech shadowing is not an automatic parroting of the material you hear. Marslen-Wilson (1985) provides evidence that both fast and slow shadowers process the syntax and semantics of the input material before they integrate it into the sentences they output. That is, response integration and its execution are important factors in speech shadowing. It may not be a full-blown case of comprehension *or* production, but properties of both processes are operative. We should then expect that this causes a working-memory overload stemming from the two cognitive processes at play: perceptual integration and decision-making on the one hand, and language comprehension and production on the other.[2]

It will have been noted that Samuels frames his discussion in the very terms I have advocated. Indeed, he makes an effort to decompose the task into atomic operations (possession of the right concepts, perceptual integration, communicative intentions; and what I added regarding comprehension) to then place them in the right mental loci (syntactic parsing would involve the language faculty and its parser only, while semantics and intentional content would clearly engage the thought systems; plausibly, perceptual integration is carried out in the central systems). This is a necessary step in our analysis if we are to understand what is going on in the experiments. A corollary of all this is that the data seem to be telling us much more about processing modules and their cognitive load than anything about the organization of mental faculties, but this is only evident once we distinguish between faculties and modules and focus on the components they might share (and how they might relate).

*****

It is the purpose of the present work to offer an analysis along these lines for the study of recursion in cognition. In recent years, there has been an explosion of studies on this ever-more-important notion, but it seems to me that most of this work is confused, confusing and sometimes both. Terminological discrepancies may well account for some of the most confusing studies, whilst the failure to provide an orderly computational theory in terms of faculties and modules hints at what is wrong with the most confused works.

In order to be more precise, the thesis will aim to examine and elucidate the following issues. First and foremost, two questions immediately arise: what exactly is recursion? and, where does this concept come from and how should it be applied in cognitive studies? The right answers for these two general queries provides an appropriate frame for the overall discussion, and significant space is consequently devoted to them at the beginning of the thesis. The core of this work, however, is devoted to the much narrower aim of establishing the proper characterisation of recursion within linguistic studies in order to then work out the role it plays in linguistic knowledge and use. Considering that linguistic generation and processing constitute different levels of analysis, this sort of study raises the question of whether a uniform understanding of recursion can be provided so that it applies to different constructs (such as operations, structures, and perhaps others). It also points to the role recursion may have in other aspects of cognition, and these matters are treated at length in the last part of the thesis.

[2]Surprisingly, Carruthers (2002) dismisses Samuels's point while ignoring the evidence that speech shadowing involves both (a type of) comprehension and (a type of) production; indeed, a type of linguistic performance that *does* involve the central systems. He specifically denies all this, referencing an out-of-date study of speech shadowing (p.712).

That, at least, is what will be argued here. In what follows, I have attempted to provide as detailed a study as I have been capable of achieving, but as with any product of creative endeavour, this is nothing but a time-sliced snippet of work that is still very much in progress.

This long essay, then, is organised as follows. The first chapter begins with introductory remarks regarding the origin and meaning of the word "recursion", including common and technical connotations. Building on that, it then describes the employment of recursion in the formalisation of algorithms and computations within the formal sciences, with clear repercussions for cognitive science. Indeed, it will be claimed that your idea of what exactly constitutes a computation —i.e., the actual abstract model, the mechanism and operations— establishes, at the very least, the level of analysis you are adopting in the study of a given cognitive domain. The second chapter chronicles the introduction of recursion into linguistic theory, offers some definitions and relates them to the capacity to produce a possibly infinite number of structures. It also argues against conflating recursive structures and recursive mechanisms (endemic in the literature), and identifies the precise connotation recursion ought to have in linguistics. The third chapter looks at the computational system underlying the faculty of language in some detail, with a focus on the abstract, but none-the-less real, computations the faculty effects, including the interface conditions these computations must meet. The fourth chapter constitutes an experimental undertaking designed to probe and illustrate the potentially recursive nature of a sub-operation of the syntactic parser by looking at the possible correspondence between recursive structures and mechanisms. Chapter 5 discusses some universality claims and describes how this bodes for the study of non-linguistic cognition. The conclusion aims to bring everything together for summary and contemplation, while the postface anticipates further work to be undertaken, with some more comments on general cognition. Finally, the appendices provide the materials employed in the experiments.

Chapter 1

Computations and theories of cognitive domains

"A running back"; or recursion as it was once recorded in *An English Expositor*, John Bullokar's XVII. century compendium of "the hardest words", and references were a-plenty at the time, at least in technical tracts. To mention but two examples, Robert Boyle talks of 'the recursions of that Pendulum which was swinging within the Receiver' in his 1660 work *New Experiments Physico-Mechanical* (chapter XXVI, page 203), while Richard Gilpin muses about how 'our Passions... depend upon the fluctuations, excursions, and recursions of the Blood and animal Spirits' in *Dæmonologia sacra, or a treatise of Satan's temptations* (1677, vol. II, chapter VII, page 307).

The word itself appears to have entered the English language as an adaptation of the Latin "recursus", a polyseme that can stand for either a noun or the past participle of the verb "recurrere" (to run back, or return, according to *Andrew's Edition of Freund's Latin dictionary*). As a noun,"recursus" is a synonym of "recursio", and just like its English descendant, it meant "a running back".

In the 1933 edition of the *Oxford English Dictionary*, this denotation is recorded as rare and obsolete, but it was precisely at this time that the term recursion was starting to gain widespread usage in the mathematical literature, albeit with a rather more technical meaning. Still, the modern and the ancient senses share the property of being about "recurrence", doing justice to its Latin root.

The first occurrences of the more technical connotation are already quite frequent in the German literature of the XIX. century. Alfred Clebsch, for example, devotes some space to "recursionsformel" in his 1872 book *Thorie Der Binren Algebraischen Formen*, while James Pierpont's 1905 *The Theory of Functions of Real Variables* contains one of the earliest examples of "recursion formula" in the English language. More to the point of this thesis, Kurt Gödel writes "rekursiv" in German in 1931 (cited in Sieg 2006) to refer to the class of functions that were to become so central to mathematical logic.

This central role revolves around the original mathematical interpretation of recursion as a "definition by induction", a technique pioneered in the XIX. century by Richard Dedekind and Giuseppe Peano (Soare, 1996). Also known as a recursive definition, it consists in 'defining a function by specifying each of its values in terms of previously defined values' (Cutland, 1980, p. 32), as the following example will illustrate.

Let us define the class of the factorials, that is: $n! = n \times n-1 \times n-2 \ldots \times 2 \times 1$, where n is a natural number. A recursive definition consists of a pair of equations, the first of which is called the "base case", while the second constitutes the "recursive step":

(1.1)

$$\text{Def. n!} \begin{cases} \text{if } n = 1 & n! = 1 \qquad\qquad\qquad \text{(base case)} \\ \text{if } n > 1 & n! = n \times (n-1)! \quad \text{(recursive step)} \end{cases}$$

Note that the recursive step involves another invocation of the factorial function. Thus, in order to calculate the factorial of, say, 4 (i.e., $4 \times 3!$), the function must return the result of the factorial of 3, and so on until it reaches the factorial of 1, the base case, effectively terminating the recursion. Self-reference is therefore the defining feature —that is, its denotation— making it a *special type* of recurrence.

It is this self-reference property that binds all correct uses of the term —or so I will argue below. Nevertheless, a distinction needs to be drawn between what makes a function recursive (viz., the aforementioned self-reference) and the uses to which this property can be employed. Hence, the different recursive functions (primitive, general, partial) and the different input-output relations they encompass.

A related but distinct concept is a so-called "mathematical induction", sometimes referred to as an "inductive proof" or even an "inductive definition". This is a mathematical technique employed to prove whether a given property applies to an infinite set, and proceeds as follows: first, it is shown that a given statement is true for 1; then, it is assumed that it is true for n, a fixed number (the inductive hypothesis); and finally, it is established that it is therefore true for $n+1$ (the inductive step). These three statements constitute the *direct clauses*, divisible into the *basic clauses*, which tell us that the value for such and such object is such and such, and the *inductive clauses*, which tell us that if the value for such and such object is such and such, it will then also be the value of a related object. The final *extremal* clause establishes that only the objects in the direct clauses have such and such value. If every step is followed correctly, it can be concluded that the statement is true for all numbers (R. Epstein & Carnielli, 2008).

Peano's definition of the natural numbers (the so-called Peano Axioms) constitutes the classic example of a mathematical induction and it may be useful at this point to offer a schematic representation. Following Kleene (1952, pp. 20 et seq.), this scheme can be represented in three steps: *a*) 0 is a natural number (this is the basic clause); *b*) if n is a natural number, then $n+1$ (or n', as is sometimes represented) is also a natural number (this constitutes the inductive clause); finally, *c*) the extremal clause establishes that all natural numbers are defined by steps (a) and (b).

It is important to note that inductive definitions are a central feature of recursive definitions in the sense that the former grounds the latter; that is, mathematical induction justifies the recursive definition of a function over the domain established by the inductive definition (Kleene, 1952, p. 260). However, I will not focus on mathematical induction here; rather, I shall draw my attention to systems of recursive equations and its derivatives. More specifically, I will be studying constructs such as recursive generation, processing and data structures as they pertain to the study of cognition, with special emphasis on the language capacity.

The next section offers a detailed analysis of what is involved in the formalisation of an algorithm and its implementation, including the data structures that these constructs manipulate. It constitutes the necessary background in order to understand the different roles recursion plays in the cognitive sciences, and by extension in linguistics. Indeed, recursion is a concept that originated in the formal sciences, and its employment in the study of cognition makes sense, I will argue below, as long as its treatment does not diverge from the original interpretation(s) too much. In doing so, a number of computational formalisms will be described, and a pretty direct connection will be drawn between these formalisations and specific cognitive models. A connection that carries, I will argue towards the end of this chapter, a significant epistemological and ontological baggage.

1.1 Formalising an Algorithm

By the time David Hilbert presented a collection of problems for mathematics in the early XX. century, one of which was to formalise the notion of what he called "effective calculability", a computation was understood in intuitive terms as a 'process whereby we proceed from initially given objects, called *inputs*, according to a fixed set of rules, called a *program*, *procedure*, or *algorithm*, through a series of *steps* and arrive at the end of these steps with a final result, called an *output*' (Soare, 1996, p. 286). The challenge was to formally characterise the finite, mechanical procedure at the centre of this intuition, and a number of different solutions appeared from the 1930s onwards.[1]

Recursive definitions, as described in the previous section, were to be widely employed in the 1930s and beyond for this purpose. One of the first proposals, however, was instead based on the lambda (λ) calculus Alonzo Church had invented in the 1920s to investigate the foundations of logic (first published in Church 1932, though).[2]

[1]Hilbert presented some of these problems at a lecture in Paris in the year 1900, and the whole list appears in an English translation in Hilbert (1902).

[2]It was Church himself who identified computability with this formalism in a letter to Gödel in 1934, but it was not well-received (Sieg, 2006, p. 191). According to Feferman (2009, p. 205), Gödel not only rejected Church's proposal, he also rejected a suggestion by Jacques Herbrand.

The λ-calculus is composed of so-called *lambda terms*, which are defined and derived in terms of an equational calculus that makes use of two basic operations: application and abstraction. There are three types of lambda terms. A variable, e.g. x, is a term; $M \cdot N$ is the application of the function M to the argument N; and finally, $\lambda\ x \cdot M$ is called the abstraction, and should be understood as stating that x is assigned the value M.

It was soon discovered that this system was inconsistent (i.e., it gives rise to paradoxes), but if the part that dealt with logic (such as logical constants, ecc.) is eliminated, this results in an "untyped", pure λ-calculus that is relevant to the formalisation of a computation. Nevertheless, in a 1936 paper, Church turns his attention to the *general recursive functions* Gödel had introduced in 1934 (Kleene, 1952, p. 274), an identification between computability and general recursiveness he felt Gödel himself had suggested in a footnote of his 1934 paper.[3] Gödel, however, made clear this was not the case at all (Sieg, 2006, p. 192); rather, his point was that you could expand the class of *primitive recursive functions* into a more general class of recursive functions.[4]

In schematic form, a recursive definition of a function f from a function g can be provided in terms of $f(0) = m$ and $f(n+1) = g(f(n))$ (R. Epstein & Carnielli, 2008, p. 91). The two basic components (or operations) of this scheme are induction and composition; that is, $f(n+1)$ is inductively defined from $f(n)$ and the scheme allows for the combination of f and g into a $g(f(n))$ compound. Starting from a set of initial, basic functions —among others, *zero* (Z) and *successor* (succ), as defined below (where the symbol $'$, it will be recalled, means $+1$)— the primitive recursive functions are 'exactly those which are either basic or can be obtained from the basic ones by a finite number of applications of the basic operations' (ibid., p. 93).[5]

(1.2) $Z(n) = 0$ (for all n)

(1.3) Def. succ

$$\begin{array}{ll} a+1 = a' & \text{(base case)} \\ a+n' = (a+n)' & \text{(recursive step)} \end{array}$$

The expansion into the general recursive functions was the result of realising that the primitive functions were not the only schema that could be defined by the

[3]It is worth pointing out that even though Church explicitly states the correspondence between λ definability and the general recursive formalism, the attention in his 1936 paper clearly falls on the latter; and hence the fact that Church's Thesis was originally described in terms of general recursiveness; see infra.

[4]Much like most of the literature, I will be using the term "class" to refer to all these constructs, but these should be understood as inductive definitions of the different labels (viz., primitive recursive, general recursive, ecc.).

[5]I defer to R. Epstein & Carnielli (2008, pp. 93 et seq.) for many other examples of primitive recursive functions. The important point at this moment is that not all computable functions can be derived with this method; that is, not all computable functions are primitive recursive —a well-known fact, and of which I will say no more.

system of equations I described above, as a 'stable and important class of functions' could be derived from taking the primitive recursive equations as a starting point and applying substitution and replacement operations as inference rules (Sieg, 2006, p. 192).

The actual formalisation of the general class of recursive function is rather cumbersome for the subject matter of this thesis, but the following description will suffice. Firstly, note that the previous functions have all ranged over one variable only, while the general recursive functions are allowed to apply with respect to two variables at the same time. Following Gödel (1934, pp. 69–70), then, if ϕ refers to an unknown function and $\psi_1,\dots,\psi_k$ represent known functions, and if the ψ's and the ϕ can be substituted into one another in a certain way, ϕ is a general recursive function if the resulting set of equations has only one solution for ϕ. The full formalisation of this class can be found in the aforementioned paper.

Church (1936), then, identified computability with the general recursive functions, and this definition has come to be known as Church's Thesis: 'every example of a function... acknowledged to be... calculable... has turned out to be general recursive' (Kleene, 1952, p. 300).[6] This statement, however, is incorrect. The fatal flaw seems to be that the core of Church's formalism consisted in a number of atomic steps that were stepwise recursive, making it semi-circular (Sieg, 2006, p. 193), but he provided no justification for this (see Soare 1996, p. 289–91 and Sieg 1997 for details).

It was in fact Church's student, Stephen Cole Kleene, who replaced the general recursive with the *partial recursive functions* in 1938. The latter are so called because they map a *subset* of the natural numbers onto the natural numbers, making them "incomplete" functions. More specifically, this formalism makes use of a "search" procedure —the least search (μ) operator— that looks for the least number with a given property. According to R. Epstein & Carnielli (2008, p. 124), these functions are the smallest class containing the basic, initial functions of the primitive recursive class (see supra) *and* it constitutes a closed system under composition, induction and the μ-operator (the partial recursive functions are sometimes referred to as the μ-recursive class, in fact). The partial recursive functions correctly identify the class of computable functions (Kleene, 1952, pp. 317 et seq.).

Coming from a different set of assumptions on how to formalise computability, Alan Turing famously put forward a proposal that was based upon the idea of an abstract, mechanical device —the Turing Machine (TM)— that constitutes the paradigmatic example of the so-called *models of computation* (see infra). It is commonly conceded that this model captures the manner in which every conceivable mechanical device computes a calculable function, and was generally accepted as the best formalisation in the field (Soare, 1996, p. 291–295). It is of great importance to point out that a TM employs no recursion, as its operations are strictly

[6]Sieg (1997) argues that Church proposed his definition in terms of λ-definability in 1934 already, but it first appears as a "thesis", as far as I have been able to determine, in Kleene (1943); it was in fact titled Thesis I therein.

speaking “iterations”.

For any two sets, X and W, an iterative mapping function ($X \rightsquigarrow W$) is a quintuple (input, S, σ, T, output), where S is the set of states, *input* is the function $X \to S$, σ is the transition function from S to S, T is the set of terminal states, and *output* is the function $T \to W$ (Moschovakis, 1998, p. 80). From a structural point of view, though, a TM is composed of a head that can read and write symbols on a tape by following the set of instructions codified in the so-called configurations —the quintuple defined above.

Emil Post concurrently worked on a similar program, independently of Turing, but his 1936 work on “finitary combinatory processes” was not as wide-ranging as Turing’s (see Soare 1996, p. 300 for details).[7] It was later in the 1940s when he developed a method that correctly formalises computability, even if it seems that he achieved these results much earlier.

Post (1921) formalises the truth-table method for establishing validity within propositional logic in terms of a number of postulates from which the set of propositions can be generated. In this work, Post talks of functions *producing* or *generating* other functions, a way of phrasing things that is at the heart of the systems he outlined later on. Indeed, introduced a “canonical form” system in (1943), and he explicitly states that this construct directly stems from the “generalisation by postulation” method of his 1921 work.[8]

A canonical form, then, is composed of a number of primitive assertions and a specified finite set of productions. In its simplest, more general form, what Post calls a “normal form”, this can be described in terms of the following mapping: *gP produces Pg'*, where g stands for a finite sequence of letters (the *enunciations* of logic) and P represents the operational variables manipulating these enunciations (Post, 1943, p. 199). The canonical and normal forms constitute an explicit method of production, to be distinguished from the objects they generate (the canonical and normal sets).

An important point to emphasise —given that I will come back to this repeatedly below— is that the whole approach ‘naturally lends itself to the generating of sets by the method of definition by induction’ (ibid., p. 201). That is, there is a sense in which recursion plays a central, global role within canonical forms, a fact that may well apply to algorithms in toto; indeed, as Sieg (1997) points out, it is most natural to consider the generative procedures underlying these systems as ‘finitary inductive definitions’ (p. 166).

In further developments (Post, 1944, 1947), these “production systems” are converted into string rewriting rules, which are therein called Semi-Thue systems.[9] A string rewriting system is a special type of a canonical form. It starts with a single initial word, and the productions are each of the form $P_1\ g\ P_2 \longrightarrow P_1\ h\ P_2$;

[7] Post’s system consisted of a two-way infinite tape and one symbol only.

[8] The final footnote in Post (1943) states that the results he is therein reported had been worked out by himself in the summer of 1921.

[9] Canonical forms and the like are usually referred to as “production systems” because Post writes the word *produces* between the related functions; he later employed the $\longrightarrow$ symbol.

or more generally: $g \longrightarrow h$ —a substitution rule.

In the last papers I have referenced, constructs such as *recursively enumerable* and *(general) recursive* sets feature extensively, and these will be of some relevance later on. A set is recursively enumerable if there is a mechanical procedure that can list/enumerate all its members, while a set is (general) recursive if there is an algorithm that can determine whether a given element is (or is not) one of its members.

By the 1950s, all these formalisms were shown to be extensionally equivalent; i.e., from the same input, they could all generate the same output. Or in a more general vein, all these systems identify/refer to the class of computable functions:[10]

$$\text{(1.4)}\quad \text{Computable Functions} \begin{cases} \text{Lambda calculus} \\ \text{General recursive functions} \\ \text{Turing machine} \\ \text{Production systems} \\ \ldots \end{cases}$$

R. Epstein & Carnielli (2008) call this the The Most Amazing Fact, given that '[a]ll the attempts at formalizing the intuitive notion of computable function yield exactly the same class of functions' (p. 85). The actual formalisms identified here as co-extensive were to be polished in the 1960s and 70s, though. I have already mentioned that general recursion was replaced by partial recursion, and I will describe a further refinement below. As for the dots in the graphic above, these refer to later formalisations, such as a "register machine", a TM-like construct that has been shown to be Turing-equivalent. Register machines are much better-suited for describing a modern-day computing device, but they will not feature in the rest of this study.[11]

Nevertheless, the equivalence among these systems suggests we are dealing with a fundamental class (Kleene, 1952, p. 320), which moved Post (1936) to consider the most amazing fact a "natural law" rather than a thesis. This state of affairs has come to be known as the Church-Turing Thesis, or sometimes just *Church's Thesis* (CT).

For the purposes of semantic hygiene, I will take a narrow interpretation of the CT to stand for the identification between general recursive functions (or partial recursive functions) and the computable functions, while a broad understanding would just be a tag name for the so-called Most Amazing Fact. Similarly, I will

[10]As a matter of fact, Turing (1936) showed his TM to be equivalent to λ-definability —he was unaware of Church's work on general recursiveness at the time of writing his paper— while Church and Kleene (op. cit.) showed that λ-definability was equivalent to general recursiveness and in turn to partial recursiveness. Rosser (1939) appears to have been the first to have shown the equivalence between general recursion, λ-definability and a TM. Cf. the discussion in Sieg (1997), though.

[11]Suffice here to say that, from a structural point of view, a register machine replaces the TM's head and tape components with registers, which are simpler in formulation. These registers are small units of memory that are stored in the central operation system, the component that controls how the rules are applied.

take the *Turing Thesis* (TT) to mean that all intuitively computable functions are TM-computable. On the other hand, I will take the *Church-Turing Thesis* to mean that all recursive relations can be reduced to iterative relations (and vice versa; see Rice 1965 for a short but concise demonstration of this).[12]

There should not be much controversy over these results and their implications once the actual claims are appropriately characterised. Soare (1996), however, sees grounds to cast doubt on subsequent developments, in particular the historical narrative accompanying them. Therein, he describes a state of affairs in which recursion is taken to be at the very centre of what a computation is, to the point that systems of recursive equations, or recursion itself, are almost synonymous with computability and/or computable, in detriment of Turing's model. He suggests this is the result of the literature tacitly following what he calls the Recursion Convention (RC): *a*) use the terms of the general recursive formalism to describe results of the subject, even if the proofs are based on the formalism of Turing computability; *b*) use the term CT to denote various theses, including TT; and *c*) name the subject using the language of recursion (e.g., *Recursion Function Theory*).

Soare has certainly amassed much evidence supporting this description of the field, and his point is well-taken (see also Soare 2007a,b). I will come back to this later on, but I could perhaps venture a couple of reasons for this state of affairs. One is anticipated in Church's review of Turing's work (reprinted in Sieg 1997), where Church points out that even though Turing's formalism has the advantage of making the identification between a computation and a TM intuitively obvious, the other solutions (that is, general recursiveness and the λ-calculus) are more suitable 'for embodiment in a system of symbolic logic' (p. 170) —naturally, the latter are likely to be more prominent in mathematical logic overall. Another reason might be sociological in nature, given the enormous influence that Kleene's seminal work —*Introduction to Metamathematics*— has had on the field, a book that vigorously defended the validity and truth of the CT (both in the narrow and broad sense outlined here).

Be that as it may, the aforementioned studies from the 1930s and 40s did not, contrary to popular belief, settle what a computation is. In the case of the models proposed by Church and Kleene, their work can be seen as an attempt to clarify the nature of the functions that can be computed —that is, a hypothesis about the set of computable functions— but there is more to an algorithm than the function it can compute (Blass & Gurevich, 2003). Turing's idea of an abstract machine, on the other hand, does not model an algorithm either; what this formalism describes is what can actually happen during a computation that is set in motion.[13]

It is in McCarthy (1963) that we find an explicit formulation of computations in terms of a simple set of recursive equations, a formalism that constitutes a precise refinement of the partial recursive functions and is considered by some as superior

[12]Kleene (1952) defines the TT in slightly different terms, but I am following Soare (1996) here.

[13]To be more precise, Turing focused on how a human "computor" performs mechanical procedures on symbolic configurations (Sieg, 1997).

in simplicity and range to both Turing's and Kleene's (Moschovakis, 2001, p. 919). This system achieves TM-equivalence while employing a few operators and a precise (λ-calculus) notation for functions.

The formalism starts from conditional expressions like the one below,

(1.5) $(p_1 \rightarrow e_1, p_2 \rightarrow e_2, \dots. p_n \rightarrow e_n)$

where p_1,..., p_n are propositional expressions taking the values True or False, and the value of the entire expression is the value of the e corresponding to the first p that has value True (T). It is possible to define functions recursively with this scheme, as shown for the factorial functions:

(1.6) $n! = (n = 0 \rightarrow 1, n \neq 0 \rightarrow n \times (n-1)!)$

The system can then be generalised by employing λ-notation so that functions and forms (expressions involving free variables) are distinguished. The latter are assigned "truth values", playing an important role in this formalism, and new functions can in turn be defined from old ones by a "composition" operation. The result is a general scheme that is capable of modelling any sort of computation. The following is how the factorial functions look like within this approach (the reader is deferred to McCarthy 1963 for many other examples):

(1.7) $n! = (n = 0 \rightarrow 1, T \rightarrow n \times (n-1)!)$

Studies like McCarthy's, then, attempt to formalise an algorithm as a formal object, that is to say, to establish what it is; more specifically, this formal object is being characterised as an equation. A number of concomitant principles follow. The fact that the equivalent formalisms subsume the class of computable functions implies that they are all underlain by an abstract object. Further, that such a mathematical object in fact exists; that it is distinct from the objects it operates on (henceforth: the data structures); that it is abstract in the same sense that the philosophers' proposition is abstract (i.e., many non-identical instances of an algorithm may denote the same truth value); and that as a result of all this an algorithm can therefore be implemented in various ways.[14]

There is a distinction to be had, then, between formalising an algorithm qua mathematical formal object and formalising an algorithm qua model of computation (the latter is sometimes called the implementation of an algorithm: a computation in course). It is to the latter construct that a TM appropriately applies and it is certainly the case that systems of recursive equations (or production systems, for that matter) may well be more appropriate to characterising an algorithm, for they do not depend on the abstract idea of a machine. Indeed, there appears to be a distinction between recursive specifications of algorithms and the abstract idea of a machine computing in a step-by-step manner. In this vein, it is interesting to note

[14]Dean (2007) englobes these identity conditions into what he calls *algorithmic realism*.

that Turing (1954) talks about computations in terms of Post's production systems; that is, 'purely in terms of combinatorial operations on finite discrete configurations' (Sieg, 2006, p. 190); the idea of a machine may well have obscured the fact that Turing was dealing with general symbolic processes (ibid., p. 203).[15]

The structural description of algorithms is precisely the type of work currently being conducted in the *analysis of algorithms* discipline, where recursion remains a central property. Thus, Moschovakis (1998) subsumes the recursive equations of McCarthy, plus an output mapping, under the notion of a "recursor", which is claimed to faithfully model the structure of an algorithm —in this sense, recursors *are* algorithms.

A recursor, a mapping function from a partially ordered set X (the input) to a set W (output) ($X \rightsquigarrow W$) is a triple (D, τ, *value*), where D is the domain of the recursor (a recursive partially ordered set), τ is the mapping function ($X \times D \rightarrow D$), and *value* is the value mapping of the recursor ($X \times D \rightarrow W$) (Moschovakis, 1998, p. 85).[16]

One of the reasons an algorithm can have a number of implementations relies on the fact that it may be expressed in different programming languages, and therefore may be *implemented* differently on different machines. The notion of an algorithm itself, however, is entirely independent of the differences of these representational schemes; all an algorithm points to is to a process in which some structures are converted into some other structures in a sequential manner. If the preceding is so, this would suggest that abstract machines are much closer to implementations —indeed, an implementation is sometimes described as an instance of a model of computation (Dean, 2007, p. 127).

The study of algorithms splits into two different approaches, then. On the one hand, algorithms are studied as abstract objects, and one needs to answer questions like 'what is an algorithm?' On the other hand, algorithms are analysed in terms of implementations, and this level also splits into two. Implementations can be studied either in the abstract, what is usually known as formulating a *theory of the computation* —an in-between field comprising (parts of) mathematics and (parts of) (theoretical) computer science— or as real-time processes, which is the focus of applied computer science.

The abstract study of implementations corresponds to Marr's computational level in the study of cognition; that is, the mapping function between input and output and the formal properties that derive therefrom. Some of these derived properties are architectural in nature, such as overall complexity and organisation, but others would appear to be more concrete, such as time and space efficiency, which might indicate they pertain to engineering studies. In actual fact, the last two properties are treated in functional and therefore abstract terms at this level; that is, in relation to a particular computational architecture (Knuth 1997 is a clear

[15] Also, Turing may have been in agreement with Post's claim that productions systems were 'far simpler than, say, the λ-calculus of Church' (Post, 1943, p. 200).

[16] Further, recursors and recursive equations are related 'in the same way that differential operations are related to differential equations' (Moschovakis, 2001, ft. 11).

example of this type of study).

There is nothing mysterious about this; the mapping function can be analysed independently of the operation that performs it in real-time, as Dean (2007, pp. 256 et seq.) demonstrates. An abstract implementation can be divided into atomic components, and cost units can be assigned to each, resulting in a sequence of stages. Though a sequence, this just means that one stage is followed by another, making the immediate predecessor relation the central feature of the analysis. Thus:

> 'we standardly factor out the actual durations which would be required to transact the various operations [of the algorithm]. . . [a]nd through the use of a size metric and asymptotic notation, we also abstract away from the contribution of additive and multiplicative factors which appear' (Dean, 2007, pp. 257–8).

Regarding the study of real-time implementations, Marr's algorithmic level, the various formalisms we have so far outlined effect the transformation from input to output in different ways, and this will certainly result in more tangible computational differences. I am pointing to a distinction between extensional equivalence and intensional differences that has clear repercussions for the study of real-time processes, given that differences in time and/or space have a clear effect on memory load and efficiency *at this level*.

This brings us to issues that have to do more with computer science than mathematical logic, and it would now be useful to clarify some of the terminology I will be employing. For this purpose, I will be relying on two sources: Abelson, Sussman & Sussman (1996), SICP henceforth, and Knuth (1997, vol. 1), AOCP hereafter.

Programs, procedures and algorithms were included in the intuitive definition of a computation at the beginning of this chapter —the "fixed sets of rules"— but there are some important differences to note (AOCP, p. 4). I defined an algorithm as a process for the successive construction of quantities; to this it is now added that the mapping must meet a number of conditions, such as Mal'cev's well-known five criteria: discreteness, determinacy, elementarity of the steps, direction and massivity (see R. Epstein & Carnielli 2008 for further details; cf. the five criteria in AOCP, pp. 4 et seq.).[17] A computer *program*, though defined as patterns of rules that govern computational processes manipulating abstract elements (the data) (SICP, p. 1), does not need to meet all the criteria. In fact, a program is to be understood as a 'machine-compatible representation of an algorithm' (Brookshear, 2000, p. 2); or as Knuth puts it: 'the expression of a computational method. . . is called a program' (AOCP, p. 5).

Thus, we treat an algorithm as an abstract entity distinct from its representation (the program). Further, some form of "representation scheme" is required for an algorithm to be represented, and this is the function of a *computer language*. The

[17]Note that these criteria collapse the distinction between a formal object and its abstract implementation, but it will not matter here.

latter I will define as a collection of primitives and rules, both finite. Finally, a *procedure* is a step-by-step list of instructions for completing a task, a definition that is informally applied to algorithms, but mistakenly so. Indeed, much of the literature employs these two terms —algorithms and procedures— interchangeably, but I follow SICP and AOCP in distinguishing them.

In what follows, I will use examples from the LISP computer language for exposition. We only restrict our notation by this choice, not what we may program. The use of LISP will be so trivial and simple that a detailed description is not needed. However, it is important to clarify a methodological question stemming from this way of describing things —one that has to do with a distinction between computer science and mathematics.

Programming is not merely about solving certain tasks; it also allows us to carry out an exercise in "procedural epistemology". That is, we can study different bodies of knowledge from an imperative point of view, so that our computer language functions as a formal medium for expressing relations and ideas about methodology. It is this imperative point of view that differentiates (to a certain extent, of course) computer science from mathematics. While the latter deals with "what is" questions, the former focuses on the "how to"; that is, computer science does not only attempt to describe the relationship between inputs and outputs, it also aims to describe the most efficient computation from one to the other.

Using a computer language to study relations between ideas/concepts, then, should not be frowned upon. After all, whatever we program/describe with a computer language is a product that is ultimately meant for people to read, and only incidentally adapted for machines to execute. Our goal is to provide an effective and efficient description of knowledge, and computer languages may prove to be an invaluable tool, at least for what immediately follows (see SICP, pp. xvii et seq. for more details).[18]

LISP makes ample use of procedures. As mentioned, these describe the rules for manipulating abstract objects, with the characteristic, peculiar to LISP, of being able to be represented and manipulated as data themselves (SICP, p. 4). In a certain sense, procedures are much like mathematical functions, with the only difference that the former must be effective (ibid., p. 21). As Roberts (2006, p. 47) states, this is because procedures are algorithmic in nature, while functions are not. Thus, procedures must meet Mal'cev's criteria, while functions do not.

As with other programming languages, LISP makes use of three basic elements: *a*) the primitive expressions, which constitute the simplest entities of the programming language; *b*) the means of combination, out of which compound elements are formed from simpler ones; and *c*) the means of abstraction, so that compound elements can be manipulated as units (SICP, p. 4).[19] As with other computer

[18]Marvin Minsky's aptly named 1967 article "Why programming is a good medium for expressing poorly-understood and sloppily-formulated ideas" makes a similar point.

[19]I will mostly deal with numerical data in this section, but the same rules may be employed to manipulate other types of data. This important point was already clear to Thomas Hobbes, though: '[w]e must not ... think that computation, that is racionation, has place only in numbers' (quoted in

languages, we interact with LISP via an *interpreter*. We type an *expression*, either simply a primitive element or two or more elements linked by an operator, and the interpreter *evaluates* the expression. The interpreter, then, carries out the processes described in the LISP language.[20]

If we type in the simple expression below:

(1.8) 1980

the interpreter will merely return it as the output.

If we type in a complex expression, the evaluation will return the result of applying the procedure specified by the operator. I follow the convention of placing the operator to the left of the operands (the so-called Polish notation), and placing the whole expression in parentheses in order to resolve ambiguities. The result of evaluating the compound expression below would simply be 1980.

(1.9) (+ 1979 1)

The interpreter then works following a basic cycle: it reads an expression, evaluates it and prints the result. Parentheses can in turn be used to write nested expressions, introducing a hierarchy (of both structures and operations) of a potentially unlimited nature. One may be confused by the compound expression in the example below, but an interpreter would not have a problem with its complexity.

(1.10) (+ (* 1980 (+ (* 1980 1) (+ 1 1980))) (+ (- 1 1980)))

As for the third basic element of programming —the means of abstraction— LISP can achieve this in various ways, but the description of one operator will suffice for the purposes at hand: *define*. The idea is to rename and recode a complex computational object under a new tag/label. Take the process of "squaring" numbers as an example. This operation can be described with a simple statement: 'to square something, multiply it by itself'; or in LISP notation:

(1.11) (square x (* x x))

The procedure definition for this compound expression is simply:

(1.12) (define (square x (* x x)))

Thus once defined, we can use the square operation as a single unit. We can simply type in the expression below and the interpreter returns the result of carrying out the procedure:

AOCP, p. 650).

[20] I am ignoring the different evaluation strategies available to the programmer, such as *call by name*, *call by value*, et alia. The λ-calculus is usually employed for these, but I do not think this omission makes my analysis lacking in any sense.

(1.13) square 25

(1.14) 625

Obviously, the interpreter needs to be able to retrieve the relations we define. That is, it needs some sort of memory, known in LISP parlance as the "environment".

It will have been noticed that procedures can apply recursively. That is, in order to evaluate the elements of a compound, the interpreter must evaluate all the elements of the sub-expressions, which involves applying the procedure of the operator (i.e., the leftmost element of the sub-expression) to the operands (i.e., the rightmost element(s) of the sub-expression). In order to evaluate a complex expression, then, the interpreter must evaluate each element of the sub-expression first. Thus, the evaluation rule contains as one of its steps the invocation of the rule itself. Much like the recursive functions, the operation calls itself.

This is an example of recursively-defined procedures; what interests me here, however, is the nature of the processes that procedures generate. That is, the computational processes I defined supra as real-time implementations. This is a difference between the actual computation in motion and the meta-rules that direct the process (see footnote 22 below for a further clarification of this point).

A direct result of the Church-Turing Thesis has it that an algorithm can be implemented recursively or iteratively. These two types of processes subsume the class of recurrent operations: both involve the repetition of an operation, and as a result both need termination conditions.[21] Recursive implementations involve self-reference (a given operation calls itself) and as a result chains of unfinished tasks develop, which automatically yields hierarchy among the operations so produced. In the case of iteration, an operation is repeated in succession, and in general its state can be summarized at any stage by the variables plus the fixed rule that establishes how these are updated from one state to another. If no termination conditions are established, both processes will proceed indefinitely. Furthermore, whilst both types of processes keep something in memory, recursive processes keep deferred *operations* rather than just *variables*, and this usually exerts a bigger load.

In order to illustrate, let us show how to calculate the factorials recursively and iteratively. The recursive implementation naturally follows from the recursive definition supra, repeated here for convenience.

(1.15)

$$\text{Def. n!}\begin{cases} \text{if } n=1 \quad n! = 1 & \text{(base case)} \\ \text{if } n>1 \quad n! = n \times (n-1)! & \text{(recursive step)} \end{cases}$$

I also include the corresponding LISP procedure, the meta-rules that will generate the recursive process (note the Polish notation for this and all the other procedures I include below).

[21] In fact, the very notion of an algorithm involves repetition, either iteration or recursion.

(1.16)
```
(define (factorial n))
    (if (= n 1))
    1
    (* n (factorial (− n 1)))
```

The iterative process, on the other hand, requires a subtler observation. Factorials can be iteratively computed if we first multiply 1 by 2, then the result by 3, then by 4, until we reach *n*. That is, we keep a running product, together with a counter that counts from 1 up to *n*. Further, we add the stipulation that *n*! is the value of the product when the counter exceeds *n*. That is, the computation carries on according to two rules: the product is multiplied by the counter, and then the counter is increased by 1 until it reaches a number higher than the number whose factorial we want to calculate. I include the procedure for calculating the factorials iteratively below, followed by a table that shows both types of implementations (SICP, pp. 33–4; n.b.: the first digit of the iterative solution shows the factorial whose number we are calculating, the second digit is the actual counter and the third is the running product).[22]

(1.17)
```
(define (factiter)
    (if (> counter max-count)
    product
    (factiter (* counter product)
    (+ counter 1)
    max-count)))
```

Table 1.1: Recursive and iterative implementations.

$4 \times$ (factorial 3)	factiter 4 1 1
$4 \times (3 \times$ (factorial 2))	factiter 4 2 1
$4 \times (3 \times (2 \times$ (factorial 1)))	factiter 4 3 2
$4 \times (3 \times (2 \times 1))$	factiter 4 4 6
$4 \times (3 \times 2)$	factiter 4 5 24
4×6	

As the shape of the recursive process reveals (shown on the left-hand side), there is an expansion followed by a contraction, the result of the number of deferred operations the process builds up. Indeed, the material kept in memory in these two processes differs greatly at any stage. In the second line of the recursive process, for example, the actual operation in course is *factorial 2*, while what is being kept

[22] Note that the procedure for the iterative implementation is also recursive, but the process it generates really is iterative. There is no contradiction here; it is the actual rules laid out in a procedure rather than how these are defined that establish the nature of the generated process. As a result, there is a certain subtlety involved in computing with a recursive definition —and in distinguishing between procedures and processes.

in memory is *4 × (3 ×...)*. Crucially, the process can only end if the operations are carried out in the right order. That is, the factorial of 2 needs to be computed before the factorial of 3, and a hierarchy amongst the operations consequently develops. Neither of these two properties quite apply in an iterative process, as the only things in working memory are the operation in course and the variables it operates upon, and there is certainly no relation between running operations and rules kept in memory. Consequently, an iterative process is in general more efficient.

Note that both implementations are recurrent in the sense that both operate over their own outputs. They differ in the type of operations they carry out, and consequently in the type of recurrence manifested. In the case of the recursive implementation, the operation carried out is a multiplication involving the factorial to be calculated, n, and the factorial of $n-1$. That is, the factorial operation applies over a *subset* of its own output, which involves a self-call.[23] In the case of the iterative implementation, however, the operation multiplies the new counter times the previous product, which is to say that it applies over its output plus a new variable, but no self-reference is involved. Naturally, reflexive calls exert a bigger memory strain, as the chains of unfinished tasks must be kept in memory until lower-level operations are carried out. Still, there exist certain data structures that merit a recursive solution.

I will describe these structures presently; for now I need to briefly expand a couple of points I have made here. I have used the factorials here precisely because they are a standard and trivial example of the formal equivalence (but computational difference) between iteration and recursion; a usefully transparent instance of translating Church-Turing Thesis into actual computational processes. There are, of course, numerous other examples in the literature. Liu & Stoller (1999), for instance, offer a framework that provides automatic transformations of recursion into iteration, an "optimization technique" that can cope with the most complex of recursive relations, such as multiple base cases or multiple recursive steps. A recursive definition of the Fibonacci sequences is an example of the former, that is to say, of a recursive definition that is composed of multiple base cases:

(1.18)

$$Fib(n)\begin{cases} \text{if } n=0 & 0 & \text{(base case)} \\ \text{if } n=1 & 1 & \text{(base case)} \\ \text{otherwise} & Fib(n-1)+Fib(n-2) & \text{(recursive step)} \end{cases}$$

Table 1.1 above provides a rather schematic view of the shape of the recursive implementation. Note, however, that since a recursor is underlain by a two-equation system, a recursive implementation creates a hierarchy of a special type, one that can be properly described with a binary tree, as shown by Fig. 1.1 for the factorials.

This hierarchy is among the operations, and not the data structures.There is no sense in stating, by looking at the tree, that the factorial of 3 is embedded into the

[23]That is, the operation *factorial* applies to another instantiation of the operation *factorial*. To work out the factorial of something you need to work out the factorial of something else first.

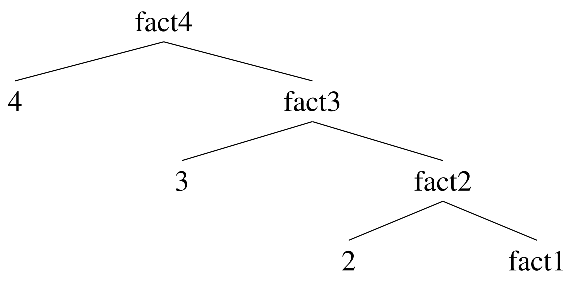

Figure 1.1: Recursive implementation of the factorials

factorial of 4. This would amount to a definition of a structure in terms of how it is generated, but why do that? After all, most people are taught at school that the factorial of 4 is calculated by multiplying 4 by 3, then by 2, and finally by 1, and this magically eliminates the once-perceived embedding. The hierarchical organisation of structures and operations should not be conflated; after all, they need not be isomorphic.[24]

Schematic trees are an ideal way to represent non-linear data structures and operations. As Knuth states, 'any hierarchical classification scheme leads to a tree structure' (AOCP, p. 312). Let us define a tree structure as a finite set of one or more nodes such that there is one node called the root, and the remaining nodes are further partitioned into subtrees of the root (ibid., p. 308). Hierarchy is accounted for by the structural, branching relationships between the nodes. Recursors are especially well-suited to operate over this sort of scheme, since 'recursion is an innate characteristic of tree structures' (ibid., p. 308), where "innate" probably means "intrinsic" here. In fact, "tree recursion" is a term widely used for such a computational pattern (SICP, p. 37).

Moreover, a tree scheme is an appropriate representational format for showing how costly the calculation of a recursive definition may be. This is clearest in the case of the Fibonacci sequences, as outlined in Fig. 1.2.

As this figure shows, the recursive implementation of Fibonacci numbers involves much redundant calculation, as the values for (fib 1), (fib 1) and (fib 0) are computed twice. This would be worse for higher numbers, given that the number of steps required by such a process is proportional to the number of nodes in the tree (SICP, p. 39). Consequently, an iterative implementation would exert less cost here too.[25]

[24]The case of the factorials is an ideal example to show, contra Zwart (2011a, p. 43), that it is possible to describe an object as recursive without access to the procedure that generated it. In fact, recursive structures are very often described and analysed in their own terms, as I will show soon enough.

[25]I defer to SICP (pp. 39 et seq.) for an iterative way to calculate the factorials. Much like

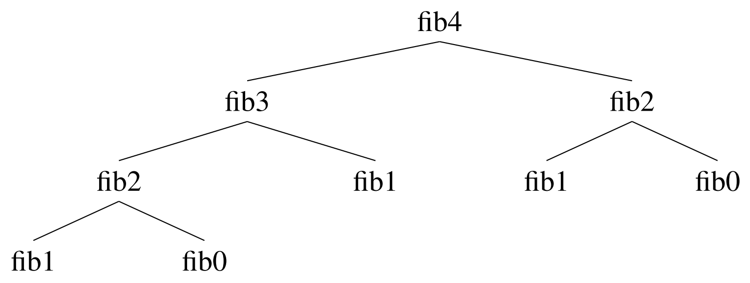

Figure 1.2: Recursive implementation of Fibonacci sequences

The issues are not as clear-cut as this discussion would suggest, though. Computer scientists have developed "optimisation" techniques to minimise the time and space required to execute a program, and this can be applied to make time-consuming recursive processes more efficient. Some of these techniques include "partial evaluation" or "loop unrolling", but the relevant one for tree structures is the so-called "deforestation", an attempt to flatten out tree schemes. One simple way of doing this, with the factorials at least, is to introduce new base cases. An implementation for the following recursive definition would return a value for *fact 4* after only two operations, but the process remains recursive.

(1.19)

$$\text{Def. n!}\begin{cases} \text{if } n = 1 & n! = 1 & \text{(base case)} \\ \text{if } n = 3 & n! = 6 & \text{(base case)} \\ \text{if } n > 1 & n! = n \times (n-1)! & \text{(recursive step)} \end{cases}$$

Be that as it may, the internal hierarchy of recursive implementations makes them specially well-suited to operate over complex objects such as a "recursive data structure", defined by the U.S. National Institute of Standards and Technology as an object or class 'that is partially composed of smaller or simpler instances of the same data structure'. That is, a structure that includes an abstraction of itself (an X within an X), and "trees", "lists" and the like constitute the prototypical cases (trees inside other trees, or lists inside lists, ecc.).[26] There is in fact a natural fit between recursive data structures and recursive mechanisms, a fact stressed in no small measure by Wirth (1986, p. 135). Despite this close correspondence, orbiting conditions —such as memory limits, architectural complexity, efficiency— more often

with the factorials, note that the underlying procedure of an iterative implementation may remain recursive, and this is in fact the case for the example that appears in SICP. The subtle distinction between procedures and processes seems to have confused Fitch (2010) in his discussion of Fibonacci sequences vis-à-vis recursion.

[26]URL: http://www.itl.nist.gov/

than not bring about iterative implementations. In other words: the reduction of recursors to iterators, where reduction means that such or such recursively-specified algorithm is implemented by an iterator.[27]

The orbiting conditions traditionally have to do with memory limitations of the physical machine that is implementing the algorithm; i.e., properties of the implementation and not of the algorithm itself. Therefore, it can be the case that even though a set of data structures naturally merits a recursive implementation, iteration is chosen instead; after all, implementations require time and space.

There is, however, nothing intrinsically recursive about the factorial of 4; it is in fact the nature of the solution to calculate the factorial that makes it apt for a recursive implementation. The recursive method employed to compute factorials was based on the rather subtle observation that we could solve the problem by reducing it to one or more subproblems identical in structure and simpler to solve (Roberts, 2006, p. 4).

In general, three properties must be met for such a solution: *a*) the original problem must be decomposable into simpler instances of the same problem; *b*) the sub-problems must be so simple that they can be solved without further division; and *c*) it must be possible to combine the results of solving these sub-problems into a solution to the original problem (Roberts, 2006, p. 8). As I will argue in subsequent chapters, it is this set of properties that would suggest recursive processes in human cognition.

The discussion in this section has introduced a distinction between "structural recursion" and "generative recursion" that constitutes part of the necessary background to understand the roles and applications of recursion in cognition, and I will turn to the latter in the next sections. More importantly, a three-stage explanatory process is being delineated: *a*) a characterisation of the general notion and structure of an algorithm; *b*) the study of its "abstract implementation", a level that studies the formal properties of the mapping function, including the goodness of fit between structures and operations; and *c*) a description of its "actual implementation" as executed in real time. It is to this program of research as it applies to cognitive science that I will focus in the next three chapters, respectively.

*****

Before I move to all those issues, it is necessary to clarify what the different theoretical terms I have employed thus far specifically stand for, given that some of these terms are employed interchangeably in the literature. Here, however, all of them will receive a unique and precise interpretation. First of all, I do not identify an algorithm with a list of steps and rules that returns the desired result if the instructions

[27] The reduction of recursively-specified algorithms to iterative models of computation is considered by Moschovakis (1998, 2001), Moschovakis & Paschalis (2008) and Dean (2007) as the basis for a "founding" of mathematical logic. I will come back to this issue at the very end of the thesis.

are appropriately followed; instead, this would be the definition of a procedure. An algorithm will here be understood as a formal mathematical object that transforms quantities, the object at the heart of any computational system; a mapping function. Relatedly, an implementation of an algorithm, what is usually called a model of a computation, results when the procedure for an algorithm is set in motion. In this case, we say that the list of steps of a procedure are being *implemented* as a computational process; a process, then, is simply the implementation of a procedure. Whether the implementation is abstract or applies in real-time, a process is composed of two things: operations —which are carried out by mechanisms (that is, adding 2 and 5 is an operation that is effected by the *adding* mechanism)— and variables, what computer scientists call the data structures. Crucially for our purposes here, the "shape" of a process is the result of the manner in which the operations manipulate the data structures. And finally, it is important to note that I have defined the term "recursion" as "self-reference", a denotation that has furthermore been associated to four different constructs (or in other words, four different connotations): a definition by induction; as a feature of what an algorithm is (viz., a "recursor"); as a property of computational processes, as in operations that call themselves, resulting in chains of deferred operations; and as an attribute of data structures, where an X is contained within an X. All these definitions and distinctions will feature extensively in what follows.[28]

[28]Cf. the different constructs Gersting (1982, p. 131) lists: a recursive sequence (wherein the first one or two values in a sequence are known, and subsequent items are defined in terms of earlier ones); a recursive set (wherein a few specific items are known to be in a set and the other items are built from combinations of items already in the set); a recursive operation (wherein a "small" case of an operation gives a specific value and the other instances of the operation are defined in terms of the smaller cases); and finally, a recursive algorithm (wherein the behaviour of an algorithm is known for the smallest values of an argument, while for larger values of an argument, the algorithm invokes itself with smaller argument values).

1.2 The spur of Cognitive Science

The acts of the mind, wherein it exerts its power over simple ideas, are chiefly these three: 1. Combining several simple ideas into one compound one, and thus all complex ideas are made. 2. The second is bringing two ideas, whether simple or complex, together, and setting them by one another so as to take a view of them at once, without uniting them into one, by which it gets all its ideas of relations. 3. The third is separating them from all other ideas that accompany them in their real existence: this is called abstraction, and thus all its general ideas are made.

John Locke

An Essay Concerning Human Understanding.
Book II, Chapter XII (1690).

The quotation heading this section describes a fact of cognition, one that requires explanation. That is, what are the underlying properties of mental domains and mechanisms that effect such phenomena? Famously, John Locke, alongside many others, proposed an associative engine: a mechanism that connects mental representations wherein the experience of one leads to the effects of another, a connection that becomes stronger as these pairings are repeated.[29]

This idea exerted a huge influence on many fields, including human psychology, especially within the so-called behaviourist theories of learning and thought. It was part of the behaviourist mantra that cognition reduced to stimulus-response pairings, dismissing the internal and therefore mental machinery effecting these pairings as a non-crucial element of the theory. It was not so much a matter of denying the existence of mental ideas and operations (at least it was not so for the less radical strains of behaviourism), so much as it seems to have been the denial that mental phenomena could serve as the units of explanation, given that whatever mental operations mediated the transformation of stimuli into responses could be swept under the carpet by a swift application of the commutative law. That is, if stimulus S_1 elicits mental representation MR_1, and this in turn elicits MR_2 and so on until producing response R_1, the mental unobservables need not preoccupy the psychologist —one could still claim that all the mind does is carry out associations, the final product of this process being a stimulus-response pair that should be considered the focus of inquiry.

[29]Note that the Locke quote describes really well the three main elements of programming I outlined in the previous section. Indeed, this very quote appears in many introductory textbooks on computer science. It perhaps verges on comedy, however, to point out that what computer scientists have precisely *not* proposed is an associate engine for their programming languages; such programs are, rather, systems of explicit symbols and rules without associative intent.

It is not here the place to catalogue the failings of behaviourism, but suffice to say that its practitioners underestimated a point of which Gestalt psychologists were well aware: it is not the physical (external) properties of the (distal) stimulus that explain behaviour, but how the stimuli are represented in the mind (that is, the distal representation). Hence, the Gestalt "schemata". Granted, the Gestalt psychologists were nowhere near providing a description of the structural properties of the distal representation, but the move from studying stimulus-response pairs to how the stimulus is represented and manipulated brings our attention to the very operations that modify the mental representations —not a trivial matter.

Indeed, the so-called "cognitive revolution" can be seen as a concerted effort to show that cognitive phenomena could not be accounted for simply in terms of associations. G. A. Miller (1956) is a case in point, as it provides evidence that the working memory capacity does not operate over individual items as if they were uncombinable bits; rather, individual items can be combined into compound units by a sort of "chunking" process, and it is to the latter that working memory is sensitive —a nice example of Locke's "abstraction" principle. The important point here is that chunking is the result of structure-preserving operations, that is, computations over representations, the foundational principle of cognitive science. This is sometimes referred to as the so-called "computational theory of mind" (CTM): the thesis that cognitive processes are computational in nature.

Even though it is sometimes stated that Thomas Hobbes and Gottfried Leibniz anticipated this point (see the relevant papers in Brook 2006 and J. C. Smith 1991), it is clearly the case that the Lockean phenomena outlined above could only be understood once mathematical logic had formalised the notion of a computation. That is, mathematical logic provided the foundation upon which cognitive science was built. Indeed, Fodor (2008), in a perhaps liberal reading of the literature, remarks that cognitive science got a head start thanks to Turing's suggestion that mental processes were not associationist, but computational. Strictly speaking, this is a rather free interpretation of either Turing (1936) or (1950). In a more accurate manner, the former was preoccupied with providing a formal description of how a computation could be at all possible, while the latter focused on a rather narrow point: whether a machine could exhibit human-like behaviour in an imitation task.

Nevertheless, it is certainly the case that this particular formalism, the TM, provides a foundation for the sort of computational models cognitive science defends, a formal system in which, as Haugeland (1981) describes it, three elements must be specified: a set of tokens, a definition of the starting position of these tokens, and the rules allowing the tokens to be changed into other tokens. This, effectively, describes the so-called "classical" model of cognition. This system, then, manipulates representations according to structure-preserving operations —a syntactic engine— to which it must be added an interpretative system so that these representations are allocated meaning —a semantic engine.

Rather surprisingly, however, the cognitive science literature appears to contain very few examples of computational models that are based on any of the other formalisms briefly described in section 1.1. Indeed, most textbook introductions to the field (for instance, Edelman 2008 and Gallistel & King 2009) focus on the TM formalism in their outline of what a computation is, with rather superficial remarks about the Church-Turing Thesis and its variants.

One could perhaps argue that the other formalisms are just not relevant, or too cumbersome to implement. Instead, I want to argue that the adoption of the TM is not surprising at all, given the division of interests and foci I outlined in section 1.1 regarding the study of algorithms. It seems to me that for most of cognitive science, and particularly so in the case of cognitive psychology, the focus falls on *processes* rather on the underlying properties and capacities of mental architecture. That is, most of cognitive science focuses on Marr's algorithmic level of explanation. If that is so, the TM is without a doubt the paradigmatic example of a "model of computation", and it accords very well with the sequential step-by-step computations that so worries the more "applied" strands of computer science.

Moreover, it is clearly the case that the vast majority of models in cognitive science are all variants of the TM, a fact that seems to have confused Pinker (2005a,b) somewhat. Therein, and as part of a discussion with Jerry Fodor, Pinker casts doubt on the centrality of the TM in cognitive science. After all, he tells us, a TM is a very costly machine to build, and the Artificial Intelligence literature does contain diverse computational models that prima facie bear little resemblance to a TM. This is technically incorrect, however, as a TM is an abstract construct that captures the way in which every conceivable mechanical device could compute. Indeed, Turing himself was rather clear in stating that his motivation was to decompose the mechanisms of a computer 'into "simple operations" which are so elementary that it is not easy to imagine them further divided' (Turing, 1936, p. 136). Hence, the overarching result of his so-called Universal Turing Machine.

Be that as it may, it will be the aim of the remainder of this chapter to show that selecting one of the other available formalisms is not a trivial matter. Even though all these formalisms can effect the same input-output pairs (they are all expressively equivalent), the intensional differences among them have a clear effect on the mapping function or operation so postulated (where the distinction between functions and operations is also of great importance; see below). This is perhaps most obvious in linguistics, which sets this field apart, to a certain extent, from the rest of cognitive science.

Going back to Locke's three principles, these underlie what in modern terms have come to be known as the systematicity, productivity and compositionality of cognition. Systematicity refers to the ability to process and represent structurally related objects. Thus, if you can entertain the thought that MARY LOVES JOHN, you can ipso facto entertain the thought that JOHN LOVES MARY. Productivity points to the fact that one can entertain many such thoughts (and many other variants) and these two properties are to be explained, according to the classical model (Fodor & Pylyshyn, 1988; Fodor & McLaughlin, 1990), in terms of the inherent structure of the corresponding thoughts. That is, complex representations are composed, literally, of constituents, so that one can entertain similarly structured thoughts in virtue of the structure and combination of their constituents;

compositionality.

Note, however, that this model is proposed in order to explain the behavioural phenomena —that is, systematicity, productivity and compositionality are *facts* of cognition, but it could well be the case that a non-classical model provides a better explanation. Contra Kitcher (2007, pp. 242–4) and her "epistemic dependencies", then, these phenomena are not derivable after-effects of the classical model of cognition; rather, any theory needs to accommodate them. An alternative theory that attempts to account for these very facts is "connectionism", and given that the present study will advance a classical model of cognition, it is worth devoting some space to a brief comparison of the two systems.

It is sometimes stated that connectionism emerged out of the desire to provide a cognitive architecture that closely resembles the structure of the brain, and since the brain is composed of interconnected neurons —its basic functional atoms— connectionism models the mind in terms of a neural network of artificial units; let us call these networks C-nets (Churchland, 1990).[30]

These artificial units are linked to each other via weighted connections, which specify the strength and the nature of the interactions: inputs may be either positive (activating other units) or negative (inhibiting them). The job of a unit, then, is to receive input, compute output value and send it to its neighbours (Rumelhart, 1989, p. 211). The units are organized in layers —the input and the output being the obvious two— with the addition of a "hidden units" layer in between. Hidden units constitute a central component of the overall system, as they mediate communication between input and output.

Given that this model does not make use of symbols in its computations, the information that a C-net represents, in contrast to a classical model, is not to be found in any explicit form. The information is to be found, intermittently, appearing and disappearing in various pools of units and, lastly, in the pattern of weights. The idea is that the resultant pattern of weights will be a reflection of the statistically-structured environment, which is to say that the context the C-net is immersed in drives the learning process. In order to be more specific, the learning process is usually carried out in the following manner: a given model is fed a "training set" (i.e., a large set of samples from some domain) and it proceeds in a trial-and-error manner until it induces the output of the correct application of a corresponding rule. As long as the network remains in error, the "guilty" weights are adjusted so that in time the C-net converges to the right rule.

Naturally, it needs to be demonstrated that a C-net could exhibit the systematic and compositional behaviour so characteristic of human cognition by just adjusting to the environment. Plausibly, a classical architecture that carries out syntactic operations over symbols will ipso facto be able to process similarly-structured objects in similarly-structured ways.[31] In this sense, it is not so much that a classical

[30]Note, however, that despite the evocation of the brain, these models are as abstract as any classical architecture.

[31]This is doubted in Matthews (1997), though.

model of cognition conforms to a statistically-structured environment; rather, the environment is interpreted given the constraints imposed by intrinsic properties of the mind —i.e., the environment is structured in the manner that the mind perceives it to be.

Let us, then, take systematicity and compositionality as a yardstick with which to measure the success of C-nets in learning and/or processing structured representations.[32] In order to do so, I will make use of the discussion in Hadley (1994a,b, 2004), as the description and outline of the problem therein is explicit, concise, and more importantly, testable (I will also draw from S. Phillips 1994a,b).

Hadley (1994a) distinguishes three different versions of systematicity, later expanded into four in (1994b), all of which are relevant here:

1. Weak systematicity: an agent meets this condition if it can process "test" sentences containing novel combinations of words (that is, a word that occurs within a specific sentence in a specific position can also occur in every permissible position).

2. Quasi systematicity: this involves weak systematicity and it is extended to include embedded sentences (that is, sentences within sentences; there will be much more about embedded sentences in subsequent chapters).

3. Strong systematicity (SS): quasi systematicity plus the ability to process known words that appear in novel positions but that are not present in the training set (e.g., this is the ability to transform an active sentence composed of known words into a passive sentence —a new pattern).

4. Strong semantic systematicity (SSS): strong systematicity plus the ability to assign appropriate meanings to all words in any novel test sentence (e.g., this involves understanding verbs that have been encountered in the indicative but that are now used in the imperative).

Children unequivocally show the last two abilities while acquiring a language (see Guasti 2002 for details), and any cognitive model would have to account for this fact of cognition. In particular, note that SSS involves a rather complicated system, as it involves very sophisticated interpretative mechanisms that are not well understood by any approach. Nevertheless, C-nets ought to at least achieve the SS level.

The literature contains a number of models that claim to achieve a non-trivial level of systematicity, but Hadley (1994a,b) finds all of them lacking. The models presented in van Gelder & Niklasson (1994a,b,c), for instance, are all inadequate, as they do not constitute, as the authors themselves admit, psychologically plausible models. Their point seems to have been rather narrow: of course C-nets can

[32]I leave productivity out, but this property appears to be even harder to account within a C-net model, as the range of thoughts that can be entertained may well be infinite, a property that most C-nets theorists explicitly deny (see Gallistel & King 2009 for more details).

achieve SS, but only when armed with many mechanisms that are clearly not part of the psychology of children.[33] Chalmers (1990), on the other hand, provides a model that is able to map structures between distributed representations but it proves unable to produce passive transformations from patterns never encountered before (i.e., it only achieved weak systematicity). There are more proposals around, but I defer to Hadley's papers, especially Hadley (2004), for a demonstration that most C-net models fare very poorly.[34]

In their critique of some C-net models, Fodor & McLaughlin (1990) expanded on what it means for a system to be systematic: the computations need to be sensitive to structure, which is to say that the processes that manipulate complex representations must have access to the internal constituents. A classic understanding of compositionality automatically accounts for this, as the internal constituents of a compound are preserved and accessible. van Gelder (1990), however, puts forward an alternative conception of compositionality, one that he argues can be employed in order to model systematicity within C-net systems. Therein, then, he contrasts two different modes of constructions —two views of understanding compositionality, that is— in virtue of how complex expressions stand towards their internal constituents. The standard version he calls a "concatenative" compositionality, because tokens are "placed" alongside each other (ibid., p. 360). The non-concatenative version is a type of functional compositionality that obtains when some general processes produce a complex expression given some constituents (and the reverse: it can decompose a compound back into its constituents).

The non-concatenative version, on the other hand, effects a composition (or decomposition) mechanism without preserving the primitives. According to van Gelder (ibid., p. 371), many C-nets make use of just this type of compositionality; in the case of Smolensky's model (1991), this is combined with a tensor calculus so that concatenate symbolic representations can be translated into tensor product representations (i.e., vectors).

Note, however, that the system does not have access to the mode of construction at all, making it insensitive to structure. S. Phillips (1994a) argues that this is a fatal flaw, and that the functional/concatenative distinction is nothing more than a tangential issue. Consider the two constructive functions below:

(1.20) Concatenation: (JOHN, ELSON) $\longrightarrow$ JOHNELSON

(1.21) Multiplication: (3,4) $\longrightarrow$ 12

As he rightly points out, the corresponding deconstructive processes —call them deconcatenation and factorising— would not know what values to return from the complex expressions, as these are ambiguous. That is, both *12* and *JOHNELSON*

[33]Such as a tagging system that identifies every single word by the syntactic role it plays in a sentence.

[34]I am ignoring hybrid systems; that is, C-nets that make use of some symbolic representations. Some of these fare a bit better but they do not cut across the dividing line between the classical and the connectionist, which is what I am interested in here.

may well be decomposed into many primitives; confining ourselves to pairs, these are all possible: (1 & 11), (3 & 4), ecc. for the former, and (JOHN & ELSON), (JOH & NELSON), ecc. for the latter. The ambiguity arises because the representation itself does not restrict the possible decompositions. In short, the underlying system must have access to the mode of construction, but functional compositionality does not allow for this. This is effectively the argument that Fodor & McLaughlin (1990) levied against many C-nets (and this includes Smolensky's system); a rather crucial failing of connectionism, it would appear.[35]

The discussion thus far is meant to clarify two points. Firstly, I have tried to justify the adoption of a classical model of cognition, given that the structures and operations to be treated in this study require a discussion of explicit representations, mechanisms and capacities. Secondly, it would have been noticed that both the classical and the connectionist are computational stories of cognitive processes that seem to focus on real-time implementations (even if they vary, at least on the surface, on the sort of computations they propose).

The sort of approach to the study of language that Noam Chomsky initiated 60 or so years ago, however, significantly differs from this way of framing the study of cognition, and recursion has certainly played a much more central role in linguistics than in cognitive science at large. As he has stated in numerous occasions, generative grammar developed within 'a particular mathematical theory, namely, recursive function theory' (p. 101 in Piattelli-Palmarini 1980) and I will attempt to clarify what this actually entails below (including the right interpretation for terms like "generative"). For now it is important to point that the impressive results achieved by mathematical logic in the 1930s provided a mechanical explanation for a property of language that had been noticed a long time ago; namely, the possibility of producing or understanding an incredibly large, perhaps infinite, number of sentences.

As I will show in successive chapters, the construction of a "generative grammar" involves, among other things, two broad tasks. On the one hand, it is necessary to describe the mechanical procedure at the heart of the language capacity —that is, what sort of computational system underlies language. On the other hand, a great effort is involved in providing a "theory of the computation" for language. That is, linguists construct theories of how complex linguistic structures are generated from atomic elements in accordance to a number of formal properties and conditions, and in this sense, so-called linguistic "derivations" are akin to the abstract implementations I described earlier. The aim is to discover what principles make the language faculty the way it is, an endeavour that can proceed independently, or so it has been argued, of matters concerning its processing and acquisition (including its neural instantiation).

Given the abstraction involved, then, recursively-specified algorithms are much

[35]This very failing carries over many other domains of cognition, in fact. For instance, C-nets have great difficulty in accounting for much of the behavioural data unearthed in animal cognition studies (Gallistel & King, 2009).

more natural at this level of explanation —Marr's computational level, similar to Chomsky's *competence*. I will discuss the different levels of explanation in some detail in the next section, but as I anticipated in the Introduction, this division of labour is not simply a distinction to do with different levels of analysis of the same phenomenon. Rather, as I will now argue, these constitute descriptions of different cognitive domains; an epistemological and ontological claim.

1.3 Mental Realities

At first sight, it seems to be a principle underlying both Marr's and Chomsky's "division of explanatory labour" frameworks that computational theories of mind ought to proceed in an orderly fashion. Indeed, Chomsky's whole oeuvre can be seen as an attempt to follow the rather sensible view that it is a prerequisite of cognitive science to understand the nature of the organism, in this case the mind of human beings, before analysing the behaviour it produces (e.g., Chomsky 1975b, p. 16). In this sense, then, there is a progression to follow in the study of cognition: from an analysis of what Chomsky calls "attained states" of the mind (in this case, the language faculty) to an attempt to determine the underlying capacities capable of acquiring them (ibid., p. 160). Marr similarly emphasises the importance of the computational-level analysis, given that the nature of the algorithm effecting the real-time transformation may be discerned 'more readily by understanding the nature of the problem being solved than by examining the mechanism (and the hardware) in which it is embodied' (Marr, 1982, p. 27). If so, it could then be concluded that the various levels of analysis describe the same phenomenon from different perspectives —and in fact, myriad studies seem to make this very claim (for an example chosen almost at random, see Neeleman & van de Koot 2010).[36]

In Chomsky (1965) and later works, competence —the knowledge base an ideal speaker/hearer possesses— is said to be embedded within wider systems, such as memory, parsing strategies, attention, and others; that is, those systems encompassing linguistic *performance*. Thus, even though the linguist is to provide a theory of the internalised grammar, the I-language of the speaker/hearer, this construct is to be compatible, indeed it is supposed to explain, how a language can at all be acquired and/or processed.

Let us call the overall collection of systems that fully describe language as the "linguistic capacity", an assemblage that will contain not only what linguists call the language faculty, including the interfaces its interacts with (C/I and SM, at a minimum), but also, the parser, the memory capacity involved in the processing of language, and general features of mental architecture. The different levels of

[36]Similarly for de Saussure, his *langue-parole* dichotomy has the same linguistic reality in mind, albeit from distinctive perspectives. It is worth noting, however, that neither *langue* nor *parole* match any of the levels of either Marr or Chomsky. Instead, de Saussure seems to have taken *langue* as a system of signs and conventions, while *parole* constituted a speaker's speech acts or behaviour. See Pylyshyn (1973, p. 23) and Jackendoff (2002, p. 29) for discussion.

explanation would, then, engage and explain different systems of a supposedly unified capacity.

As mentioned earlier, a study of the language faculty involves providing an explanation for two different constructs, the underlying mechanical procedure (a necessary postulate, given that the number of possible sentences we produce and understand exceeds memory) and a *theory of the computation* from lexical items to the interfaces. I will come back to a specification of the underlying procedure in the next chapter, for now it is of interest to focus on basic properties of the sound-meaning pairs the language faculty generates.

Generative grammarians have taken the "function in intension" that generates sound-meaning pairs as their topic of research; that is, linguists focus on the actual properties of the internal, formal generative mechanism that 'computes a certain recursive function' (Pylyshyn 1973, p. 44; cf. G. A. Miller 1975). Naturally, the grammar specifies the function that is computed in language use, *not* the algorithm that is in fact employed to do so in real-life interaction. The whole point of a "theory of the computation" is that the function that effects these pairs can be provided, in fact it should be provided, prior to figuring out how the function is computed in real-time processing. Moreover, the function in intension and the algorithm that computes it need not bear a transparent relation to each other whatsoever (Matthews, 1992; Steedman, 2000).

There is in fact some consensus in philosophy of language that the function in intension that specifies sound-meaning pairs is indeed what constitutes the subject-matter of linguistics (Matthews 2006; B. C. Smith 2006; J. Collins 2007, 2008b),[37] a position that was already defended in Chomsky (1975b, p. 120), and even more explicitly in Chomsky (1980, p. 82). As Chomsky has pointed out elsewhere (for instance, in the first edition of his *Language and Mind* book), this sort of study has historical precedents in certain strands of "faculty psychology", such as in the work of the XVI. century scholar Juan Huarte de San Juan (pp. 8–9). In *Examen de Ingenios*, his only book, Huarte focuses on the intensional, generative *potentiæ* of the mind, such as the capacity for understanding, an *ingenio* capable of 'generating figures within itself' (p. 193).[38] For the purposes of this essay, then, I will call these intensional *ingenios* of the mind "faculties".

There is much less consensus, in philosophy of language at least, as to whether the function in intension that speakers/hearers internalise ought to be identified with what is usually called "knowledge of language", and if so, whether this term has any epistemological import. Naturally, the locution "knowledge" suggests a propositional format to philosophers, and Fodor (1983, 2001) has not been alone in characterising linguistic knowledge in terms of propositional attitudes, a position that has been resisted by Chomsky since, at least, his 1975b book (and more forcefully in Chomsky 2001b). Therein, Chomsky has attempted to divert atten-

[37]Furthermore, I subscribe to their analysis and rejection of some other positions, such as those of Michael Devitt and Stephen Stich, of which I will not say much here.

[38]My translation from the Spanish edition.

tion from the epistemic connotations of the word "knowledge" by, first, replacing it with the more neutral, or so he thinks, "cognize" (1975b, p. 164); further, he has protested that one should not conflate what the theory is proper with how it is informally described, the implication being that "knowledge of language" is the term employed in casual characterisations, while the technical notion I-language is what is in fact being studied.

The epistemological question splits the aforementioned consensus into those who think that the locution "knowledge" may profitably retain an epistemic connotation, and those who do not. Matthews and B. C. Smith appear to believe that the very feasibility of linguistic research depends on an epistemic reading of "knowledge of language", while J. Collins remonstrates that this is an acceptation that stems from English collocation.

The debate revolves around the status and nature of the intuitions native speakers have of their internalised language. Historically, intuitions of grammaticality and acceptability have formed the main empirical evidence for linguistics. Indeed, they offer information of great value about the linguistic capacity, but as J. Collins (2007, 2008b) emphasises, it is up to the linguist to decide what exactly the intuitions tell us about the sound-meaning mapping the overall assembly effects; after all, the judgements themselves are not transparent statements of the underlying function in intension.

Nevertheless, it is certainly the case that speakers/hearers know something about the meaning of words and sentences. Therefore, according to Matthews (2006) and B. C. Smith (2006, 2008), linguists need justification for crediting grammatical and/or acceptable judgements as part of their data. Further, continues Matthews (2006), the very reliability of this methodology is only 'justified... on an epistemic conception of... competence' (p. 215). The latter point is related to the assumption that linguistic knowledge is, to some approximation, first-personal (and sub-personal), immediate, authoritative, *and* fallible (B. C. Smith, 2006, 2008). A fortiori, the very fact that speakers *can* be right or wrong about their judgements indicates an epistemic conception.

The "justification" argument seems to be clearly overstated, however, for grammatical intuitions are only part of some of the data that linguists in fact employ. Thus, in his response to Quine's worries that one cannot choose between two different grammars when both of them can generate the same set of sentences (an "extensional" equivalence), Chomsky (1980) points out that *there are* in actual fact explanatory constraints that decide between competing theories. Learnability properties, acquisition facts, psycholinguistic data, and even brain-imaging evidence are all potential sources of information for the linguist. Again, it is up to the linguist to catalogue and classify this information in a format that is informative of the sound-meaning pairs he is interested in, but there seems to be no truth of the matter regarding the epistemic conception of *any* of these. That is, they do not require any justification beyond the fact that they are useful; moreover, why would an epistemic conception justify methodology at all?

In a sense, the different sources of information share the same origin; that is, they all are the output of the overall assembly of systems that compose the linguistic capacity. This, of course, applies to linguistic intuitions too, as they are, after all, *performance* data: they are a reflective judgement over what the *parser* returns. Who is to say, then, that these judgements constitute knowledge of the sound-meaning mapping function, rather than the overall capacity? B. C. Smith (2008) is clearly aware of this, but argues that when 'performance factors don't intervene', the knowledge the speaker has pertains to 'linguistic facts fixed by his internal language faculty' (p. 68). It is hard to imagine situations in which "performance factors" are not operative; even harder it is to imagine a situation in which competence and performance are transparently disentangled in linguistic behaviour.[39] Nevertheless, B. C. Smith (2008) states that intuitions are *'about which strings are acceptable'* (p. 67; his emphasis), but this is merely the answer to a question posed by a linguist, it certainly does not ipso facto follow that intuitions track the operations of the sound-meaning mapping function. In short, the grammaticality/acceptability of a string is determined by the overall linguistic system, not the language faculty alone.

Moreover, as J. Collins (2007) correctly points out, it is of no relevance whatsoever whether the native speaker is right or wrong in his judgements; all it matters is whatever information the linguist can extract from them. At best, we as native speakers have knowledge of what Ludlow (2011) calls 'surfacey facts' of language, as in those occasions in which we reflect upon the meaning of a word or a sentence, or even when we pass "judgement" on whether this or that sentence sounds OK or not. However, there is no reason to believe that we stand in any sort of epistemic relation to the underlying function in intension linguists (re)construct.[40]

More importantly, and as I will presently argue, the very connection between the grammar and the parser is entirely accidental; that is, even though there is ample interaction and various points of contact between the two, there are reasons to believe that they are entirely different mental realities, their connection being an evolutionary accident in the history of the mental architecture. Thus, if our capacity for drawing grammatical judgements is the result of such an accidental connection, it cannot be the case that the language faculty "fixes any linguistic facts"; its main role in cognition must involve something else. This, I want to argue, is the result of there being a substantive distinction between faculties à la Chomsky and modules à la Fodor (1983) —a distinction that is closely connected to the intensional differences among the distinct formalisms of what a computation

[39]At one point (ft. 12, p. 67), B. C. Smith (2008) mentions intuitions that can provide information about the interactions between competence and performance, but again, it is hard to imagine what sort of data these would be.

[40]Stich (1971) was on the right track in one respect; one could well study the "grammatical judgement" capacity, that is, the systems of the mind that generate grammatical judgements, which might as well have sui generis principles of their own, despite their close connection to the linguistic capacity. Where he went wrong was in concluding that this judgement capacity was the real subject-matter of linguistics. In any case, the grammatical judgement capacity does not correspond with the underlying function in intension.

is.[41]

Granted, both Chomsky's and Fodor's general frameworks are usually taken to be paradigmatic examples of "modular" approaches to the study of the mind. Nevertheless, and whilst it is true that both have inherited certain principles and properties from old-style "faculty psychology", they have been influenced by different traditions within it.[42] I have already mentioned one of Chomsky's historical precedents supra; Fodor (1983), on the other hand, has acknowledged the influence of Franz Gall in the outline of the vertical systems he proposed (essentially, Fodor only considers peripheral, perceptual systems to be modular). The discrepancy is perhaps clearest when we take language as the object of study, as Chomsky's approach focuses on the theory of the computation from lexical items to the interfaces, while Fodor has been mainly interested in the operations of the linguistic parser —that is, whatever mechanisms compute the underlying function in real time. In fact, the identity conditions Fodor (1983) put forward for modules (they are fast, mandatory, ecc.) do not quite apply to the language faculty qua function in intension. Uncontroversially, faculties and modules do not only encompass different identity conditions, their study also involves differing methodological and explanatory constraints.

In the case of modules, one is to study the actual step-by-step computations being carried out; the focus, then, lies on the memory load and the overall complexity that results from the operations being executed and the character of the representations manipulated.[43] Naturally, human processing abilities are limited in various ways, the memory capability being perhaps the most conspicuous case of all. Indeed, it is to be expected that working memory and general storage limitations will determine what sort of operations can at all be possible.[44] This general outlook differs greatly from a *theory of the computation*, with its focus on the "immediate successor relation", "size metrics", ecc.; in the precise case of linguistics, its methodology has focused on various linguistic tests, such as whether a structure respects the rule for conjunction, the intrusion of parenthesis, the ability to enter transformations, substitution tests and many others. I am outlining a distinction, as stated earlier, between the function being computed, and how it is in fact calculated in an implementation.

The birth and development of psycholinguistics is a case in point. The 1960s and 70s saw a proliferation of studies (e.g., G. A. Miller & Isard 1963, 1964) that attempted to experimentally probe the constructs that generative linguistics had put forward, a research programme that eventually developed into the so-called *derivational theory of complexity* (DTC; G. A. Miller & Chomsky 1963).[45] The DTC

[41] J. Collins (2004) runs, I believe, a similar argument.

[42] This is no news to them, of course. Fodor (1983), for instance, does point out that his is 'a notion of psychological faculty that is rather different from Chomsky's' (p. 3).

[43] This applies in toto, I believe, to CTM, of which Fodor is a staunch defender.

[44] Naturally, one has to take into consideration possible optimization operations, like the capacity for recoding information into more manageable chunks, as briefly discussed supra.

[45] I ignore that strand of cognitive science that tested "the psychological reality" of linguistic theor-

stated that there should be a correspondence between processing complexity and the number of linguistic rules employed to derive a sound-meaning pair; that is, the more rules a pair effects, the more difficult to process it should prove to be. However, experimental evidence quickly dispensed with the attempt to provide such a direct connection between the grammar and the parser (see the discussion in Fodor, Bever & Garrett 1974). Nevertheless, the advent of online methods such as the recording and measuring of reaction times provided appropriate means to determine the memory load and overall complexity of implementations, unearthing myriad properties of the parser. Indeed, it was precisely these successes that eventually resulted in the modular view of perceptual processes defended in Fodor (1983). In retrospect, however, one has to wonder why psycholinguistic evidence was ever deemed to be able of providing direct information on the underlying theory of the computation; perhaps a better grounding in computer science could have avoided this mistake.[46]

The actual point of contact between linguists and psycholinguists remains as unsettled as many other areas in the cognitive sciences, but I believe that the position defended in Fodor et al. (1974) is perhaps implicit in most studies. Therein, they argued that language processing involves the construction of the right *structural description* of the incoming signal, that is to say the right internal (and therefore mental) representation (p. 21); consequently, a theory of these mental representations becomes a necessity. Gestalt psychologists, as Fodor et al. point out, understood this point very well. They realised that what is important to behaviour is not only the proximal stimuli —the actual energy pattern received by the organism— but how these are represented (the distal representation); hence, their attempt to subsume the perceived stimuli under schemata. They certainly had a theory of the proximal stimuli —this was provided by the physical sciences— but the principles and properties governing their schemata-descriptions were of an altogether different nature, and they had no viable theoretical account of the mental representations (ibid., p.xvi). Fodor et al.'s proposal addressed precisely this fault, arguing that the grammar constructed by the linguist constitutes just such a theory, since it specifies the set of structural descriptions the parser must encode and decode. Naturally, both studies —that of the linguist and that of the psycholinguist— are independent endeavours.

A stronger claim is warranted, I think, as there seems to be very little evidence to suggest that the fact that language is put to use —that is, that linguistic representations are encoded and decoded by a parser— has *any* effect on the intrinsic properties of the mapping from lexical items to the interfaces. Clearly, this is different from the point made above that psycholinguistic data may form part of the

ies, by which it was meant, mistakenly in my opinion, whether the properties and rules that linguists postulated were used in real-time processing. Rather, a computational-level analysis constitutes a *psychological* theory of the cognitive capacity underlying behaviour. Cf. Chomsky's belief that linguistics is an exercise in theoretical psychology (1975b, p. 102).

[46]This is not to deny that there might be some truth to the DTC, as recently argued by C. Phillips & Wagers (2009) and Hornstein (2009). I will come back to this in later chapters.

evidence that linguists employ in their theories; rather, this is a point about mental architecture.

Note, as an illustration, the problem that linguists face in the study of the language faculty. At the most basic level, linguistic structures exhibit two main properties: they convey a meaning and, more prominently, on the surface at least, manifest a sound. At a minimum, then, language interacts with systems of meaning (known as the conceptual/intentional interface, as mentioned in the Introduction; C/I) and with systems of sound (in this case, the sensori-motor interface, also mentioned in the Introduction; SM). The task of the linguist, therefore, is to specify the function that generates structures according to the principles and pressures these two interfaces impose. On the C/I side, the meaning of sentences is compositional, and the manner in which lexical items combine is hierarchical (in terms of containment and dependencies), whereas the SM interface imposes a flat and linear signal that is employed in human communication (in both production and processing).[47] It follows that simplicity and efficiency phenomena —occurrences that are widely attested in complex systems (and not merely methodological goals of the theorist)— should be present in the mapping to both interfaces, albeit in different ways. As catalogued in Chomsky (2007a,b), the efficiency found in the mapping to the interfaces is *computational* in nature, which is to say that its operations follow a "formal" simplicity metric, rather than a "contentful" one. By a contentful metric, I have in mind whatever measures a system may employ in order to make the expression of information more effective —i.e., what Chomsky calls communicative efficiency.[48] Clearly, communicative efficiency ought to be much more prominent if the constant buzz of linguistic performance had any effect on linguistic structure, but this is not quite what we find.[49]

Take the case of "movement" phenomena, which by most accounts, is the result of an imposition of the SM system (see Moro 2000). Movement refers to the phenomenon in which some elements appear in a position that is different from where they receive interpretation, such as in *the car I saw yesterday*, where *car* is interpreted as the object of *saw* but does not appear next to it. In the case of the C/I mapping, the computational system preserves a copy of each instantiation of the lexical item *car* —that is, the generated structure would, very roughly, be something like *the car I saw the car yesterday*— preserving the right meaning relation: *car* is both the topic of the sentence *and* the object of *saw* —what has come to be known as the duality of semantics (i.e., argument structure and conversational

[47]I ignore the question of whether the "content" of words (or the concepts they stand for) is the result of a connection between words and the things in the world they refer to, which is usually what philosophers have in mind when talking about meaning.

[48]I will analyse the actual computational principles being proposed in the literature in chapter 3, including how they apply at both interfaces; for now the distinction between form and content will suffice.

[49]I do not ascribe to the suggestion that these efficiency considerations may be the result of "natural law"; neither do I believe that linguistic structures have anything to do with superficially similar patters in nature, such as Fibonacci sequences and others; these two views are defended in both Medeiros (2008) and Soschen (2006, 2008).

functions). It is immediately clear that the efficiency manifested in the C/I interface is formal in character.

In the case of the SM interface, however, only one copy of *car* is in fact preserved (only one copy is eventually pronounced), even if communicative needs would plausibly merit the externalisation of *all* copies. After all, the elimination of the copy after *saw* introduces a complication for the parser; as psycholinguists know full well, the processing system predicts the presence of an object after a verb such as *to see*, given that the sub-categorisation frame of the verb requires it, and this gives rise to "filler-gap" phenomena. Perhaps arbitrarily, Chomsky (2007a, pp. 12–4) sees the elimination of all but one copy in the SM mapping as another case of computational efficiency: the SM interface returns a structure ready to be externalised, and it is to be expected that only the last copy of *car* in the derivation is preserved, given questions of articulation effort, memory, ecc. That is, Chomsky reduces the elimination of all other copies to a matter of linear order efficiency (an SM effect, but a formal simplicity metric nonetheless).[50]

Note that the issues I am considering here are rather different from the factors that usually engage scholars interested in communicative efficiency. The claim here is that external pressures to communicate more effectively do not have an effect on linguistic derivations —that is, the computations that construct syntactic objects remain entirely unaffected by these influences. Studies such as those of Hawkins (2007) and Jaeger & Tily (2011), on the other hand, seem focused on a) the strategies that the performance systems employ to process and produce language more efficiently, and b) how these strategies 'correlate with typological patterns' (Jaeger & Tily, 2011, p. 323). There can be no doubt that something along these lines is true, but Hawkins (2007) is confused when he claims that there is a correlation between performance and 'conventionalized rules of grammar' (p. 105). He has shown nothing of the sort; what he has shown is that performance preferences determine, to a certain extent, the patterns that are preferred in behaviour, but that is a different matter. As linguists we are interested in *potentialities* —the set of possible structures— and no matter how simple or complex a syntactic object is, the character of the underlying computations is determined by formal and not contentful properties.[51]

This is related to some of the phenomena unearthed by disciplines such as historical linguistics and sociolinguistics, wherein some rather minor "economy-driven" changes have been outlined, such as phonetic reduction of speech forms, lexical and spelling changes and some others. When attention is diverted to changes in linguistic form, however, such as in the wide phonological change experienced in Germanic languages known as Grimm's and/or Verner's Laws, one finds that

[50]Further, Chomsky views the C/I mapping as more prominent than the SM mapping; in fact, he takes the latter to be secondary and ancillary, but for reasons that are not clear to me.

[51]At best, communicative efficiency would operate as some sort of filter so that the most appropriate structures are frequently employed in real-life interaction. If so, these external pressures ought to be allocated outside grammar (as I will do soon enough); in other words, competence is to be left alone.

structures and derivations do change, and sometimes become simpler, but with no connection whatsoever to communicative needs. That is, simplification phenomena seem to follow general principles of efficient mental computation and organisation —or as Bromberger & Halle (1991) put it, 'sound changes are due to the addition of phonological rules' (p. 71)— with no connection to external systems such as those that encode and decode language in human communication.[52] It seems certain, then, that the language faculty is an internal computational system whose derivations do not appear to be moulded by communicative pressures, at least not to a significant extent.

We should not conclude from this, however, that the interaction with the SM interface is entirely accidental, as there are reasons to believe that the SM mapping is in fact not *directly* related to communication. As stated, the very fact that displacement is conspicuous in language is explained by a condition imposed by the SM system, but the SM mapping *does not* predict that some computational units will in fact not be pronounced in communication. All the SM interface imposes is a linear and flat format that is appropriate for externalisation, but this is entirely compatible with a representation in which two copies of the same unit are present; a fortiori, this is precisely the type of string that one would expect if communicative needs played any role.

If there is any pressure to eliminate all copies but one —say, as mentioned earlier, because of articulatory effort, memory, ecc.— these are surely properties of the perceptual systems that externalise the SM string, not of the SM *interface* itself. It is a recurrent point about the role of the interfaces (see, e.g., Chomsky 2007b) that they receive structured expressions that can be viewed as "instructions" to external systems. In the case of the SM interface, the instructions are sent to the articulatory-perceptual apparatus, but it does not follow that these instructions are transparent to what the perceptual systems can in fact achieve. That is, it cannot be the case that limitations in memory and articulation impose (effectively, top-down) conditions, first, on the SM interface, and in turn, on linguistic structure and derivations —that is, ultimately, on the function in intension.

In these terms, there is certainly some tension in the suggestion that external limitations in memory et alia (performance variables) may constrain the operations of the underlying mapping function (competence). Indeed, in order to account for the observation that the mappings to the interfaces respect computational efficiency only, we must never abandon the point alluded to in ft. 52: intrinsic properties restrict and organise the architecture of the language faculty, and this surely includes the SM interface; it is the process that takes place once the SM instructions are sent to the perceptual systems that is secondary and ancillary.

To clarify matters further, consider the "evolutionary fable" presented in Chomsky (2000b), where he speculates that the emergence of the language faculty may

[52] This is compatible with the overall picture outlined in Chomsky (1965), where a distinction is drawn between perceptual systems and the innate ideas and principles that determine the acquired knowledge in a 'restricted and highly organized way' (p. 48).

have been the result of a random mutation in an individual. The mutation would have conferred this individual with a computational system of great power, but such system would not have generated sound-meaning pairs ab initio; it would only have started doing so once the individual was under the need to externalise the internally generated structures. There is a two-stage story to be told here. Firstly, environmental pressure to communicate forces the language faculty to create structures capable of preserving meaning while additionally providing an appropriate format for externalisation. That is, the SM interface returns a string (what used to be called a PF —for phonetic form— representation) ready to be externalised, but the externalisation process is the result of the manipulation (this is the second part of the story) that the pre-existent perceptual systems of the mind carry out on the SM string —a labour for which they were plausibly not designed. Indeed, there is no reason to suppose that the SM representations the perceptual systems inherit remain unchanged after the actual externalisation process. In other words, the linguistic structures we daily experience (in either production or comprehension) are the products of the perceptual systems, but these structures need not be isomorphic to what the language faculty generates. We know the final, externalised products, but we do not know the exact character of the structures at the SM interface.

Similarly, it is safe to assume that 'systems of planning, interpretation, reflection' (Berwick & Chomsky, 2011, p. 20) —the "thought systems" that employ C/I structures— manipulate these C/I "instructions" in ways that are at present a mystery; if it were not so, the relationship between language and thought would not be such a complicated affair. We *do* have a reasonable idea about the structures the language faculty sends over to the C/I interface for interpretation, but we just do not know how these are employed by other systems of the mind. Moreover, we have no reasons to believe that the "thought systems" do not change the C/I structures, and it is in fact very possible that they do.[53] If we knew the relevant facts, it is probable that we could make the mistake of believing that the resultant, manipulated-upon structures the "thought systems" create are precisely the structures that the language faculty generates. This is, in fact, the mistake that I believe is being committed in the case of the relationship between the SM interface and the articulatory-perceptual apparatus. Galileo was probably right when he complained that many philosophical mistakes could have been avoided if humans had been born (relevantly in this case) deaf.

It is, furthermore, plausible to postulate that, in the case of the SM interface, the computational units that are not to be pronounced in communication are eliminated at the stage in which the perceptual systems take over, and *not* within the SM mapping. If so, it is the connection between the SM representations and the perceptual systems that gives rise to some sort of communicative efficiency. As far as the language faculty is concerned, however, the mappings to the interfaces are under two very broad pressures only: on the C/I side, the computations must be

[53]Perhaps a relevant case is the relationship between *theory of mind* structures and the corresponding natural language sentences, which are rather different in detail, as I show in chapter 5.

hierarchical and compositional, while at the SM interface, the resulting representation must be linear and flat. Further transformations to the sound-meaning pairs are the result of the operations of other systems of the mind *when* these representations are employed beyond the intensional mapping function that generated them.

If this point of view is right, the language faculty and the perceptual systems that encode-decode linguistic structures in communication are entirely independent systems of the mind, and have come to be related by some random rewiring in mental architecture; out of this connection, the parser was born. As I will come to defend in later chapters, but have already anticipated above, the grammar-parser nexus is sufficiently explained by adopting an analysis-by-synthesis model for language comprehension, a solution that has been proposed for many other domains where similar problems arise. Still, this is a strikingly makeshift connection, but not an unsurprising one, on my view. If it is true that the grammar-parser connection gives rise to communicative needs, then it is this nexus that should be seen as entirely accidental.

Naturally, I am arguing for a much stronger position than merely holding faculties and modules to be different levels of explanation of the same phenomenon. Whilst there is no doubt that their study fall under different explanatory constraints as to what they are supposed to account for, and they clearly diverge on the type of methodology they employ, if my argument here is at all sound, these two theoretical constructs are not only different levels of explanation but ought to be regarded as different *mental realities*. Or in a less shocking tone, the grammar and the perceptual systems are different realities of mental life; and mutatis mutandis for the computations underlying their operations.

This is not a very surprising result when viewed within the prism of mathematical logic and computer science. In this chapter, I have framed the discussion of faculties and modules in terms of a distinction between the abstract implementation of a theory of the computation and the real-time implementation of a model of computation. These are not different levels of explanations, but completely different theoretical entities. Moreover, I have suggested that a recursively-specified algorithm is the most appropriate formalism for the former, while a TM fits in well with the purposes of the latter. This state of affairs transmutes piecemeal to the cognitive sciences. Unsurprisingly, then, recursion has been very central to linguistic studies (as I will show in the next chapter), while the TM has been the formalism of choice in cognitive psychology and most of the cognitive sciences, with their focus on processing. The point of this chapter has been to show that choosing one of the formalisms from the class of extensionally-equivalent systems I described supra is not a trivial matter; that is, intensional differences are important. At the very least, the choice of formalism identifies the level of explanation; here I have tried to show that it in fact establishes the sort of cognitive domain you are studying.

1.4 Segue

According to Hunt (1999, p. 29), the vast majority of studies in cognitive psychology pertain to Marr's computational level. That is, given a cognitive domain, a computational model is defined and experiments are then devised to evaluate it. This is not quite correct, as a computational model that has been devised to be experimentally tested is not a *theory of the computation*, but a proposal regarding the algorithmic level instead; that is, it constitutes a prototype of the actual real-time mechanism the experiments would eventually elucidate.

It seems to me that the distinction between determining the function being computed and the quest for discovering the algorithm that effects the real-time transformation escapes most cognitive scientists; and even those who extol the virtues of Marr's system seem to do so entirely in virtue of the practicalities of such a "division of labour" (Hunt, op. cit., for instance). The latter is certainly a real and positive attribute of Marr's framework, but there is much more under the bonnet. As I mentioned above, Chomsky's competence is rather similar to Marr's computational level, but the same cannot be said in relation to performance and the algorithmic level. For a start, Chomsky's competence is but one aspect of the overall linguistic capacity, a specific component that is embedded in the systems that underlie performance. That is, competence and performance are not two different levels of explanation of the *same* phenomenon; on the contrary, they constitute different *mental realities* that have come to be connected serendipitously. It is unremarkable, then, that studying Chomsky's theoretical constructs involves choosing among different computational formalisms whose selection comes accompanied with commitments that go beyond the usual methodological and explanatory concerns.[54]

Be that as it may, there is some use to following a step-by-step explanatory plan such as the one I outlined at the end of section 1.1, and the next three chapters are orderly devoted to the three stages outlined therein. It is to the description of the mechanical procedure (i.e., the algorithm) underlying the language faculty that the next chapter is dedicated, including ample discussion on the distinction between recursive mechanisms and recursive operations.

[54]I am of course ignoring much material here, including Marr's third level of explanation (the neural implementation). I will also not discuss any of the other levels of explanation that have been proposed in the literature, such as those of Dennett's, Newell's or Pylyshyn's. I believe that the study being carried out here is narrow enough to justify these omissions.

Chapter 2

Recursion in linguistic studies

2.1 Recursion and Discrete Infinity

But the procedure of language is not simply one whereby a single phenomenon comes about; it must simultaneously open up the possibility of producing an indefinable host of such phenomena, and under all the conditions that thought prescribes. For language is quite peculiarly confronted by a unending and truly boundless domain, the essence of all that can be thought. It must therefore make infinite employment of finite means . . .

Wilhelm von Humboldt

Über die Verschiedenheit des menschlichen Sprachbaus und ihren Einfluss auf die geistige Entwicklung des Menschengeschlechts (1836).

Discrete infinity refers to the property that the array of possible linguistic expressions (where an expression is understood as a structure composed of discrete constituents) is infinite, a fact that is argued to be reflected in the 'behavior of speaker[s] who. . . can produce and understand an indefinite number of new sentences' (Chomsky, 1957, p. 15). It has been identified as a central property of language —and it consequently constitutes an explanandum for linguistic theory— ever since Chomsky's *The Logical Structure of Linguistic Theory* (LSLT), a work written in 1955-56 but only published in 1975. In LSLT , Chomsky is rather explicit in stating that recursion constitutes an explanation for this phenomenon; indeed, it is therein claimed that it is the 'recursive character' of phrase structure rules, to be introduced presently, that allows for the 'generation of infinitely many sentences' (pp. 171–2). This is mirrored in works that *were* published in the 1950s, where it is stated that if a grammar contains 'recursive devices, it will produce infinitely many sentences', a statement that appears in both Chomsky (1956, pp. 115–6) and (1957, p. 24). The connection between recursion and infinity is maintained some twenty years later in the introduction to the published version of LSLT, wherein '[t]he recursive property of the grammar' is identified as 'the property of generating an infinite class of structures' (p. 16). Finally, a recent statement of Chomsky specifies that all recursion means is discrete infinity, the need to enumerate the potentially infinite number of expressions (in Piattelli-Palmarini, Uriagereka & Salaburu 2009, p. 387).

Discrete infinity should not be confused, however, with the "creative aspect of language use", the observation that speakers/hearers can understand and produce an unbounded number of sentences in appropriate circumstances and relatively free of external stimuli (see, for instance, Chomsky 2006). That is, discrete infinity is but one aspect of the creative use of language, but the latter is a much richer notion. A fortiori, recursion should not be taken to explain creativity, a mistake that has been pointed out in numerous occasions (for example, in Chomsky 1975b, p. 230 and 2006, p. xviv) but continues to be committed (as in Pullum & Scholz 2010).[1]

There are two strands, I believe, to explaining discrete infinity, but they are not always clearly separated in the literature. In fact, much of the controversy to be reviewed here stems, as I will attempt to show, from very simple mistakes in both approaches. The first take on the matter is a rather straightforward affair, but it will be nonetheless claimed here that it accurately describes the introduction and right application of recursion in linguistic theory —at least in Chomsky's writings. Showing this is the case will involve an exegetical exercise of Chomsky's oeuvre, and while I will not claim to have been exhaustive, or that he has always been entirely consistent, a common thread will nevertheless be delineated. The second approach, on the other hand, involves drawing too close a connection between linguistics and mathematics. In one sense, this translates into an attempt to mathematically prove that the set of grammatical sentences is infinite; that is, language is reduced to an abstract mathematical system, supposedly in order to formally prove the infinity of language. In a more nuanced but related sense, this strand mistakenly focuses on certain rewriting rules and the sort of structures they are said to be capable of generating. The second half of this section will here argue that the second strand is not only beset with significant problems, but essentially misguided.

The first strand on explaining discrete infinity makes the very general point that in order to account for the unbounded novelty in linguistic behaviour, it is necessary to postulate an underlying mechanical procedure —a computational system— as the number of possible sentences one can understand/produce surely exceeds storage. This was already clear to Humboldt, as the quotation heading this section shows, but it was not until the 1930s that we were provided with 'a clear under-

[1]Moreover, Pullum & Scholz (2010) seem to equate creativity with "originality", but this is certainly not the connotation that Chomsky has attributed to the creative aspect of language use.

standing of how a finite mechanism can construct an infinity of objects' (Chomsky, 2002, p. 48); namely, once mathematical logic had succeeded in formalising a computation.

In the particular case at hand, the formalism Chomsky employed in the 1950s was Post's production systems. As shown in chapter 1, the general format of these systems is a mapping of the form $g \longrightarrow h$, where the whole approach 'naturally lends itself to the generating of sets by the method of definition by induction' (Post, 1943, p. 201); or to follow Sieg (1997), generative procedures in general are underlain by 'finitary inductive definitions' (p. 166). This general recursive property of production systems was in fact recognised and explicitly endorsed by Chomsky from very early on. As Chomsky & Miller (1963) pointed out, the $\longrightarrow$ relation mediating the conversion of some structure $\phi_1, \ldots \phi_n$ into some structure ϕ_{n+1} can be interpreted as 'expressing the fact that if our process of recursive specification generates the structures $\phi_1, \ldots \phi_n$, then it also generates the structure ϕ_{n+1}' (p. 284). In the next section I will show that it is this precise interpretation that runs through Chomsky's vast output, and despite the numerous changes and developments the theory has undertaken, I will claim that he has in fact been rather consistent in his formulation. That is, unbounded linguistic behaviour necessitates a computational system, and the generative procedures that ought to interest linguists (more accurately, those linguists interested in characterising the function in intension at the heart of the language faculty) are underlain, by definition, by inductive definitions (that is, recursion).

Whilst it is safe to say that much of the confusion in the literature is the result of missing this very point, it is also true that the second approach to explaining discrete infinity has contributed a great deal to this state of affairs, and I now turn to this. Pullum & Scholz (2010) constitutes a paradigmatic case of the approach I have in mind. Therein, these authors take issue with the prevalent belief of what they call the "infinitude claim"; that is, the claim that, for any language, the set of possible sentences is infinite. Their discussion is rather eclectic in structure, but I will only centre on their appraisal of what they call the "standard" argument supporting infinitude claims. The standard argument has three parts, according to them: (i) there are some 'grammatically-preserving extensibility' syntactic facts of the kind *I know that I exist*, *I know that I know that I exist*, ecc. (p. 115) that lead us to believe that (ii) there is no upper bound on the maximal length of possible sentences (at least for English); these two facts together, in turn, warrant the conclusion that (iii) the collection of all grammatical expressions in a given language is infinite.

The argument is well-put together as far as it goes, and their main worry falls not on the move from (ii) to (iii) (which is simple mathematics, they tell us), but on the transition from (i) to (ii). Interestingly, they do not tell us what is actually necessary to warrant the troubled transition; instead, they dismiss three different possibilities that could be employed for its justification: the use of an inductive generalization, mathematical induction, or by arguing from generative grammars (the latter they take to be, strictly speaking, systems of rewrite rules only; see infra). As they consider that all these recursion-related constructs fail to justify the move from (i) to (ii), they consequently conclude that recursion is not a principal property of language.

It is not clear to me at all that any of these strategies have ever been explicitly employed in the literature in order to support the standard argument —at least not in the sense that Pullum & Scholz seem to have in mind. More importantly, later on in their paper (p. 124 et seq.) they point to the (supposedly widely-held) assumption that languages are "collections" —in the sense of how this notion is treated in set theory— which they link to specific concomitant repercussions for the linguist. This consequently translates into a rather heavy burden on the linguist, as the whole issue turns out to be predicated on too close a connection between natural languages and mathematical systems, to the point that the infinitude of the former is to be proven by the standards that we impose upon the latter.[2] That is, if the collection of grammatical sentences is an infinite set, and mathematicians have developed ways to prove the infinity of a given set (which usually involve recursion), these tools ought to be employed in linguistics to evaluate infinity claims.

This is, however, unwarranted. It is certainly true that many linguists have employed mathematical techniques to study natural languages —these remain in use because they are useful— but this should certainly not result in a reduction of linguistic phenomena to abstraction. This is precisely what mathematical linguistics does —viz., reducing structure to sequences of symbols— but this plays a rather limited role in linguistic explanation. Instead, linguists typically focus on informants' grammatical judgements in order to unearth the underlying structure of strings. That is, linguists focus on the structure that a certain mental state —viz., the linguistic capacity— imposes upon the strings, and not on the strings themselves in isolation from these judgements. Ultimately, it seems that these authors confuse the use of mathematical concepts as a useful tool-kit for a call to reduce linguistics to mathematics, but no such thing ought to be accepted by the working linguist.

Furthermore, there are a few more problems with the "standard argument" as conceptualised by Pullum & Scholz. Note, first of all, that their argument is focused on proving whether a "given language" is infinite, but this is not how a standard version of the infinitude claim ought to be construed. Discrete infinity is a claim regarding the finite mechanism at the heart of the language faculty, not a fact about any particular language (less even is it a claim about all languages). Clearly, the very fact that a new-born can acquire any language is exposed to indicates that the underlying computational system remains uniform across different languages. A fortiori, and in combination with the previous points, it is certainly true that all languages of the world manifest a gigantic set of possible sentences, exceeding memory and necessitating a computational system.

[2]Langendoen (2010) also orbits these very issues, and ends by urging the field to come to an agreement 'upon a basis for determining whether a language is closed under one or more of its iterative size-increasing operations' (p. 45).

That notwithstanding, if their approach was at all right, one would expect to see the opposite reasoning (that is, the finiteness of a given language) to be placed under the same burden, but this does not appear to be the case for languages that prima facie lack the 'grammatically-preserving extensibility' syntactic facts mentioned in (i) (pp. 130–131). Surely a similar argument would arise: (i') there are some syntactic facts of language *A* that suggests this language lacks grammatically-preserving extensibility structures, which leads us to believe that (ii') there is indeed an upper bound on the maximal length of its sentences; therefore, (iii') this language is a finite collection of sentences. Clearly, the transition from (i') to (ii') is as troubling as that of (i) to (ii) —but only if we grant Pullum & Scholzs burden. But the burden must be rejected, as I believe that the role of recursion in language ought to be defended in the terms I outlined above.

Relatedly, whilst it is certainly possible that not *all* languages make use of the very structures Pullum & Scholz identified in their argument (viz., sentences inside sentences; or self-embedded sentences), this is clearly a moot point here, as recursion has nothing special to do with self-embedded structures. As we will see, however, vast swathes of the literature draw a very close connection between these two constructs. In a sense, this is understandable, as it is very natural to put together an argument —just like Pullum & Scholz (2010) do— that uses sentences like *I know that I know that I exist* as clear examples of extensibility syntactic facts that preserve grammaticality. Still, there are many other constructions that meet these criteria —such as *and*-conjunctions— and it goes without saying that employing any of these instead would eliminate the centrality of self-embedded sentences in such arguments.

The focus on the self-embedded sentences brings me to another factor at play within the second strand on how to account for discrete infinity, a factor that explains the close connection many scholars perceive between recursion and self-embedding; namely, a number of misguided beliefs regarding the manner in which Post's production systems have been employed in linguistics.

Recall that Chomsky introduced production systems into linguistics in the 1950s, and furthermore, that he was well aware of the recursive generation at the heart of these systems. Naturally, the scheme $g \longrightarrow h$ had to be adapted for linguistic usage, and the symbols Post made use of were consequently replaced with linguistic terminals and non-terminals; among others: S stands for sentence, NP for noun phrase, VP for verbal phrase, D for determiner, ecc. As a result of this, an *internal* application of recursion within rewriting rules can be identified, one that is particular to linguistics. In order to clarify, consider the small sample below, where the rewriting rules should be understood as transformations of the strings on the left-hand side of the arrow into the strings on the right-hand side.

(2.1) (a) $S \rightarrow NP\ VP$
(b) $NP \rightarrow D\ N$
(c) $VP \rightarrow V\ NP$

(d) $NP \rightarrow N\,(NP)$

(e) $VP \rightarrow V\,S$

It has come to be customary to describe the last two rules as recursive, as categories to the left of the arrow are reintroduced on the right-hand side. To be more precise, there is a direct recursion in the case of (d), but indirect in (e) —i.e., an S rule, (a), generates NP and VP, and (e), a VP rule, reintroduces S. This recursive property, however, is an internal application *within* production systems and applies to specific rules only. Indeed, this internal application ought to be kept distinct from the general and global recursive property of *collections* of rewriting rules *qua* production systems; i.e., the connotation I identified in section 1.1 and immediately above.

That aside, these two rules can naturally be seen as being capable of generating the sort of structures that have come to be known as self-embedded (or more generally, nested structures). Thus, rule (e) can generate sentences inside other sentences, such as *John thinks (that) [Michael killed the policeman]*, while rule (d) can return NPs inside other NPs, as in *John's [brother's [teacher's book]] is on the table*. Given that self-embedded sentences were being generated with specific recursive rules in the 1950s, it is perhaps not surprising that these very structures are usually described as being recursive; hence, the close connection between recursion and self-embedding.

As we will see later on, this connection is the result of conflating structures and rules, but let us now, arguendo, restate the argument presented in Pullum & Scholz (2010) in slightly different terms. Note, first of all, that we can reapply rules (d) and (e) and generate ever-longer sentences, and no doubt there's no such a thing as *the* longest sentence, an interesting fact about language in itself. The point, however, is that the successive applications of a recursive rule can be stopped at any particular moment, and the resultant output would be another unit in the set of grammatical sentences —a symptotic tendency of the grammar. Consider Pullum & Scholz's example for a moment; we can obviously keep adding an *I know that* in front of any sentence, and *I think that I know I exist* and *I think that I think that I know I exist* are two different units in the set of English sentences. Obviously, the recursive application of such rules can generate more and more structures; that is, more and more units that are part of the set of expressions a grammar can construct.

This is meant to illustrate that the magnitude of a set (and whether this magnitude is infinite) is usually calculated by, first, placing it in a one-to-one correspondence with the natural numbers (which makes the set enumerable, Kleene 1952, p. 3); then, it is to be shown that there cannot be a non-arbitrary bound on the size of individual elements; if so, the set of sentences would be countably infinite.[3] Pullum & Scholz's example meets these criteria, and the sentences below show further constructions for which a non-arbitrary bounded length cannot be established:[4]

[3]Cf.: '[o]bviously the sentences can be put in one-to-one correspondence with integers so that the language will be denumerably infinite' (Chomsky & Miller, 1963, p. 284).

[4]Unbounded sentence length does not mean that any sentence is of infinite length, only that there

(2.2) The mouse [the cat [the dog chased] bit] ran.

(2.3) The man [that wrote the book [that Pat read in the cafe [that Mary owns]]].

(2.4) John's [brother's [teacher's book]] is on the table.

In these examples, further material can appear either in the centre of the structure, or in one of the edges, which moved Chomsky & Miller (1963, p. 290) to define the former as self-embedded expressions and the latter as either right- or left-recursive. Naturally, they are all instances of nested structures, but the distinction Chomsky & Miller (1963) drew remains applicable, even if the terms have changed (see infra).

Let me summarise some of the points I have made so far. Formal approaches to demonstrating that the set of possible sentences may be infinite have focused on self-embedded sentences, and since these specific strings were once upon a time generated with recursive rewriting rules, the nested constructions themselves have come to be defined as recursive sentences. There is nothing wrong with this in principle, even if there are reasons to believe that language manifests a much more significant type of recursive structure. The problem is that the literature contains myriad studies that focus on whether particular languages exhibit this specific type of structure (a self-embedded one) with the intention to show that if there is a language that prima facie does not generate such structures, it is then the case that recursion is not a central property of the language faculty. Such a conclusion is entirely unwarranted, however. For a start, structures other than the self-embedded can be employed to demonstrate that the overall system goes beyond finiteness; indeed, coordinative sentences such as *the lion attacked the tiger and the rat bit the cat and the dog chased the pigeons and...* can very well be employed to make the point that there are no grounds to establishing an upper bound on sentence length, automatically undercutting the importance of the centrality of self-embedded sentences in such arguments. Another important point to keep in mind is that the overall system of production systems is underlain by "finitary inductive definitions" (indeed, this is true of *any* generative system), and consequently, the actual nature of the generated structures is completely irrelevant; recursion remains a global property of generative systems and should not be confused with the outcomes of such systems.

*****

In the late 1950s and 60s, the mathematical properties of specific rewriting rules (or more precisely, specific combinations of rules that could be said to encompass particular grammars) were thoroughly studied and this resulted in rather substantial discoveries concerning expressive power (i.e., what strings could a grammar

is always a longer (but ultimately finite) sentence.

generate) and complexity.[5] Unfortunately, these developments have also given rise to certain misunderstandings in linguistics and cognitive science at large. One that is particularly troubling has to do with the distinction in expressive power between recursion and iteration *when* these two properties are applied to rewriting rules; a distinction that all but disappears when, say, *partial recursive functions* and an iterator such as a *Turing Machine* are compared: both formalisms can compute exactly the same input-output pairs.

The discussion in Bloom (1994) suffers from these very shortcomings and it is worth spending some space clarifying what is at stake here. Firstly, however, let me clear a terminological point. In this critique piece of a proposal by Michael Corballis, Bloom (1994, p. 178) points out that Corballis, like Chomsky, takes Humboldt's 'infinite use of finite means' quote as a definition of generativity; an astonishing claim in any case —how is that a definition?— but one that is certainly false in the case of Chomsky —at best, it would refer to discrete infinity, another matter altogether. Rather, Chomsky (1965) unambiguously states that *generate* is being employed in the same way it is used in logic —or in Post's combinatorial systems (p. 9); namely, in terms of how a formal proof is constructed from axioms and rules of inference (Chomsky, 2007a, p. 6). Indeed, as noted in section 1.1, Post's system is composed of a set of postulates from which the set of logical propositions can be generated, a "generalisation by postulation" formalism that eventually developed into production systems. It is precisely in these terms that a generative system ought to be construed.[6]

More to the point, Bloom (1994) argues that in a general sense, both iteration and recursion can be considered as examples of generativity —perhaps a trivial point— even if it is well-known, according to him, that 'iterative mechanisms' like a finite-state Markov system cannot account for some linguistic structures, which suggests to him a significant difference in expressive power between recursion and iteration (p. 178). Naturally, he can only conclude thus because he describes recursion in relation to rewriting rules, a property he surprisingly claims to be 'of *some* systems of rules' (ibid.; my emphasis). In order to explain what this means, he compares two different samples of rules and claims that recursion emerges from 'the manner in which the rules go together' (p. 179) —a holistic property of rule systems.

The last point can be swiftly dismissed. Recursion is a general property of production systems in toto and sometimes a narrower characteristic of particular rules when a category appears on both sides of a rewriting rule. Nevertheless, it is worth discussing the differences in expressive power between iteration and recursion within rewriting rules, as it will resurface later on —furthermore, the confusion stemming from this is endemic in parts of the literature. Indeed, it is not rare to find papers in which it is claimed that certain structures can only be

[5]These developments literally created the field of *mathematical linguistics*, a discipline that is of great importance to computer scientists.

[6]Cf. Soare (2009), wherein it is stated that Post's system is generative instead of computational because it provides an algorithm for listing a set instead of computing a function.

generated recursively, but not iteratively, a claim that is usually framed in terms of production systems. As stated above, it is imperative to emphasise that this is a point that only applies within these systems (if at all, see infra).

Chomsky (1956) is the starting point of the formal discoveries alluded above; therein, he showed that production systems could be employed to characterise different collections (classes) of *formal grammars* and the sequences of symbols (also called strings; these sequences constitute the corresponding formal languages) that these grammars are said to generate/produce. A ranking can then be so devised as to classify these grammars in terms of their *expressive power*; that is, the sort of strings a specific formal grammar can generate. Some of the most important classes are the following:

- Finite-state systems (a Type 3 grammar), which exhibit rules of the type $A \longrightarrow \begin{cases} aB \\ a \end{cases}$, capable of generating a^n strings, where n is a number.
- Context-free grammars (Type 2), with rules such as $A \longrightarrow \alpha$ and exemplars of the following type: a^n b^n.
- Context-sensitive grammars (Type 1) include rules like $\phi A \psi \longrightarrow \phi \alpha \psi$ and can generate a^n b^n c^n strings.
- Type 0 grammars can generate recursively enumerable languages by employing rules such as $\alpha \longrightarrow \beta$.

In this classification, the upper-case symbols stand for non-terminal elements (such as an S, for sentence), the lower-case ones represent terminals (such as the head of a noun phrase; as it may be: *car*), and the Greek letters symbolise strings of terminals and non-terminals. The word "context", in turn, refers to the material that appears on either side of the arrows of rewriting rules. Thus, Type 2 grammars are context-free because A can be rewritten as α regardless of what material A is surrounded by; in contrast, in a Type 1 grammar A can only be replaced by α if it is surrounded by both ϕ and ψ. This ranking has come to be known as the Chomsky Hierarchy, a containment hierarchy of different classes of grammars, as shown in Fig. 2.1.

Chomsky (1956) showed that certain linguistic expressions could not be generated by some of the grammars he identified, but he did this by abstracting away from the underlying structure of these sentences —that is, the linguistic expressions were paired with sequences of lower- and upper-case symbols. This is a point about the expressive power of grammars that has come to be known as "weak generative capacity" (i.e., the generation of strings); the linguist, however, is interested in "strong generative capacity" (the generation of structure), a notion that still awaits formalisation.[7] More specifically, Chomsky (1956) was able to demonstrate

[7]The Chomsky Hierarchy pertains to a study of what I termed above "the theory of the computation". A corresponding hierarchy of automata —that is, the machines that recognise the different

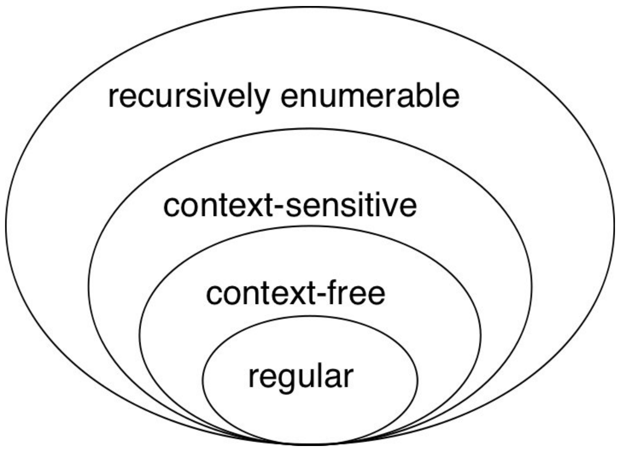

Figure 2.1: Chomsky Hierarchy

that finite-state systems were incapable of generating well-attested linguistic structures such as self-embedded expressions, as these require rules like $S \longrightarrow aSb$ —a context-free rule.[8] The expressive power of natural language, it was then argued, is at least context-free, and this imposes a necessary, if not sufficient, condition on any linguistic theory: all formalisms must be able to exhibit the right expressive power.[9]

The differences among these grammars are real enough, but these facts should not be translated into substantial general computational properties vis-à-vis the recursion/iteration pairing. Clearly, these results are rather narrow in scope — they are nowadays mainly of interest to computational linguists— and it is certainly the case that they are not directly related to computability theory. Naturally enough, Post never came to know the differences in expressive power within distinct classes of production systems, but even if he had, the advances of mathematical linguists do not invalidate the facts he helped established; namely, that his recursive-generation formalism is extensionally equivalent to a TM (an iterator).

Moreover, given that "finitary inductive definitions" underlie production systems in toto, it is rather misleading to refer to finite-state systems as "iterative mechanisms". J. Collins (2008a, p. 53) is certainly right to refer to finite-state machines as the 'simplest recursive device', and the inability to distinguish between

formal languages— can also be outlined, but their analysis introduces memory and space factors, and it therefore belongs to a study of real-time implementations.

[8]This is not quite correct, but I will come back to this in section 2.3 below.

[9]The Chomsky Hierarchy has been augmented with grammars and languages that were unknown to Chomsky at the time, and it is now believed that the expressive power of natural language is in fact *mildly context-sensitive*. Furthermore, there has been a convergence of linguistic formalisms that exhibit the right expressive power (see Joshi, Shanker & Weir 1990 and Stabler 2011 for details).

the general recursive property of generative systems and the internal recursive application within production systems can only obfuscate well-established issues to do with expressive power and general computational properties.

The point of this short discussion is that there is no difference in expressive power between recursion and iteration. There are differences in expressive power among different grammars of the Chomsky Hierarchy, but none of these systems can be directly identified with either recursion or iteration per se.

In this section, I have tried to show a number of things. Firstly, that one of the aims of the linguist is to account for the constant, novel linguistic behaviour humans manifest, and this requires postulating a generative system —of which recursion is a central feature. However, it does not follow that the linguist is under the burden to formally prove the actual infinity of the set of possible sentences as if this "set" were, literally, qualitatively similar to the sets mathematicians study; all that is required is the observation that storage is impossible. Contra Tiede & Stout (2010), then, it is not the case that infinity and recursion are such interconnected notions that one must assume one and derive the other from it. Rather, the unbounded and novel linguistic behaviour is the observation —the explanandum— and a recursively-specified computational system is the explanans.[10] Contra Mukherji (2010, p. 213), in turn, self-embedded sentences are not the evidential basis for recursion; novel linguistic behaviour is.

2.2 The progress of the theory

In section 1.1, I outlined four different connotations of the term recursion that I believe appropriately identify well-defined theoretical constructs: *a*) definition by induction, its primary meaning; *b*) a *recursor*, that is, a general property of computational systems; *c*) a feature of real-time processes when these contain an operation that calls itself; and *d*) as an architectural attribute of structures, an X within an X. In this section, I will try to show that there is a common thread running through Chomsky's vast output; namely, the second connotation here outlined. Still, this brief exegetical excursus into the history of recursion within the different developments of generative grammar will also show that all four meanings pop up in Chomsky's writings. This should not be taken as a charge of inconsistency or contradiction against Chomsky, I am merely pointing that it is necessary to disentangle the different connotations in order to elucidate his leitmotif —something that many scholars have failed to do.[11]

[10]The very same point applies to Luuk & Luuk (2011), who argue that discrete infinity is a redundant notion in linguistics, and this, according to them, does not bode well for recursion, given the close connection between the two. They also think, for independent reasons, that recursion itself is redundant, but this is based on confusing a *theory of the computation* with a *model of a computation*. Bickerton (2009) makes exactly the same mistake, but I will only retake this issue in chapter 3.

[11]I am here going to focus on Chomsky's writings only; this is just a reflection of the fact that it was he who introduced recursive devices into linguistics (see ft. 15, though). That aside, the next section will catalogue the treatment recursion has received at the hands of many other authors. The

It is customary to refer to Chomsky (1965) as perhaps the first mature statement of the conceptual underpinnings of generative grammar. Indeed, the first chapter of this book devotes much space to such foundational notions as competence, performance, the language acquisition device or the nature of rewriting rule systems. Chomsky (1957), on the other hand, was nothing more than a set of lecture notes for MIT graduates, and the material therein constituted but an extremely small part of LSLT, a work that was only published 20 years later. Indeed, the only explicit mention of "recursive devices" in Chomsky (1957, p. 24) is, confusingly, in terms of the closed loops of finite-state machines.

Be that as it may, Chomsky (1963), Chomsky & Miller (1963) and G. A. Miller & Chomsky (1963) are the first publications in which competence and performance, and many other issues, are introduced and discussed at great length. Furthermore, and as stated before, it is in Chomsky & Miller (1963, p. 284) where the global recursive feature of production systems is clearly delineated, and this connotation needs to be distinguished from the internal recursive applications of production systems.[12] Thus, in the early days of generative grammar the computational system (CS) underlying the language faculty, a production system, was divided into two components: the base (composed of rewriting rules that returned strings with associated phrase markers) and the transformational system (a component that would convert phrase markers into other phrase markers, preserving structure).[13] In Chomsky (1957), the recursive property of certain rules is ascribed to the latter system, while Chomsky (1965) assigns it to the base component.[14] A few years later, and in much more general terms, Chomsky (1967) states that 'the output [of the language acquisition device] is a system of recursive rules' (p. 7).

This is clearly and entirely independent of Bar-Hillel's suggestion (1953) that the social sciences might benefit from employing recursive definitions. Presumably, Bar-Hillel was interested in a more precise definitional technique for theoretical constructs —or as he put it, recursive definitions are 'a perfectly legitimate means of concept formation' (p. 162)—, which may or may not bear on discrete infinity. Nevertheless, Chomsky (1955, p. 45) manifests his agreement in spirit, while two years later sees 'success along these lines unlikely' (Chomsky, 1957, p. 58). Still, Bar-Hillel's point does not seem to relate directly to generative mechanisms, even if it is clearly possible that inductive definitions may be useful for

introduction to Chomsky (2002), written by Rizzi & Belletti, is a useful short guide for what I am doing in this section.

[12]This needs to be qualified somehow. The LSLT does not explicitly mention the competence/performance distinction, but it is clearly implicit throughout. Further, the general recursive property of generative systems seems to be outlined in pp. 194–5, but I would not want to press the validity of my interpretation on this specific point.

[13]Interestingly, Marr (1982) considered Chomsky's theory a "computational-level" account because he perceived that it was the structure-preserving transformations that 'look[ed] like computations' (p. 28).

[14]In LSLT, Chomsky goes to great lengths to rid the first component of "recursive statements", as recursive rules are called there. He codified this as a "non-recursion requirement" for the base component (pp. 517–18), which he vigorously defended at length.

certain constructs.[15]

Keeping to the early days of generative grammar, we also find some references to recursive phenomena other than mechanisms. Thus, Chomsky (1965, p. 196) considers whether multiple branching sentences (such as *John, Bill, Tom, and several of their friend...* should be regarded as left- or right-recursive structures (to follow the nomenclature introduced in Chomsky & Miller 1963; see infra), while in the 1960s edition of his *Language and Mind* book, the recursive property of language is identified as an embedding operation that forms $[S\ldots]_s$ within other structures (p. 27); or less roughly, as an NP that is rewritten into (DET) N (that S), where S is reintroduced from an early stage of the derivation (p. 128). I will treat these distinctions in much more detail in the next section, but it is imperative we keep all these constructs —recursive mechanisms, recursive structures, and embedding operations (which are something else altogether)— clearly separate.

By the 1970s and 1980s, most of the rewriting rules were in fact being eliminated from syntactic theory, perhaps completely so by the time Chomsky (1986) was published.[16] With the advent of the so-called *government and binding* theory, the emphasis was placed on structural configurations and constraints, as it was then argued that rewriting rules were merely recasting lexical properties, and therefore were redundant (Chomsky, 1986, p. 83). Following Mukherji (2010), then, this was a period in which the CS underlying language was not the main focus of study; rather, linguists turned their attention to structural constraints, which eventually resulted in the X-bar scheme, a structural geometry that could faithfully model any type of syntactic phrase.

In the 1990s, though, the *minimalist program* redirected linguistic theory to the study of the mapping function from lexical items to the interfaces, and one single mechanism was identified for this purpose: *merge* (Chomsky, 1995b).[17] Chomsky has been rather clear that recursion underlies *merge*, as it is a procedure that 'recursively constructs *syntactic objects* from [lexical] items...and syntactic objects already formed' (Chomsky, 1995b, p. 226) —in turn, a *syntactic object* is recursively defined in p. 243.[18]

A recent description (namely, Chomsky 2008), delineates *merge* in very general terms as a set-theoretic operation in which repeated applications over one element yield a potentially infinite set of structures, drawing an analogy between the

[15] Further, it is a mistake to suggest, as Karlsson (2010, p. 50) does, that Bar-Hillel's paper constitutes a (re)introduction of recursion in linguistic theory. By reintroduction I suppose Karlsson is referring to the fact that the IV. century BCE Indian grammarian Panini devised a system of recursive rules that was very similar to Post's production systems. In any case, Bar-Hillel's suggestion is unrelatable to recursive rewriting rules or generative systems tout court.

[16]N. V. Smith (1999, p. 63) points out that rewriting rules were being dispensed of as early as 1962, though.

[17]To be more precise, Mukherji (2010) points to Chomsky (1995a) as the point in which the CS is again placed at the centre of linguistic concerns.

[18] According to Chomsky (1995b, p. 243), syntactic objects are of two types, they are either lexical items or complexes of lexical items of the following type: K=[γ [α, β]], where α and β are objects and γ is the label of K (the latter constitutes the "recursive step" of the definition).

way *merge* applies and the successor function, one of the first functions to be recursively defined (cf. Kleene 1952, p. 21). The successor function also underlies what is known as the "iterative conception of set" (Boolos, 1971; T. Forster, 2008), a process in which sets are 'recursively generated at each stage' (Boolos, 1971, p. 223), which has moved T. Forster (2008) to state that what mathematicians really mean by the iterative conception of set is, 'in the terminology of modern computer science at least', the recursive conception of set (p. 97). Strictly speaking, this is incorrect; for a process to be recursive, according to the 'terminology of computer science', it must contain chains of deferred operations, the result of an operation calling itself, but this is not the case here. Rather, by 'recursively generated at each stage' we understand the 'repeated application of the successor function', drawing our attention to the analogy between 'the way sets are *inductively generated*... and the way the natural numbers... are inductively generated from 0' (Boolos, 1971, p. 223). That is, it is the successor function that reapplies at every stage; consequently, it is mathematical induction that *justifies* every generated set, but this makes it not a recursive process. The process really is iterative, it just happens that every stage of the iteration is recursively generated —a subtle distinction between process and generation.[19] To keep with the analogy to the natural numbers, we can follow Benacerraf (1965) and ascribe to syntactic objects the same properties he ascribes to the natural numbers. Thus, we can advance that the set of grammatical sentences forms a "recursive progression", as manifested in the overall 'structure which the members of the sequence jointly exhibit' (ibid., p. 69). In this sense, linguistics, like arithmetic, may well just be the science that 'elaborates the abstract structure' of a specific type of progression; namely, a recursively-specified, linguistic progression of objects.[20]

This is perhaps the clearest statement yet that *merge*, qua generative system, is underlain by "finitary inductive definitions".[21] In a somehow similar analysis, Tomalin (2007) reconsiders the role of recursion in linguistic studies and concludes that this term should be replaced by "definition by induction" in order to avoid confusion, given that the CS of language is 'a component that enables syntactic objects to be generated by means of a procedure that utilises inductive definitions' (p. 1799). I do not think this is entirely correct, however. There are two things to distinguish here: one is the fact that inductive definitions are a central part of generative systems, the other is the way in which syntactic objects are defined — the two are entirely independent. Indeed, in this paper (and also in his 2011 article), Tomalin seems to believe that given that syntactic objects are recursively defined in Chomsky (1995b, p. 243), this is what it is meant for *merge* to 'recursively

[19]Contra Pullum & Scholz, again, linguistic formalism can profitably employ concepts from, in this case, set theory, but it does not follow that linguistic theory must be built from the theorems of set theory.

[20]Cf. the connection Chomsky (2007a) draws between language and arithmetic.

[21]It is certainly in this context that most of the discussion on *merge* and recursion in Soschen (2008) should be understood, particularly her statement that 'singleton sets are indispensable for recursion' (p. 199).

construct' them (ibid., p. 226). The two are obviously related, but they should not be conflated. Furthermore, I think he overstates one aspect of the Recursion Convention (RC) he inherits from Soare's discussion; to wit, it is one thing to say that a recursor is equivalent to a TM or the lambda-calculus, it is another thing completely to state that recursion is being used to connote all these constructs (and all the terms Soare and Tomalin list). Clearly, this is a strand of the RC that is being exaggerated.[22]

To carry on with the general point I am making here, the connection I am drawing between production systems and *merge* is certainly not fortuitous; nor could it be said that there is a schism between the two formalisms, or that Chomsky has been inconsistent regarding this point.[23] Instead, I think Chomsky has been rather persistent on the matter, as the following quotations illustrate. Starting from the 1960s, he has clearly stated that a grammar 'must meet formal conditions that restrict it to the enumeration of recursive sets' (Chomsky, 1965, p. 208), a statement that is repeated twice in the first edition of his 2006 book, originally published in 1966: 'in general, a set of rules that recursively define an infinite set of objects may be said to generate this set' (p. 112); and, 'generative grammar recursively enumerates structural description of sentences' (p. 165). This is still defended in Chomsky (1981, pp. 11–3), at the height of *government and binding* theory; that is, at the precise moment that the CS was being put aside in favour of structural constraints. More recently,Chomsky (2000a) has been rather explicit that all recursion means is the need to enumerate the potentially infinite number of expressions (p. 19).

Recall that a recursively enumerable set is a collection of items for which there is an algorithm that can list all its members. This is also known as a computably set, another instance of the general gist of the RC: recursive is taken to be synonymous with computable. In this context, it is to wonder whether Chomsky has at all been affected by the RC; that is, if the situation outlined in Soare (1996) applies to him as well. Interestingly, Chomsky *has* in fact pointed to the influence

[22]I would not want to overstate my differences with Tomalin, as I do believe his analysis is, with some minor modifications, on the right track. Nevertheless, I believe that my discussion here improves upon his analysis. Firstly, Tomalin's 2006 book focuses on Chomsky's work prior to *Syntactic Structures*, even if the very last chapter —which was published as Tomalin (2007), effectively— focuses on recursion within the *minimalist program*; a great leap in time, surely. It is also worth noting that Tomalin (2007) is supposed to be a "reconsideration" of recursion in syntactic theory, but there is no progression from *Syntactic Structures* to the *minimalist program* in Tomalin (2006) to offer the necessary background for this reanalysis. A similar, limited range of study is also the case for Tomalin (2011), as the focus there lies exclusively on Chomsky (1957). Apart from being unable to nail the precise interpretation of recursion in Chomsky's writings, these papers by Tomalin are also unable to trace the common thread in Chomsky's writings I am about to delineate.

[23]It is worth mentioning that Bickerton (2009) also traces the history of how production systems were replaced by *merge*, but he proves utterly incapable of recognising a) the general and global recursive property of productions systems and b) the recursive generations *merge* effects. The latter is the result of not understanding what the *iterative conception of set* entails, while the former can only be put down to an unawareness of the relevant facts concerning Post's systems. As a result, he is left wondering why Chomsky keeps insisting on this recursive property, given that he cannot see it anywhere; I hope my discussion here relieves him of such bewilderment.

of mathematical logic on numerous occasions. For example, he has stated that the formalism employed in the 1950s was part of 'a particular mathematical theory, namely, recursive function theory' (Piattelli-Palmarini, 1980, p. 101), a recast of the theory of computations (Chomsky, 1986, p. 56), chosen for no other reason that it was then deemed to be the most appropriate at the time (Chomsky, 2007b, p. 15).

Remarkably, it would not be surprising if all these quotations by Chomsky could be easily subsumed under the widespread influence of the RC. A recent introductory book on Chomsky's thought, for example, states that 'via the Church/Turing thesis, computation is just recursion defined over a finite set of primitive functions' (J. Collins, 2008a, p. 49), a statement that is as close to the spirit of the RC as one may expect to find in the cognitive science literature.[24] At this point, then, it seems 'a reasonable conjecture' to claim that at root 'there is only one fixed computational procedure that underlies all languages' (Chomsky, 1995b, p. 11); a "recursive" *merge* in the sense elucidated here.

Note that what I have discussed so far tells us nothing about what sort of operation *merge* effects; at best, it tells us that qua mechanical procedure *merge* is some sort of recursor. In order words, all we have done is establish what *merge is*, but not *what it does* (what operation it carries out qua mechanism) or *how* it proceeds (whether recursively or iteratively qua recurrent function). The last two aspects pertain to an analysis of an algorithm's abstract implementation, i.e. Marr's computational level of analysis, which is precisely the type of analysis that a derivational account of competence typifies. Chapter 3 will be devoted to these issues; here, I have attempted to specify the nature of the CS underlying the language faculty.[25]

On a related note, the identification of *merge* as the 'simplest form of recursive generation' (Chomsky, 2007b, p. 15) is entirely independent of the character of the structures it generates. More specifically, and even though there are grounds to believe that basic linguistic structure is indeed recursive, the latter is an independent discovery. The conflation of recursive mechanisms and recursive structures into one general phenomenon is an endemic problem in the literature, and I will devote the next section to this unfortunate state of affairs. Before I move on to that, though, I will outline the general structural pattern all syntactic phrases conform to —a result stemming from X-bar theory.

According to Pinker & Jackendoff (2005), a recursive structure results when 'a constituent... contains a constituent of the same kind' (p. 203), which is a particular

[24]In a personal correspondence with the present author, Chomsky has stated that 'there is a technical definition of "recursion" in terms of Church's thesis (Turing machines, lambda calculus, Post's theory, Kleene's theory, ecc.)', the only used he has ever used, 'a formalization of the notion algorithm/mechanical procedure'. Further, he admits he is 'always tacitly adopted the Recursion Convention' (May 2009).

[25]Incidentally, the distinction between what *merge* does and how it proceeds holds, mutatis mutandis, for the iterative conception of set. In fact, T. Forster (2008) resorts to metaphorical language when it comes to describing the operations underlying the iterative conception of set; namely, a *lasso* (the image is attributed to the philosopher Saul Kripke) gathers some elements, while a *wand* creates a new set out of this gathering.

reading of the general definition I provided in section 1.1. Naturally, it is rather important to establish what the "X within an X" stands for; at the very least, one would want to focus on a recursive structure that is of some relevance for cognitive science. In the case of Pinker & Jackendoff (2005), *kind* refers to the category of the element that *heads* a constituent. Accordingly, an XP within an XP would be a case of recursion whereas an XP inside an YP would not be. This is a recast of the definitions provided in Chomsky & Miller (1963) and Chomsky (1965), in fact. The latter, for instance, draws a distinction between nested constructions —i.e., 'phrases *A* and *B* form a nested construction if *A* falls totally within *B*, with some nonnull element to its left within *B* and some nonnull element to its right within *B*' (p. 12)— and self-embedded structures —'phrase *A* is self-embedded in *B* if *A* is nested in *B* and, furthermore, *A* is a phrase of the same type as *B*' (ibid.)— but nowhere are these defined as recursive —neither is it stated that these structures constitute the locus of recursion within the language faculty.[26]

Nevertheless, an XP that is embedded inside a YP is part of an expression that exhibits some general features of linguistic structure. For a start, the sort of hierarchy underlying linguistic structure is binary only —that is, for any level of a given tree scheme, each node is split up into two "unambiguous paths", to borrow Kayne's phrase (1981); consequently, binary hierarchy is a feature of self-embedded structures as much as it is of simply embedded ones. Indeed, there are grounds to believe that language manifests a much more general type of recursive structure. At the appropriate level of abstraction, a structure that contains an instance of itself appears to be a property of any type of syntactic structure: every syntactic phrase (NPs, VPs, ecc.) accords to the same geometry, an asymmetric structure [Specifier [Head - Complement(s)]] (Moro, 2008, p. 68). This is the direct result of X-bar theory, graphically represented in Fig. 2.2.

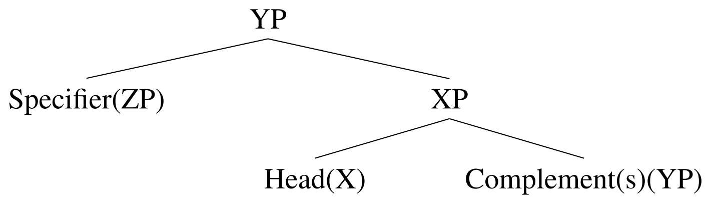

Figure 2.2: Asymmetric S-H-C structures

Therefore, a Complementizer Phrase (viz., the top node of a clause) is a complex [S[H-C]] structure composed of a number of architecturally-equivalent but simpler [S[H-C]] structures. As Moro (2008, pp. 205 et seq.) shows, all human languages appear to follow this scheme, despite some variation in the linear order. Linear order is not the key property; rather, the central point is the basic hierarchical con-

[26]It is true, however, that Chomsky & Miller (1963) discuss left- and right-recursive structures. I will come back to all these constructs in the next section.

figuration: S is always more prominent than [H-C] and H is always more prominent than C.

Recent developments suggest that the Specifier position does not really exist; that is, that the Head of any constituent does not "project" any object into this position (Chomsky, 1995a; Starke, 2004). If so, the overall architecture would be something like this: [. . . Head. . . (Compl). . . [. . . Head. . . (Compl). . .]. . .].[27] As Jayaseelan (2008) puts it, the union of X and YP would be the result of *first merge*, while *second merge* generates the ZP-X configuration, *e così via*. The point I am making still applies; that is, there is a sort of general recursive structure that is present in all languages, and this is independent of the most usual forms of self-embedding. That is, a [NP[VP]] structure is ultimately [. . . H. . . C. . . [. . . H. . . C. . .]].[28]

In this sense, then, structural recursion appears to come for free, but remains an interesting and surprising fact about language. It in fact identifies natural language as a subcategory of infinite systems, one that manifests a specific type of embedding of endocentric and asymmetric structures. As such, "category" recursion is a subtype of structural recursion (i.e., self-embedding is a subtype of general embedding), and it is perhaps in this sense that contemporary debates on the universality of embedding ought to be understood (see chapter 5).[29]

2.3 Conflation of Structures and Mechanisms

The cognitive science literature contains a great number of works in which the recursive nature of specific structures is defined in terms of how they are generated. The most conspicuous example is perhaps that of Corballis (2003, 2007a,b, 2011), as recursive rewriting rules are therein not only employed to generate self-embedded sentences, they are also used as a sort of definitional technique so that the resultant structures are identified as recursive too. This is of course not necessary; after all, in these very papers Corballis also considers the possibility of recursive structures in general cognition, but with no attempt whatsoever to delineate the underlying systems that would generate them. Indeed, all he does is describe mental structures that are structurally similar to self-embedded expressions, and no more is in fact needed.[30] Nevertheless, even such prima facie innocuous expository strategies can result in misunderstandings of much more substance, as I will here now argue.

[27]Granted that, and following Brody (1994), there are still some relevant specifier-head relations, at least in configurational terms, as I will discuss in the next chapter.

[28] It is in these terms that Medeiros's discussion (2008) of "recursive templates" should be understood, something I will come back to in successive sections.

[29]Note, also, that the nested structures linguistic derivations generate are asymmetrical, a fact that the "iterative conception of set" must account for too (see chapter 3 and cf. Boeckx 2009c).

[30] Furthermore, you cannot run the same argument with, say, *merge*, as this mechanism proceeds in the same manner for both embedded and self-embedded structures alike. Contra Corballis (2011, p. 229), it is not so much that rewriting rules are "old-fashioned" but useful to make a point. Rather, they play no role whatsoever in linguistic theory, and they are a hindrance in popular descriptions.

To begin with, recall that Chomsky (1956) made use of production systems qua grammatical models in order to make a point about the expressive power of different formal languages and their relation to natural language properties. Further, Chomsky used these systems in perfect consonance with how Post treated rewriting rules: as string substitution operations —that is, no internal structure was presupposed. Take the *aaabbb* string as an example. If the *a's* stand for the subject of a sentence and the *b's* for the verb, a string of these characteristics can at first sight mimic self-embedded structures such as *the mouse the cat the dog chased bit ran*. With no further assumptions regarding the internal structure of such a string, however, the *aaabbb* exemplar can also be generated by a concatenation of non-recursive rules, such as: $A \longrightarrow aB$, $B \longrightarrow aC$, $C \longrightarrow aD$, $D \longrightarrow bE$, $E \longrightarrow bF$ and $F \longrightarrow b$. Naturally, a context-free rule of the type $S \longrightarrow a(S)b$ is a much simpler way of generating a potentially infinite collection of *a's* and *b's* —namely, $a^n b^n$.[31]

In fact, one of the reasons recursive rules of this type were introduced into the theory was in order to simplify the grammars; cf. in precisely this context, the statement in Chomsky (1956, pp. 115–6) that 'if a grammar has no recursive steps... it will be prohibitely complex' —that is, the grammar would otherwise be an infinite list of rules, one for each structure. Perfors et al. (2010) provide (partial) confirmation for this intuition by employing a qualitative Bayesian analysis to calculate the ideal trade-off between simplicity of a grammar (treated as a prior probability) and the degree of fit to a corpus (treated as the likelihood). Even though recursive rules, they tell us, are costly because they predict sentences that are not observed in a corpus (which hurts their goodness of fit; see pp. 161–164), the calculation ultimately returns, perhaps unenlightening, a grammar with recursive and non-recursive rules as the preferred choice. I qualify these results as uninformative because they do not seem to differ from what was being proposed in the 1950s. Granted, this sort of analysis offers a much more formal understanding, but one should not mistake formalisation for insight if the issues were already well-understood. Further, there are two aspects of this work that are somewhat troubling. First, the study places too much emphasis on the actual "observed" data found in corpora. These are not to be disregarded, obviously, but linguists ought not to forget that the actual subject matter, that is, the actual phenomenon to be explained, remains the cognitive state that underlies linguistic behaviour. Secondly, it is an obvious point to make that this analysis only applies to those theories that postulate production systems as grammars; change the formalism to *merge*, and this study does not have anything to say about the role of recursion therein.

More importantly, a context-free rule is closer to capturing the fact that the corresponding self-embedded sentence *does* have an internal structure. That is, the recursive rule $S \longrightarrow a(S)b$ generates a string of a specific type —viz., *[a [a [a b] b] b]*— which is closer to the architecture of *[the mouse [the cat [the dog chased] bit]*

[31]In order to be more accurate, I should state that a finite-state grammar can only approximate the output of a context-free grammar; that is, the former *can* generate some context-free strings such as *aabb* and *aaabbb*, but it cannot generate $a^n b^n$ strings, where the value of n can run ad infinitum. See Rogers & Pullum (2011) for some details.

ran]. It is precisely because of this point that Chomsky argued that the expressive power of language must be at least context-free. Be that as it may, however, the correspondence remains rather weak, and in a sense, misleading. Frank (2004) retakes these issues and points out that the fundamental difference between finite-state and context-free grammars lies in that fact that in the latter the *ab* pairs are introduced at the same time in the derivation —supposedly a crucial distinction. Whilst that is certainly correct, the *ab* pairs so generated do not, as a matter of fact, bear any equivalent structural relation with the subject-verb pairs of a linguistic expression. Rather, subjects and verbs enter into an "agreement" relation that is based on abstract morpho-syntactic features, something that lies beyond what a formal (read: artificial) language can model.[32] Ultimately, and this was certainly the case in the early days of generative grammar, the rewriting rules of the base component return, strictly speaking, *strings*, while its associated structure was a matter of interpretation from the part of the linguist.[33] The point I have made in this paragraph already tells against drawing too close a connection between recursive structures and recursive mechanisms, but there are more aspects to this state of affairs.

A second and rather obvious point to make is that the existence of recursive structures in a given cognitive domain does not necessarily mean that they were, or indeed, that they must be, generated recursively. Discussing this detail unavoidably anticipates some issues to be treated in the next chapter —issues to do with derivational properties, that is, features of the step-by-step computations the language faculty effects (a study of the theory of the computation, or abstract implementation)— but it is important to at least establish what this at all entails. In a general sense, the point I want to make already follows from section 1.1: any input-output pair that a recursive function can compute/derive can also be computed/derived, iteratively, by a TM, and this applies to any sort of structure. That aside, the literature seems to be confusing a recursive step in a computation (i.e., the manner in which an operation proceeds) with an operation that embeds elements into other elements.

Fitch (2010) is a case in point. Therein, he offers an analysis of what he calls three different interpretations of recursion for biolinguistics, by which he really means how recursion is understood in metamathematics, computer science and linguistics. He seriously misunderstands the role recursion has in the first two disciplines, but I should probably not get into unnecessary exegesis here, interesting though it would be.[34] More importantly for our purposes, Fitch puts forward two problematic claims; firstly, that a recursive rule has the property of self-embedding

[32]Or as Chomsky (2002, p. 109) puts it, a formal language lacks morphology, phonology, pragmatics and basic syntactic properties such as the dislocation principle.

[33]I would not want to overstate this point, though. The formalism described and defended in Frank (2004), for instance, is literally a tree-rewriting system of rules known as a tree-adjoining grammar.

[34]Among other things, it would allow me to dissect the belief conveyed in Barceló-Coblijn (forthcoming) that the computer science connotation Fitch describes constitutes the relevant meaning for linguistics —an astonishing claim.

(p. 78), and secondly, that it is a 'linguistic stipulation' for a self-embedding rule to entail a self-embedded structure (p. 80), which I suppose carries over to simply embedding rules and embedded structures.

As stated earlier, rewriting rules, technically speaking, only return strings, not structures, which is presumably one of the reasons for which rewriting rules were eventually eliminated from linguistic theory (cf. J. Collins 2008a, p. 58); a fortiori, there is no such thing as a self-embedding rewriting rule. It is, however, certainly true that the long-held stipulation Fitch identifies was meant to salvage the discontinuity between rules and the resultant structures —a discontinuity which holds for any type of linguistic expression though, and not merely self-embedded ones. This, however, is not related to recursion in any way, as there is no point in claiming that recursive rules and recursive structures are linked by stipulation while *merge* remains a recursive mechanism due to that fact that it contains a self-embedding operation.

Both *merge* and production systems are recursive devices for the same reason; that is, they are both generative systems underlain by the successor function. It is nevertheless true that the replacement of one for the other involves the further postulation of an operation that 'embeds (an object) within some construction...already formed' (Chomsky, 1995b, p. 248). The last point should not be confused with the aforementioned definition of *merge* as a procedure that recursively constructs syntactic objects. That is, embedding objects into other objects is *what merge* does, while the recursive generation that takes place at each stage of the derivation is one way in which *merge* proceeds; clearly, the two should not be conflated. Consequently, it is important to stress that recursion and (self)embedding are two different things, as (for that matter) are (self)embedded structures and (self)embedding operations. To believe otherwise is to confuse what an operation does with how it proceeds.[35]

As a matter of fact, the research Fitch outlines has to do with self-embedded structures and nothing else. He defends the idea that constructing the right interpretation of a self-embedded structure constitutes an 'empirical indicator' for recursion (pp. 80–81), and, therefore, an independent empirical way to investigate the meanings assigned to certain strings could be devised as to obtain behavioural evidence of self-embedding.[36] This is, however, an unfortunate lapse into semantics for a notion that is intrinsically syntactic; that is, a notion that only refers to the nature of the generative or processing mechanism, independently of the interpretation the resultant structure receives. More importantly, we are owed an explanation as to why this could tell us anything about recursive generation (com-

[35]The recursively-defined factorials from the previous chapter illustrates this point rather well. Recall that the operation they effect is a multiplication of a variable n by the factorial of $n-1$, but this has obviously nothing to do with recursion per se. Given how close the connection between embedding and recursion is perceived to be within cognitive science, its conflation is perhaps not a surprise.

[36]That is, if subjects demonstrate that they interpret a self-embedded structure correctly, this is taken as evidence that they are making use of a recursive operation.

petence) or processing (performance), as opposed to the nature of self-embedded structures only.

Surprisingly, given the other claims in the paper, Fitch does get it right, at least for a study of processing, when in the section on computer science he discusses ways of finding out, without looking at the source code directly, whether an algorithm in a given machine is proceeding recursively; namely, by 'probing registers and logic gates with *measurement* devices' (Fitch 2010, p. 78; my emphasis). This is the correct way of evaluating real-time *processes*, but it disappears from the rest of the paper, which turns its focus to the evaluation of structures, an entirely independent matter (in fact, I will provide a study of real-time implementations precisely along these lines in chapter 4).

This position is perhaps too widespread for comfort, and has recently even made it into the popular press (see M. D. Hauser 2009). However, when the actual claims are laid out and analysed, the precise nature of the data becomes clear. Roeper (2007, 2009), for instance, while also defining recursion as an operation that puts something inside itself, offers a panoply of interesting facts about something that has little to do with it, such as the diverse range of self-embedded sentences that different languages exhibit, the path the child goes through in the acquisition of these structures, or the character of the syntactic derivations that generate them. *Merge*, however, remains a recursive generator for reasons that lie elsewhere.

The present discussion does not, obviously, rest importance to self-embedding in natural language, a phenomenon that has been closely linked to semantics: a way of 'organizing and constraining semantic information' that appears to be construction- and language-specific (Hinzen, 2008, pp. 358–9). However, the conflation, apropos recursion, between what an operation does and how it proceeds is rather common in the literature. van der Hulst (2010b), a collection of papers on the role recursion plays in the study of language contains many examples of such unfortunate mistakes.[37]

Some of the contributions of this collection (namely those of Karlsson, Verhagen, Kinsella, Harder, Hunyadi) discuss various constructs and it is necessary to clarify what they stand for; these include center-embedding rules, tail-recursive rules, the sort of structures these generate, their relationship, ecc. A center embedding rule is supposed to generate nested structures in which a sentence is embedded in the middle of a bigger sentence, like those which were called self-embedded expressions above: *[the mouse [the cat [the dog bit] chased] ran away]*. A tail-recursive rule, on the other hand, embeds elements at the edge of sentences, either on the left-hand side (*John's [brother's [teacher's book]] is on the table*) or on the right-hand side (*the man [that wrote the book [that Pat read in the cafe [that Mary owns]]]*) (these can be simply called either left- or right-branching sentences).

[37]I will only provide the full reference of those contributions I treat to some extent, otherwise I will just state the last name of the author(s). For a full review of the overall collection, see my Lobina (2011a), sections of which are adapted here.

These terms, however, while constituting a redundant excess in terminology, have absolutely nothing to do with the recursive character of the rules themselves, only to the type of embedding the resultant expression manifests.[38]

A center-embedding rule, after all, is not a rule in which the reflexive call occurs, literally, in the middle of a derivation. The employment of the term tail-recursive is perhaps more unfortunate, as this widely-used term from computer science refers to a process in which the recursive call of the algorithm occurs at the very end of the derivation (Abelson et al., 1996). Quite clearly, a nested structure on the left-hand side of a sentence cannot be the result of a tail-recursive rule if the derivation process undergoes left-to-right applications of rewriting rules. In a nutshell, these terms refer to specific properties of the structures, not to recursive mechanisms or operations.

Perhaps the clearest case of the conflation I am denouncing is to be found in Parker (2006) and Kinsella (2009) (both papers belong to the same author, actually; a version of the former appears under the author's married name —Kinsella— in van der Hulst 2010b). Strikingly, this author quite literally defines iteration and recursion in terms of what sort of operation they carry out. Thus, iteration involves, she tells us, repeating an action an arbitrary number of times (Kinsella, 2009, pp. 115–9), while recursion implicates embedding an object within another instance of itself (ibid.). On the other hand, she claims to derive these definitions from the computer science literature (ibid.), but not a single reference is in fact provided —perhaps unsurprising, given the obvious mistakes. In the end, it is quite clear that she is focused on recursive structures only (in fact, she offers a clear statement of what "structural recursion" is on page 114). Indeed, she quite explicitly states that recursion 'inherently involves semantics' (p. 127), and it boils down to two constructions only: possessives and subordinate clauses (p. 150).[39,40]

Going back to the collection of papers aforementioned, some of its contributors seem to have a much stronger claim in mind. Karlsson, following Parker (op. cit.), contends that "nested recursion" rules (i.e., centre-embedding)[41] cannot be reduced to iterations (while tail-recursion supposedly can), a claim that is repeated by Harder (p. 239) and, with qualifications, in Zimmerer & Varley's contribution (p. 397). They could not possibly mean this as a general point about computability theory, however. In fact, one of the references mentioned in van der Hulst (2010b), albeit indirectly (p. 347), —namely, Liu & Stoller (1999)— offers a framework that provides automatic transformations of any type of recursion into iteration, an "optimization technique" that can cope with the most complex of recursive rela-

[38]The literature contains some other terms for essentially the same constructs, but there really is no point in cluttering the terminology any further. The ones employed here suffice, I believe.

[39]Cf. Hornstein & Pietroski (2009), where it is stated that only adjunction is the truly recursive part of grammar.

[40]Furthermore, Kinsella proves utterly incapable of recognising the recursive quality of *merge*, which she confusingly denies by stating that *merge* is a procedure, while recursion is a characteristic (ft. 20, p. 129).

[41]Verhagen, p. 103 tells us that this is sometimes referred to as "true recursion", but where is this the case? No reference is provided.

tions, such as multiple base cases or multiple recursive steps, of which Fibonacci sequences are an example (contrary to what Fitch 2010, p. 78 seems to think).

If this point does not hold for mechanisms, one may still wonder if it holds for structures. Some of the papers just mentioned seem to believe that self-embedded sentences cannot be converted into other types of phrases, but this is explicitly denied in Kinsella's contribution (p. 188). As she makes clear, even if languages like Pirahã (Everett, 2005) were shown to fully do without self-embedded sentences, this does not translate into an "expressive" loss to their speakers. That is, there is no reason to believe that Pirahã cannot 'express [similar] concepts using alternative means' (ibid.). Indeed, a self-embedded sentence such as *the mouse the cat the dog chased bit ran away* seems to be easily converted into either *the dog chased the cat that bit the mouse that ran away* (which I would call a right-branching structure) or *the dog chased the cat and the cat bit the mouse and the mouse ran away* (conjunction).[42] The latter would be considered an iterative structure by Karlsson op. cit.; that is, a flat output structure with no increased depth,[43] much like the multiple branching sentence mentioned supra, or the many other enumerations one can easily imagine (cf. Uriagereka's chapter in Piattelli-Palmarini et al. 2009).[44]

A rather more subtle conflation of self-embedded sentences and recursive generation is to be found in Arsenijević & Hinzen (2010), which is rather unexpected, as one of the authors has elsewhere correctly characterised *merge* as being underlain by the successor function (namely, in Hinzen 2009). In fact, Arsenijević & Hinzen (2010) start by appropriately describing *merge* as a recursive function (p. 166), but immediately after, in a blink-and-you-miss-it sort of moment (that is to say, within brackets), they further state that this is reflected in the observation that linguistic expressions may exhibit structure in which a category becomes a part of a bigger category of the same kind. What then follows is a rather inconsequential

[42]This is an old point, actually. Langendoen (1975, p. 199) mentions that English extraposition allows the conversion of centre-embedded structures into right-branching ones; in addition, left- or right-branching sentences can easily be converted into coordination.

[43]Be that as it may, there is an interesting mismatch between the structure of so-called iterative sentences and their prosody. Whilst it is certainly correct to state that parataxis exhibits a flat structure, their prosody, as Wagner (2010) points out, may well be hierarchical. A sentence such as *Hermia and Lysander and Demetrius* can be pronounced as either *Hermia and (Lysander and Demetrius)* or *(Hermia and Lysander) and Demetrius* (where the parentheses mark the intonational phrases), but the underlying syntactic structure does not change at all; that is, *Hermia*, *Lysander* and *Demetrius* are not hierarchically related. In the case of left-branching structures such as *John's mother's car*, however, while the prosody is manifestly flat, there is a clear intricate hierarchy between *John*, *mother* and *car*; it is not John's car, it is his mother, ecc. Confusingly, the focus in Wagner (2010) lies on the "transmutation" of compositional structure between syntax and prosody, but he seems to ignore the obvious mistmatches I am pointing out.

[44]Strikingly, however, Kinsella (2009, p. 119) remarks that even though the equivalence between recursion and iteration is well-established in computer science (she uses the factorial as an example of a task that can be computed recursively or iteratively), it does not hold for linguistic structure, as "semantics" forces a precise interpretation; hence, her belief, in that publication at least, that recursive structures cannot be reduced to iterative ones.

discussion as to the locus of self-embedded sentences.[45] In fact, recursive functions and self-embedded sentences are not comparable in nature or structure. The latter, but not the former, exhibits a containment relation between two elements of the same category; recursive functions, on the other hand, are defined with a two-equation system in which the recursive step specifies values in terms of previously defined values, but there is no self-embedding. The self-calls merely point to the values a simpliciter function calculates, but there is no hierarchy of functions. In a related manner, MacWhinney (2009) offers an account that is conceptually very similar. Right after stating that he will identify recursion with an inductive analysis —following Tomalin and Bar-Hillel (op. cit.)— he provides examples for the 'syntactic effects of recursion', namely relative clauses (p. 406), apparently oblivious to the fact that self-embedded sentences and inductive definitions bear no relation whatsoever.[46]

In yet another strand to this problem in terminology, while some scholars state that embedding is responsible for the recursive characteristics of language (Boeckx & Uriagereka, 2007),[47] they join the vast majority in the field in disregarding the underlying recursive generation. Indeed, for the most part linguists talk of recursion with no reference to mechanisms. Neeleman & van de Koot (2006), for instance, tell us that recursion results 'if there is a set of primitive trees that can be combined into a structure in which the root node is repeated in the yield' (ft. 5, p. 1530), but we are not told anything regarding what sort of mechanism effects the combination of these primitive trees. The onus seems to be on the structures that are repeated, that recur, as Hinzen (2008) explicitly states: 'it is only particular domains of syntactic organization that productively "recur" at all' (p. 359).

Other scholars sometimes refer to recursive generation very loosely as a process that applies over its own output (Boeckx, 2009a; Hornstein, 2009; Everett, 2009), which is of course trivially true, but it does not differentiate it from iteration, as discussed in section 1.1.[48] I will come back to this particular mischaracterisation in chapter 3, but this at least brings me to the final point I want to make on the conflation between mechanisms and structures, this time in terms of real-time processes, a topic I will treat at great length in chapter 4.

[45]Namely, they compare two proposals regarding how self-embedded sentences are generated: either directly by *merge* or as a result of the combination of *merge* with the constraints the C/I interface imposes. They offer plausible reasons to favour the second possibility.

[46]Just like Arsenijević & Hinzen (2010), MacWhinney does not regard self-embedded structures an independent property of the mind, but the result of the combination of various other systems. According to the *functional* strand of linguistics he favours, self-embedded structures can only come about by the merging of many subsystems: auditory organisation, articulatory organisation, lexical organisation, positional patterns, storage and mental models.

[47]They cite the LSLT in this context, but given that they do not include a page number, and that I cannot connect this claim to any portion of that book, this seems to me like a vacuous citation.

[48]This mistake also appears in Corballis (2011, p. 5). The latter contains a barrage of errors, though: p. 7 conflates recursive generation and embedding; on p. 23, I-language is described as a "language of thought"; and on ft. 12, p. 229 Corballis offers his peculiar take on the history of generative grammar when he states that "deep structure" gave way to "Universal Grammar", and this in turn to I-language; and the latter, by implication, to the language of thought.

Take the methodology of the artificial grammar learning (AGL) paradigm as an example. Under this paradigm, experiments proceed more or less as follows: subjects are initially exposed, in the training phase of the experiment, to regular patterns of strings. Then, they are told that the set of strings they have just seen was not random, but generated by a specific set of rules (a grammar). Subsequently, they are exposed to a new set of strings and asked to identify which strings from this new set conform to the patterns they saw in the training phase. If the subjects are successful, this would indicate, it has been argued, that they have learnt the rules, extrapolating the grammar. In the specific case of self-embedded expressions, Corballis (2007a) has proposed that in order to demonstrate 'true recursion' (p. 702), subjects would have to realise that some $a's$ are paired with some $b's$ within the a^nb^n strings. That is, 'recursive syntactic parsing' would be demonstrated if subjects bound *ab* pairs from the outside inwards (ibid.), and "cues" could be employed to indicate these internal relationships.

In some of the experiments carried out after Corballis's paper was published (see my Lobina 2011a for details), the *ab* pairs were linked up by phonetic features. Clearly, subjects would have to keep these features in memory in order to link the different pairs, but this does not mean that the processing is recursive in any sense. As I explained in section 1.1, the memory load exerted by real-time recursive processes results from self-calls and the chains of deferred operations subsequently created. This is certainly not the case for the general strategy of keeping the right phonetic feature in memory and linking its bearing element with the next element that carries this same feature. More importantly, matching features among long-distance elements bears no relation to the recursive rewriting rules that are supposed to be *literally* employed in the processing of paired elements. That is, by linking certain elements by phonetic feature, and then eliciting subjects to construct the right pairs, one is in fact changing the operation that is supposed to be under analysis (and perhaps even changing the nature of the underlying grammar that has been posited).

A recursive process does indeed result when a given procedure calls itself, but this self-call is *simpliciter*; in the factorial example of section 1.1, the factorial of 4 becomes (4 $\times$ (factorial 3)), and then the factorial of 3 turns into (3 $\times$ (factorial 2)), and so on until it reaches the simplest case, the factorial of 1, for which the case base immediately returns a value. As a consequence of this, an internal hierarchy among the operations develops so that the factorial of 4 cannot be calculated until the factorial of 3 is, and the latter will not be completed until the factorial of 2 is, and so on; it is the operations, in other words, that are hierarchical. This is not the case for the feature-linking operation in either respect. Firstly, a simpler self-call does not take place; instead, the same operation applies to different variables. Secondly, no hierarchy among the operations develops as, quite clearly, a string such as $a_1a_2a_3$ does not necessitate that the *b* elements appear in any particular order for the correct linking of features to take place; this is the case in some experiments merely as an artefact of the way the experimenter creates and presents the materials. The resultant memory load therefore follows from this linking of features, and not from the parser rewriting an S into $a(S)b$, and then the resultant S into another $a(S)b$, and so on and on. At best, the latter would be an added operation, but would only be therein where recursion takes place, and not elsewhere. At worst, it could be a by-product, meaning that recursion would effectively be "out" of the processing system entirely.

The problem is that Corballis (2007a) is extrapolating the recursive character of the actual parsing operation (i.e., the actual computations the parser carries out) from the correct processing of hierarchical structures, which blurs once more the structures-mechanisms distinction. As noted, non-recursive mechanisms are in fact capable of processing recursive structures; indeed, the correct processing of hierarchical structures does not even mean *hierarchical*, processing building, let alone recursive, processing building. In other words, when Corballis speaks of "recursive parsing", what he is in fact speaking of is the processing of recursive structures, *not* the parsing operations themselves.

2.4 Via Via

According to this chapter, generative systems in general, and that of the language faculty in particular, are underlain by a general and global recursive property: finitary inductive definitions. On a related note, the expressive power of natural language is mildly context-sensitive, and there has been a convergence of different theoretical formalisms that precisely exhibit this type of output (Stabler, 2011). Combining these two points, it is safe to state that some of the differing formalisms proposed for linguistic knowledge —namely, *Minimalist Grammars*, *Tree-Adjoining Grammars*, *Categorial Combinatory Grammars*, et alia— also converge on the locus of recursion within their frameworks, despite misleading appearances. It is also rather clear that the myriad articles on recursive structures and mechanisms, their conflation and other issues are a red herring —at least regarding basic properties of the computational system for language. Still, it is worth discussing how the latter relate to issues such as "universality" claims and non-linguistic cognition, but I will devote the last chapter to that. Finally, it should be pointed out that this convergence of grammars pertains to expressive power only, and therefore, to weak generation (string-generation). Linguists, however, specifically focus on structure-generation, a formalisation of which remains beyond current understanding. One reason for this, Frank (2004) argues, is that different formalisms effect different mapping functions, which makes it hard to compare them in structural terms. Be that as it may, the next chapter continues with the three-stage explanatory strategy delineated in section 1.1, as it provides a detailed analysis of the mapping function at the heart of the *minimalist program*: the abstract implementations *merge* effects.

Chapter 3

The derivations into the interfaces

Up to this point, I have defended the necessity of postulating a computational system at the heart of the language capacity, as the sheer number of sentences we can produce and understand surely exceeds the storage capacities of human memory. That being the case, though, there is a question surely lurking underneath: must sentences actually be constructed at all? Would it not be possible to account for our linguistic knowledge simply in terms of precise combinations of "structural templates"?

The latter possibility has been explored in one guise or another in a variety of approaches, such as *construction grammar* (Goldberg, 2005), *head-driven phrase structure grammar* (HPSG; Sag, Wasow & Bender 2003) and *lexical functional grammar* (LFG; Bresnan 2001). According to the first framework, an attempt should made to draw up the 'conventionalized pairings of form and function' (ibid., p. 3) that diverse languages manifest. Whilst constructionists readily admit that there must be some way to combine these templates into novel structures (p. 4), they nonetheless claim that 'networks of constructions' capture our knowledge of language '*in toto*, i.e. it's constructions all the way down' (p. 18; emphasis is Goldberg's, bold font eliminated). Plausibly, this would appear to be a rather heavy burden on a theory of language, for there seems to be at first sight a great cross-linguistic diversity of structures. Even if a finite collection of templates is to be provided for a specific language, a thoroughgoing linguistic theory must also accommodate the fact that a child can acquire any language he/she is exposed to, and must thereby be endowed with the appropriate mental armamentarium. Thus, a constructionist would have to postulate a finite set of underlying templates whose combination results in the right structures given the input the child receives. Similar concerns have driven the practitioners of the other two approaches (HPSG and LFG), but they have offered a subtle take on issues related to the subject-matter of linguistics, which is worth discussing in some detail.

Whilst both HPSG and LFG employ rewriting rules in order to provide a formal characterisation of linguistic structure, these rules are not taken to implicate actual derivations. Rather, rewriting rules merely *define* syntactic trees (Bresnan, 2001, p. 44) in a 'formally precise way' (Sag et al., 2003, p. 525). The latter, moreover, make much of the fact that their approach is, apparently, direction and process neutral (p. 35), by which they mean that their theory does not describe the direction of a derivation or the manner in which the structures are produced or understood in processing. This point, however, seems to me to be clearly exaggerated. First of all, these scholars need to show what repercussions, if any, the postulated direction of a derivation has for a theory of competence, and whether this has any consequence for a theory of performance. Regarding the second point, Chomsky has repeatedly insisted, at least since his 1965 book, that a derivational theory of grammar is indeed process neutral, and there is no need to add anything in that respect. As for the first point, I do think there *is* a genuine point to resolve in relation to the direction of a derivation, and I will devote the next section to clarify precisely what the issues are.

Be that as it may, though, it is my impression that the charges Sag et al. (2003) levy at theorists working within *minimalism* —a derivational theory— do not quite land. For a start, it is seldom the case that clear examples of the mistakes they denounce are actually identified. More importantly, potentially substantive consequences stemming from particular theoretical choices are rarely if ever spelled out. I will discuss some of the relevant issues in the next section, and as I will show, Sag et al. ascribe views to the practitioners of minimalist syntax that for the most part they do not seem to hold at all —among others, the literalness of "movement" operations (see ibid. pp. 303 et seq. for details). In my view, this reflects a (mis)perception of the critic rather than an accurate description of the mind of the minimalist grammarian.[1]

Moreover, the manner in which both Bresnan (2001) and Sag et al. (2003) describe what exactly a grammar is supposed to characterise cannot be regarded, in my opinion, as an account of competence as understood in Chomsky (1965); a fortiori, it cannot be seen as a clearer understanding of what competence is supposed to explain. It must, instead, be regarded 'an epistemologically distinctive conception', as Bresnan (2001, p. 89) puts it. It is, after all, a historical fact that competence *was* defined in terms of derivations from its very inception, a characterisation that greatly influenced Marr's conceptualisation of his computational level of analysis. Indeed, Marr (1982) perceptively noted that linguistic theory was

[1]The same applies to a criticism of generative grammar that is repeatedly made in Goldberg (2005); namely, that generative theories 'derive one construction from another' (p. 10; see also pages 19 and 23). This may well have been the case in early generative grammar, but it cannot be predicated of current *minimalist* approaches; or at least not in the sense that Goldberg has in mind. That is, whilst early generative grammar did derive passive sentences from their active counterparts, or questions from declarations, *minimalism* derives all sorts of structures from a set of underlying forms, a perspective that is shared by formalisms such as *tree-adjoining grammar* (TAG) in the form of "elementary trees" (see Frank 2004 for a brief description). A slightly different taken is offered by McNeill (1975), for whom relating structures to each other is the very thing linguistics ought to be doing, a practice that goes back to Ancient Greece, according to him.

'defined by transformations' —what we would call today derivations—, 'which look like computations' (p. 28). Given that rewriting rules merely replace some strings for others, while transformations were meant to be meaning-preserving mapping functions —and structure-preservation is surely the hallmark of what a computation is—, the connection Marr made is unsurprising.

All in all, it seems to me that providing a "procedural" account of linguistic knowledge is a much more cogent solution. As Pylyshyn (1973) argues, such a standpoint is commonplace in computer science, and there is no reason for this viewpoint not to apply in cognitive studies too. Rather fittingly, Pylyshyn discusses the procedural knowledge underlying the infinitude of the natural numbers, which maintains the analogy I drew above between the generation of linguistic structures and the generation of natural numbers. As he puts it, a computer scientist would not bother to endow a machine with a list specifying that such or such integer belongs to the set of the natural numbers; instead, knowledge of a recursive definition of the successor function would suffice. In short, to know a recursive definition like the successor function is to know a procedure that can generate the natural numbers, and I here assume that the same holds for linguistic cognition.[2]

Some linguists hold a stronger view on these issues. In the introduction to a collection of papers on minimalist syntax (S. D. Epstein & Seely, 2002), its editors claim that in order to explain a given phenomenon, one must be able to "build" it, a job they see befitting the derivations linguists propose, perhaps connecting linguistic methodology to that of the natural sciences. There is perhaps some truth to this, but I would not want to take this methodological point too far. Still, it *is* the case that we must account for the manner in which words combine into complex structures, and a procedural explanation is no more than a characterisation of how the theory of the computation pans out. In a much more neutral manner, Chomsky (1995a, pp. 391–2) draws an analogy between a derivational account of syntactic structure and the manner in which phonology employs cyclic rule applications in explaining phonological phenomena; again, a procedural view of things.

In the case of the derivations that will engage me here, some of the components that are relevant include the set of lexical items (and their internal structure), the computational system and the conditions imposed by the sensorimotor (SM) and conceptual/intentional (C/I) interfaces. Naturally, a detailed analysis of the mapping from lexical items to sound/meaning pairs would require a full-length book, and I do not aim to undertake an exhaustive study here. Rather, my concerns are

[2]In a way, adopting a procedural approach entails favouring a theoretical account that explicitly posits a *theory of the computation*. Whilst there is some truth to that (at least in my case), the framework adopted here —viz., the derivational theory usually associated with linguistic *minimalism*— is merely the natural selection given the subject matter of this chapter; that is, an analysis of the properties underlying linguistic computation *qua* abstract implementation. At the same time, and while there is certainly a real issue at hand with regards to whether we should favour a derivational or a representational approach (see Brody 2002 for a relevant discussion), my choice here does not reflect the belief that a representational account does not also provide a *theory of the computation*; cf. Brody's statement that his pure representational theory *generates* structure in the mathematical sense of generation (2002, p. 22); Steedman (2000, p. 5) voices a similar point.

much narrower in scope. First of all, I will be solely concerned with the computations of "narrow syntax", that is, the component that maps structures to the two interfaces. Consequently, I will not discuss the computations carried out by the phonological component at the SM interface and by the semantic component at the C/I interface —two computational processes that, according to Chomsky (2008), proceed in parallel to narrow syntax. Instead, I will be focusing on the possible recursive character of the derivations *merge* effects in narrow syntax and the full generative power of the language faculty.

Regarding the first of these two broad questions, it may be contested that in the previous chapter I have already established the recursiveness of *merge*, rendering this point moot and in no need of further elaboration. Actually, what I did in the previous chapter was to show that *merge* generates syntactic objects recursively —this is (an aspect of) what it does— which is not quite the same as the issue of how it proceeds in a derivational process. The recursive character of a derivation would be a fact about its "shape" and it certainly has no obvious direct connection to *recursive generation* (as defined in pages 2.1 and 2.1 supra).

Before I proceed, an(other) inconsequential aspect of the first question needs to be clarified. This can be swiftly done, as it merely requires correcting some not quite correct definitions from the literature. This is the case in Di Sciullo & Isac (2008) and Hornstein (2009) (but there are many more), given that they define the recursiveness of derivations in terms of how *merge* applies over its own output. As explained in chapter 1, that would be a definition of recurrence tout court, and whilst it is (trivially) true that a recursive operation does just that, such a property does not differentiate recursion from iteration (recall my discussion of the "iterative conception of set" above). Rather, a recursive process results when an operation calls itself, and that usually translates into an application of *merge* over a *subset* of its own output (chains of deferred operations would naturally follow). Whether this applies for linguistic derivations is one of the issues to be explored below.

3.1 The Structure of Derivations

A rather noticeable difference between the derivations of old-style rule systems and those of current-day minimalism involves their "direction". In the 1950s and 60s, a derivation would start with an S (for sentence) rule, which would rewrite the non-terminal S symbol into, as it might be, an NP (noun phrase) and a VP (verb phrase) —other non-terminal symbols. Subsequent rule applications would rewrite these symbols into others until reaching terminal symbols (such as N for noun, V for verb, ecc.) which would simply be substituted by words. Further, a tree structure would be associated to the whole rule-application process; in this sense, the tree structure would be said to codify the history of the derivation. In short, the whole process starts by assuming that there is a sentence —that is, the derivation starts from the "top"— and it then proceeds to "expand" all non-terminal symbols downwards until all rules have been applied. In order to illustrate, I include below a set of rules and a tree structure for the sentence *the candidate has prepared a speech about the health service*.[3]

(3.1) (a) $S \rightarrow NP\ VP$
(b) $NP \rightarrow D\ N$
(c) $VP \rightarrow V\ NP$
(d) $NP \rightarrow NP\ PP$
(e) $PP \rightarrow D\ NP$
(f) $NP \rightarrow D\ AdjP$
(g) $AdjP \rightarrow Adj\ NP$

(3.2) (a) $N \rightarrow$ *candidate speech service*
(b) $D \rightarrow$ *the a*
(c) $V \rightarrow$ *has prepared*
(d) $P \rightarrow$ *about*
(e) $Adj \rightarrow$ *health*

According to a minimalist analysis, however, *merge* starts a derivation by selecting what at first sight would look like the most embedded elements of a fully-specified structure.[4] To keep with the structure in Fig 3.1, the first items to enter the derivation would be *health* and *service*, and from that point on *merge* proceeds "upwards" until the whole structure is completed. The old-style derivations, then, were "top-down" affairs, while *merge* proceeds in a "bottom-up" fashion. Naturally, talk of top-down and bottom-up derivations is clearly metaphorical, but it does nonetheless illustrate the main topic of this chapter: the shape of a computational process.[5]

[3]It is customary to use syntactic trees to represent the underlying structure of a sentence, but this is no more than a useful graphical format. The underlying structure of a sentence can be alternatively described with brackets: [The candidate [has prepared [a speech [about [the [health service]]]]]]. Further, I should point out that current linguistic theory analyses NPs as being part of a larger DP structure, but I will keep using the label NP here. Nothing hinges on this particular choice for the purposes of this thesis, but it does need to be kept in mind, as I may refer to other works that use the DP label instead.

[4]This is not quite true from a historical point of view. Chomsky (1971), a clearly pre-minimalist work, was already postulating that syntactic operations 'apply in a cyclic fashion, first to the most deeply embedded structures, then to structures that contain them, and so on' (p. 90).

[5]In drawing a parallel between the derivations of rewriting rules and *merge*, I have disregarded a period of generative grammar in which derivations were not so central. In the 1970s and 80s, so-called *government and binding* theory (which eventually mutated into the principles-and-parameters approach) focused on the structural constraints that explained why some structures and not others are permissible, and while some computational-like operations were kept (such as *move*-α), the main focus of this perspective laid on the structural information codified in the lexicon, such as X-bar theory (roughly speaking, the Specifier-Head-Complement(s) structures I described in the previous chapter). Still, some *government and binding* theory properties will be of relevance soon enough, as they can be shown to relate to the bottom-up nature of *merge*. It could also be argued that the

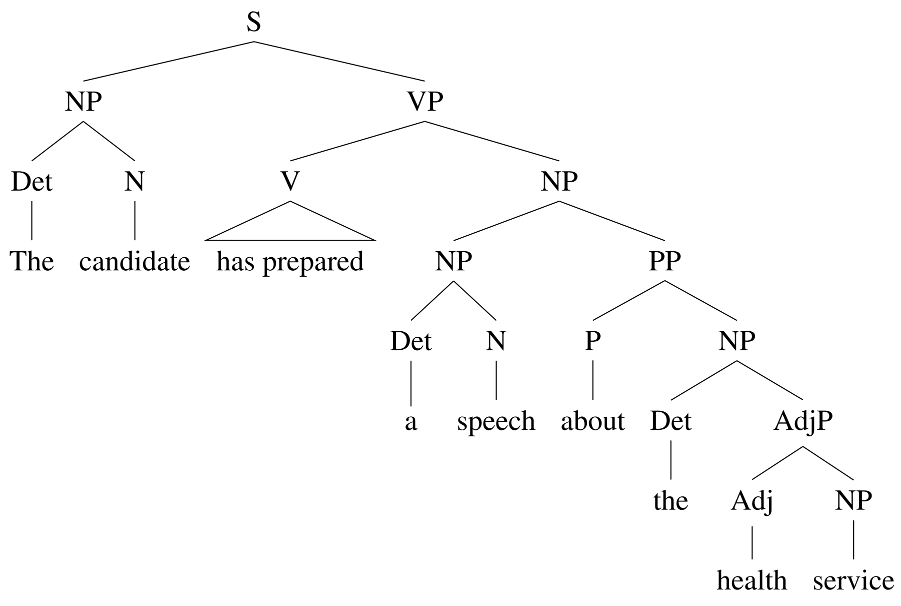

Figure 3.1: A diagram of ‘the candidate has prepared a speech about the health service’

It is worth noting that a distinguishable feature of a bottom-up derivation is the centrality that the argument structure of a sentence (i.e., its proposition) plays in the overall process.[6] Indeed, constructing the underlying argument structure of a sentence has been postulated as being the first complete set of operations *merge* carries out in the course of a derivation. By a set of complete operations, it is meant that *merge* proceeds cyclically, or in *phases*, in the parlance of Chomsky (2008). A derivation, then, is a succession of stages (phases) that is brought to an end once a sound-meaning pair has been formed. In accordance with the literature, call the representation that is sent to the SM interface a PHONological representation, and the representation fed to the C/I interface a SEMantic one. The role of *merge*, then, is to construct PHON-SEM pairs by operating on the syntactic features of lexical items (in this sense, lexical features drive the derivations). The representations PHON and SEM, though, cannot be accessed (that is, they participate in the computations of the phonological and semantic components, respectively) until narrow

criticisms of Sag et al. mentioned above relate to now-abandoned features of *government and binding* theory, rather than to *minimalism* itself. In what follows, I will offer a rather sketchy description of *merge*-conducted derivations in which much data and many properties will not even be mentioned. This, however, will not affect the main point of this chapter.

[6]A very positive characteristic of merge-based analyses, in my opinion, given the close connection between propositions and language. Plausibly, language is either the medium in which propositions can be expressed or the unique format in which propositions can at all be represented.

syntax has completed a phase —in this sense, the completion of a phase permits the *transfer* of material to the two interfaces. I now turn to a closer analysis of the basic principles and properties of derivations.[7]

3.1.1 The nature of lexical items

Lexical items are complexes of syntactic, phonological and (maybe) semantic features.[8] In order to there being a derivation at all, a finite subset of lexical items is gathered from the whole set of lexical items that must surely be stored in long-term memory; this is usually called a *numeration* or *lexical array*, which can be exemplified thus: $N = \{(\alpha_1, i), (\alpha_2, i), (\alpha_3, i), (\alpha_4, i)\}$. Each α stands for a lexical item (LI), and the integer indicates the number of times each LI is entered into a derivation.[9] *Merge* operates by taking pairs of syntactic objects (SOs; as defined in chapter 2, ft. 18) and maps them onto a new SO;[10] that is, *merge* (α_1, α_2)= $\{\alpha_1, \alpha_2\}$, in the usual notation of set theory. A derivation is nothing more than the construction of a single SO from a list of LIs. Tomalin (2007, p. 1794) offers a schematic description of the process:

(3.3) **Numeration:** $\sum = \{(\alpha_1, i), (\alpha_2, i), (\alpha_3, i), (\alpha_4, i)\}$
Step 1: merge (α_1, α_2), resulting in $\{K_1, \alpha_3, \alpha_4\}$
Step 2: merge (K_1, α_3), resulting in $\{K_2, \alpha_4\}$
Step 3: merge (K_2, α_4), resulting in $\{K_3\}$

For the sake of simplicity, let us assume that every LI enters the derivation just once. An operation —*select*— takes two elements from the *numeration* (N) —in this case, α_1 and α_2— and places them in a *workspace*. *Merge* takes these two items and combines them into a new syntactic object, K_1, the unordered set $\{\alpha_1$,

[7]To some extent, I will be following the formalisation of minimalist syntax laid out in C. Collins & Stabler (2011); C&S here and after.

[8]I say "maybe" because it is not at all clear that there are any semantic features, as forcefully argued by Fodor (1998). In my view, lexical items contain syntactic and phonological features, and they are linked up to specific *concepts* somehow. Thus, the connection between lexical items and concepts —whatever that turns out to be precisely— constitutes the locus for whatever semantic computation there might be.

[9]Note, though, that α is merely a tag for the underlying features. Even though I will be employing labels such as *candidate* below, these should not be confused with the actual words they are associated with. Thus, when it is stated that *candidate* is introduced into the derivation, this merely means that its syntactically-relevant features have been so entered, not the actual word. In fact, according to the *distributed morphology* approach (Halle & Maratz, 1993), vocabulary insertion takes place *after* narrow syntax has finished its operations.

[10]*Merge* is taken to be a binary operation on account of Kayne's (1981) unambiguous paths. Note, however, that Kayne was therein preoccupied with the best way to account for the correct establishment of binding and government relations among syntactic nodes. He put forward a solution involving the postulation of unambiguous upward "travels" from one node to another, which he achieved by imposing a binary branching condition. Whilst this unambiguous path travelling has now been dispensed with, the binary branching condition must be kept, and hence binary *merge*.

$\alpha_2\}$. Despite being unordered, one of these two SOs projects, thereby providing a label for the overall SO; that is, *merge* returns a $\{\gamma\ \{\alpha_1,\ \alpha_2\}\}$ set (where γ is the label of the SO), which might as well be a syntactic node such as an VP. That is, *merge* 'embeds (an object) within some construction. . . already formed' (Chomsky, 1995b, p. 248); this is the most important aspect of what *merge* does.[11] Subsequent applications keep building up the tree (and thereby creating new nodes) by adding new structure to the root node until every element from the *numeration* has been employed (that is, merged). Every stage of the derivation can then be described as being composed of a numeration (N) and a *workspace* (W).

If α_1 and α_2 are both part of W, then they are combined by so-called external *merge*. This would be the case if either *select* takes two elements from N and places them in W or if there is already an SO in W and *select* brings another LI from N into W. However, if α_1 is part of W and contains α_2, α_1 and α_2 would be manipulated by internal *merge*. That is, α_2 would "move" from its current position and be copied onto newly-created structure. In this sense, *select* would prima facie play no role in such computations, as *merge* would be triggered to operate on the SO so far constructed without the need for the introduction of any new elements.[12] These factors point to rather fundamental properties of the overall process, such as: What determines the selections of *select*? Why are some elements introduced into the derivation before others? What precisely does it mean for a LI to "move" from its current position? Does the latter presuppose that the position where an element is to be copied onto already exists?

Given that derivations are driven by lexical features, it is natural to suppose that some of the answers to these questions ought to be found in the very constitution of lexical items. Stabler (2011), for instance, distinguishes four different types of features: categorial (such as *D*, *N*, *V*, *P* ecc. for determiner, noun, verb and preposition, respectively); "selector" features, which specify the categories that some lexical items require (for instance, a verb selects a determiner phrase (DP) as a complement, which Stabler symbolises as a *=DP* feature of the verb); "goal" features that require licensing, such as *-focus*, *-case*, ecc.; and the "probe" features of those heads that do the licensing, marked as *+focus*, *+case*, ecc. In these terms, a derivation would quite simply begin when the relevant "selector" feature is satisfied, and in minimalism this translates into a process in which the first stage involves the verb selecting its complement —call this stage the VP—, a complex that is then merged with the subject of the whole sentence —call the latter stage a vP.[13] At this point, the derivation is not complete, as a number of features remain

[11]This is *merge* at its simplest, sometimes referred to as *set-merge*.

[12]Cf. Brody (2002); for this author, copying/movement is just a case of selecting the same LI from the lexicon twice (note 6, p. 36).

[13]Note that in current minimalism, the entire verbal phrase is described in terms of outer and inner shelves —i.e., a "split" VP structure— with the subject initially generated within the outer vP shelf. There are good reasons for such an analysis, but they are not of relevance for our discussion (see Radford 2004 for an introductory description). Since my interest here has to do with the "shape" of derivations, it is a moot point what we call the actual stages or where precisely the subject is initially

to be licensed. The verb, for instance, must become tensed and in order to do so it must have its *tense* feature licensed by the relevant functional projection,[14] call this stage of the derivation the TP (for tense phrase). Further, the so-called *extended projection principle*, or EPP, specifies that every sentence requires its subject position to be filled, and this involves the NP in vP to check its *epp* feature in the specifier position of the TP. The final stage, the CP (for complementiser phrase), involves all movement to do with topicalisation and focus, and replaces the S node of phrase structure grammars. Note, then, that in very general terms a derivation is nothing more than a sequence of feature-satisfaction that proceeds, roughly, thus: V-v-T-C.[15]

In perhaps a more general manner, C. Collins & Stabler (2011) identify three types of what they call the "triggering" features that kick-start the whole derivation: subcategorisation features, which specify the arguments the main verb requires; the EPP feature; and a "movement" feature that attracts material to the specifier position of the CP (see ft. 21 infra). These three properties are said to trigger *merge*, and definitions for the main projections of X-bar theory —i.e., specifier, head and complement— can be built from them, as C. Collins & Stabler (2011, pp. 20–2) show. The latter is a rather important point, as it relates a classic property of *government and binding* theory with the bottom-up derivations; that is, the derivation proceeds in a bottom-up manner because of the order in which these triggering features apply (subcategorisation $\rightarrow$ *epp* feature $\rightarrow$ movement to SpecCP). Selection involves head-complement configurations, while licensing and movement establish specifier-head relations.[16]

In an attempt to unify all these properties, Chomsky (2001a, 2008) postulates an *edge* feature that, within a derivation, makes certain LIs "probes" in search of structural "goals", and the latter includes *subcategorisation* requirements and the *epp* principle. In this sense, the *edge* feature would be *the* property that drives the computation. To illustrate, imagine that α_1 and α_2 are merged, an eventuality that would be triggered by one of these LIs selecting the other. The LI that does the selecting would be the head of the resulting XP, therefore carrying the *edge* feature. A further step of the derivation would merge this XP with an LI from N if this newly-entered LI selects the head of the XP; for instance, a verb selecting a DP, where the verb is the new LI and the determiner the head of the previously-constructed XP.

generated.

[14]A distinction is usually drawn in linguistics between categorial phrases such as NPs and VPs and functional phrases such as a TP or a CP.

[15]Note, though, that in the bottom-up perspective I am outlining the actual final structure is more appropriately described as [C-[T-[v-[V]]]]. The progression in the main text merely indicates the temporality of stages in a left-to-right manner. Additionally, according to Chomsky (2008) a derivation is composed of two phases: vP (which specifies the argument structure) and CP (which codifies discourse-related functions.

[16]The usual choice of words in the literature is of features being "checked" or "valued", but I am playing fast and loose with the terminology here. I am also simplifying a great deal; for a description of lexical features and their role in derivational accounts of linguistic structure, see Adger & Svenonius (2011).

The result would be an SO in which the last LI to enter the derivation heads the overall structure and the previous SO —viz., its complement, the XP— would at this point be transferred to the interfaces for interpretation, rendering it inaccessible to further operations within narrow syntax. Subsequent applications of *merge* can only operate over the remaining LI in combination with another LI from N (recall that *merge* is binary) in exactly the same manner as before. That is, what Chomsky is proposing here is that only the edge of a SO is visible to *merge*, making *merge* an asymmetric mechanism; in consequence, it is this *edge* feature that is postulated to explain the fact that SOs can be further expanded —a property referred to as the *extension* condition (I will come back to this property below).

An analysis of derivations in terms of an *edge* feature has the virtue of clarifying an issue to do with movement. As discussed in section 1.3, movement is the result of displacement, a phenomenon in which a lexical item or phrase appears in a different position from where it receives interpretation. In a simple sentence such as *the car I bought*, while the phrase *the car* appears at the very beginning, it is in fact interpreted as the object of *bought*. An account within minimalist syntax postulates that this phrase is initially constructed as the complement of the verb and then copied/moved to its surface position. Narrow syntax and the semantic component compute both copies, but the phonological component is supposed to feed the SM interface only one of them; namely, the last generated copy.[17] At first sight, such a description may be taken to suggest that the SO undertaking movement eventually lands at a position that is already there —that is, previously constructed. In past work, in fact, I mistakenly concluded that such a perspective provided a glimmer of a recursive sub-routine within syntactic derivations (see Lobina & García-Albea 2009 for details). That is, I therein adopted an interpretation according to which it is sometimes the case that external *merge* creates SOs containing features that will result in the movement/copying of one of their internal elements, and this I took as a future operation that is being put on hold while further structure is constructed, therefore indicating a deferred operation —the hallmark of a recursive process. As a result, I specifically linked recursion to the operations of internal *merge*, such a connection appearing to be explicitly postulated by various scholars.[18] In actual fact, these scholars had something slightly different in mind and there really were no grounds to conclude that any deferred operations existed within syntactic derivations. To begin with, it is not the case that a SO is moved to a position of a syntactic tree that is already there before the derivation starts. A derivation does not consist in placing all LIs in N in their right configurational positions as if there was a template to fill. Rather, the *edge* feature and the *extension* condition conspire so that *merge* keeps building an intricately structured SO, but the resulting complex is the sole result of the underlying features at play. Thus, the "movement" of a LI is

[17] Recall that I offered a modified view of this point in section 1.3.

[18] Namely, S. D. Epstein & Hornstein (2000) mention that within the minimalist program recursion is 'relegated to the transformational (i.e., movement) component' (p. xii), while Soschen (2008) states that a 'relation between individuals may constitute a phase and induce movement (recursion)' (p. 212).

(obviously) not literal; all there is is continuous structure building that sometimes involves the copying of internal SOs.[19]

In order to illustrate, consider the sample derivation of the simple sentence *John likes the dog* (adapted from Hornstein 2009, p. 54):

(3.4) **(a)** Merge *the* and *dog* $\longrightarrow$ {the, dog}

(b) Merge *likes* and {the, dog} $\longrightarrow$ {likes {the, dog}}

(c) Merge *John* and {likes {the, dog}} $\longrightarrow$ {John {likes {the, dog}}}

(d) Merge T(ense) and {John {likes {the, dog}}} $\longrightarrow$ {T {John {likes {the, dog}}}}

(e) Copy *John* and Merge it and {T {John {likes {the, dog}}}} $\longrightarrow$ {John {T {John {likes {the, dog}}}}}

The graphic above does not codify any information regarding the features that drive the merging of LIs, but these are assumed to be implicated in the process somehow. My present point is merely that all *merge* does is create structure anew from a given numeration, and there is certainly no pre-emptive plan to follow. That is, there are no structural templates to fill. The fact that regular patterns appear to emerge out of this ensemble of properties —in particular, the general Specifier-Head-Complement(s) format— is, in my opinion, entirely accidental (see infra, though).

Note that the derivation in (3.4) also assumes that the right LIs are introduced into the derivation at the right time, a state of affairs that is accounted for in Stabler (2011) and C. Collins & Stabler (2011) (and to a certain extent in Chomsky 2008 too) by definition; that is, the start and the successive development of a derivation are established by the manner in which lexical items are characterised. Di Sciullo & Isac (2008) contest that this is an unreasonable proposal, as a more principled solution should be provided. Their own proposal involves characterising *merge* as an asymmetric operation that applies over two LIs when their sets of features are in a proper inclusion relation (p. 261).[20] That is, α_1 and α_2 would only be merged if the set of features of, say α_1, is a proper subset of the set of features of α_2. Naturally, the whole approach can only succeed if the sets of features are appropriately defined. To this end, Di Sciullo & Isac (2008) draw, firstly, a broad distinction between interpretable (roughly, Stabler's categorial features; see supra) and uninterpretable features (loosely, Stabler's "selector" features, see supra again;

[19]The edge-feature/extension-condition duo can perhaps be combined with Stroik's (2009) *survive* principle. According to the latter, unchecked/unvalued features are carried along within every application of *merge*, and so as further structure is being constructed, the unvalued features are pari passu checked against it. In this sense, a derivation carries on until all features have been *satisfied*. That is, derivations manifest *delayed* operations, not *deferred* operations. The *survive* principle of Stroik's is embedded within the *satisfy* architecture proposed by Frampton & Gutmann (1999), but I will not say more about it here.

[20]Note that this is a different type of asymmetry in the operations of *merge* from that postulated by Chomsky (see supra).

the latter are written *uDP* by Di Sciullo & Isac). Further, they draw a narrower distinction between categorial and operator features, the latter being those features understood to relate to discourse-related elements, such as the features *wh*, *topic* and *focus*.[21] The last distinction underlies, they claim, the dichotomy between internal and external *merge* in the sense that categorial features trigger the latter and operator features the former.[22]

Di Sciullo & Isac (2008) furthermore assume that the first step of the derivation involves selecting from N the LI that only contains interpretable features, which always means that a noun will be selected first, as it only contains a categorial *N* feature.[23] In the second step, the selected noun would be merged with that LI in N with which its features enter a proper inclusion relation; namely, the noun is merged with a determiner bearing features *D* and *uN*, forming a DP structure as a result.[24] In turn, the DP is selected by a verb carrying a *uDP* feature, *e così via*.

So long as the N continues to provide LIs that can enter in a proper inclusion relation with a given SO in W, external *merge* keeps constructing an ever more complex object. When no element in N meets this condition, then the *search* procedure applies within the SO itself, resulting in an application of internal *merge*. That is, by default *search* looks in N in order to establish a set inclusion relation with the SO in W; when it fails to do so, it attempts to establish such a relation between the head of the current SO in W and one of its internal elements.[25]

The process seems at first sight simple enough, but I think that in comparison with the proposals sketched above, it actually involves a significant amount of complexity. First of all, the successful outcome of a derivation involves much more than getting the characterisation of lexical items right. Indeed, some of the extra theoretical features Di Sciullo & Isac are forced to introduce include: sub-arrays within N so that *merge* successively focuses on the right (sub)group of LIs; the supposition that DPs are in fact constructed in parallel within the W; and a corollary of the latter, the further postulation that specifiers and adjuncts are con-

[21] The *wh* feature refers to the so-called wh-movement that takes place in the formation of questions. For example, in a simple question such as *what do you know*, the wh-word *what* is initially generated as the complement of *know*, since it is therein that it is interpreted, but is subsequently copied to SpecCP, where it is pronounced.

[22]Moreover, Di Sciullo & Isac (2008) do away with features that have usually played a role in narrow syntax, such as Case (a property that indicates the grammatical function of nouns and pronouns) and phi-features (the gender, number and person features of the verb), which they consider to be computed in a different space (p. 269). I will ignore Case here, but I will come back to phi-features below.

[23]I am brushing aside the distinction between subject and object nouns for the time being. It is also important to note that given that all applications of *merge* are binary and strictly speaking form sets, the very first step would actually involve merging the selected noun with an empty set; cf. Soschen (2008).

[24]Establishing set inclusion relations must necessarily require some sort of *search* procedure, and Di Sciullo & Isac (2008) postulate that *merge* is in actual fact composed of *select* and *search* sub-operations.

[25]To be more precise, *search* can only look into the SO within a phase, as any material transferred after a phase would not be accessible.

structed separately and introduced whole into the (main?) derivation. Clearly, the proposal to have specifiers and DPs constructed separately solves the issue of what kick-starts the derivation —the verb's complement is the only LI with only interpretable features— but at what cost?[26] Having parallel derivations of DPs and specifiers/adjuncts is clearly a more complex architecture than one that simply postulates that first-merge builds the SO verb-complement, second-merge adds the specifier, ecc.[27] Still, the idea of characterising the derivational process as one involving proper inclusion relations among lexical features is an interesting one, and further research will tell whether it can be empirically supported.

3.1.2 The internal structure of *merge*

At this point, though, I would like to discuss whether it is necessary to defend a decomposition of *merge* into sub-operations, a perspective that can be construed in a substantive manner (as in Boeckx 2009b, Hornstein 2009 and Hornstein & Pietroski 2009; let us call this view BHHP) or manifest itself as no more than a passing comment (as in Di Sciullo & Isac 2008 supra). According to BHHP, *merge* qua combinatory operation ought to be divided into two sub-operations: *concatenation* and *label*. A *concatenating* operation, we are told, is a domain-general mechanism that brings together two objects, and is presumed to be operative in many domains. Plausibly, some sort of concatenation is necessary in language, as "bringing things together" is the very least that syntax does. In addition, linguistic theory needs to account for the fact that syntactic structures are endocentric, which is to say that every phrase is headed by one of its internal elements; to wit, a noun phrase is headed by a noun, a verb phrase by a verb, ecc. Given that this property does not appear to be present in the structures of other cognitive domains (see chapter 5 for some discussion, though), these scholars postulate an operation specific to language and whose task is to identify the head of a phrase in order to correctly *label* the resulting SO. Thus, *label* would be a unique feature of language, the bare necessary for many other properties, such as what I called in chapter 2 categorial recursive structures.

Considering that those working within the minimalist program strive to find ever simpler accounts, the BHHP proposal immediately strikes as more complex

[26]Note that if the LI that is to be the subject of the sentence was to be included in the same sub-array as the LI for the object position, the former would then have to be ascribed an *epp* feature so that its (eventual) role is distinct from the latter, but this is nothing but sheer postulation. Admittedly, this applies to the proposals sketched above too.

[27]To be more precise, external *set-merge* builds the argument structure, while internal *set-merge* constructs scopal and discourse-related properties. So-called *pair-merge* —introduced in Chomsky (2001a)— deals with adjuncts such as *health* in the phrase *the health service*; that is, *health* is imply adjoined to *service* instead of being merged according to either an *edge* feature or a proper set inclusion relation. In this sense, *pair-merge* would be responsible for generating specifiers and adjuncts. Langendoen (2003) puts forward another type of *merge* —*list-merge*— which represents argument structure by listing the predicate and its arguments in a sequence. He argues that this type of *merge* is better at building argument structure *and* specifiers, but I will not treat this matter here.

than the *minimalist* framework so far described. For a start, *concatenation* seems like a very costly operation to propose, given that it merely involves putting a number of objects next to each other; that is, it does not combine them into a compound, which is surely what syntax exclusively does (cf. the open-floor discussion that follows Boeckx 2009b).[28] Moreover, what evidence is there, after all, for a concatenating operation in narrow syntax? Naturally, BHHP would remonstrate that the SOs linguists study are the result of the combination of *concatenation* and *label*, but if we are to atomise *merge* into these two operations, we should be able to point to stages in a derivation where the effects of *concatenation* are evident. This, however, does not appear to bear out. To point to the existence of concatenation in other cognitive domains is not to offer a demonstration of its connivance in the language faculty. A fortiori, it is not clear that *concatenation* is so common in other cognitive domains; if anything, non-linguistic domains clearly exhibit complex structures that hint at a sort of merging operation as conceptualised by Chomsky (think of mental ascription abilities, for instance). If that is indeed the case, what needs to be postulated is a domain-general *merge* that in the case of language is augmented with some sort of mechanism that accounts for headedness. Thus, even if we were to be persuaded, BHHP might say, that *concatenation* is too weak an operation and at the same time too costly a primitive of the language faculty, is it not the case that adopting a *merging* mechanism still requires some sort of *labelling* operation?[29]

I think the last suggestion can also be rejected on the grounds that labelling structures takes the existence of syntactic phrases as conceptualised by linguists perhaps too literally. After all, it is one thing for a phrase to be endocentric, it is another matter completely for phrases to be distinguished in terms of the category that heads each one of them. To be sure, the linguist plays fast and loose with the terminology and the notation —and why not say it, with some of the most substantive claims too— but these are simply facts of conventional research that ought to be given significant leeway. The actual issue at hand is, I think, much more subtle than that. A strong point of the *edge* feature account is that it provides a very simple perspective regarding the ontology of linguistic categories. According to Chomsky (2008), all that is needed in order to explain headedness is a sort *minimal search* operation that "looks" into the structure of the newly-built SO as to identify the object that carries the *edge* feature, ipso facto establishing it as the head of the SO. In other words, the head of a structure projects its features to the *edge* of an SO, making it visible to further operations of narrow syntax. This, coupled with

[28]Interestingly, Roeper (2009) discusses and-conjunction structures in terms of concatenation and suggests that this sort of operation is not part of narrow syntax or core grammar at all; according to him, it is a phenomenon external to the language faculty altogether.

[29]Cf. C&S's take on this matter: 'there is a close relation between triggered Merge and labels: both are ways to indicate that Merge is asymmetric, and furthermore, the Label function is defined purely in terms of how features are checked. Given this close connection, it may be that one or the other is redundant' (p. 21). Immediately below I offer a reason for considering the *labelling* mechanism de trop.

the *extension* condition and some other computational principles that participate in structure building (to be treated immediately below), results in a computational process in which the different tag names linguists use (such as v, VP, TP, CP and many others) refer, or so I here claim, to the different stages of a derivation; that is, they are not *labels* for static, frozen-in-time syntactic phrases. In this sense, syntactic phrases per se do not exist; their labels are just a useful notation in order to outline how SOs are built (or to map their syntactic distributions).[30]

As for the decomposition of *merge* into *select* and *search* (Di Sciullo & Isac, 2008), it is not clear that these operations should be viewed as participating in an atomisation of *merge* at all. Rather, they seem to be accompanying mechanisms of *merge* rather than internal sub-operations. In support of this interpretation, it is of note that both *select* and *search* are explicitly employed in the aforementioned proposals of Stabler's, C&S's and Chomsky's, having no effect on the primitiveness of *merge* qua combinatory mechanism.

3.1.3 General computational properties

In the discussion so far provided, I have only focused on the features lexical items are composed of as the main driving force behind a derivation, but the literature has studied at least two other factors. One of them was already mentioned above —viz., the interface conditions— and I will come back to these (briefly) below. For now I want to draw my attention to what have come to be known as "third-factor" features of linguistic structure (see, e.g., Chomsky 2007b); namely, basic computational properties that are supposed to remain invariant across diverse computational systems. Some of these properties have already been alluded to above, but it is worth expanding the discussion of this specific issue.

To begin with, though, it must immediately be accepted that the study of the computational properties subsuming linguistic derivations should be treated in the same manner as the abstract implementations I discussed in section 1.1. That is, the very first thing that needs to be provided is a specification of the algorithm underlying the language faculty, and I believe this to have satisfactorily been done in the case of *merge*, both here and in the literature. Following upon this, we can study the implementation of *merge* qua computational process in an abstract and therefore functional manner by allocating cost units to its atomic steps —this can be done by employing a "size metric" and an "asymptotic notation", that is, by providing a *theory of the computation*.[31]

[30]This is not to deny the glaring fact that some syntactic operations target specific collections of lexical items (that is, phrases), as in question formation or extraction. In my view, however, such operations target specific stages of a derivation, and not structures per se.

[31]At the end of their paper, C&S assure us that 'it is not possible to define Merge' and 'its recursive application' in isolation (p. 30). This is only true for the construction of the *theory of the computation*, though, as *merge* qua algorithm (that is, as an abstract mathematical object) can certainly be characterised in isolation. In fact, I have here claimed that doing so is the very first step of an explanatory account for computational theories of faculties and modules.

Precisely because of the very possibility of providing a “computational level” explanation, one should be careful not to draw the conclusion that conjecturing operations such as *minimal search* introduces features that pertain to a different level of analysis, such as memory limitations and the like. On the contrary, an operation like *minimal search* ought to stand on the same ground as the “least search” operator of the partial recursive functions I described in section 1.1; that is, it is nothing more than a mapping function and its study can proceed in the manner there delineated.[32]

Having said that, though, it is fair to say that linguistics is very far from a satisfying analysis of the computational features at play within derivations. First of all, most of the “basic computational principles” under current discussion have noticeably been introduced in a rather loose and informal manner, which does little in the way of offering a precise formulation of the concomitant repercussions. On the other hand, the principles that have been proposed so far are so intuitive and practical that providing a clear definition for each one of them is in fact not very hard, and C&S furnish us with appropriate formulations by employing the terminology of set theory. The properties I want to focus on have for the most part been formulated by Chomsky in his (1995b; 2001a; 2004; 2007a; 2008) works —or at least his are the proposals that have received the most approval— and these include the *no-tampering condition* (NTC), the *extension condition* (EC), *inclusiveness* (IN), *local economy* (LC) and the *phase-impenetrability condition* (PIC).[33]

The NTC establishes that after the merging of a pair of SOs, these two objects are left unchanged. In the formulation of C&S, ‘for any two consecutive stages in a derivation…, for all $A \in W_1$, either $A \in W_2$, or there is some $C \in W_2$ and $A \in C$’ (p. 14).[34] According to C&S, what this says in plain English is that an SO in a given W must find a place in the next stage of the derivation; in other words, no element of a specific W can be destroyed (ibid.).

At the same time, and as stated earlier, the EC obligates *merge* to keep extending the structure of the SO in W, and this is rendered by C&S as stating that if a stage of the computation is derived by *merge* from an immediately previous stage, ‘there is some $A \in W_1$ and $C \in W_2$ such that… [i.] $C \notin W_1$ (C is created by Merge)… [ii.] $A \notin W_2$ (A is extended)… [iii.] $A \in C$ (A is extended to form C)’. In plain English, this just means that A in a certain W is extended to C in a successive stage (ibid.).

The IN, on the other hand, specifies that each stage of the derivation is exclus-

[32]Both Johnson & Lappin (1997) and Uriagereka (2008) appear to hold the opposite view. The former believe that properties such as economy, complexity and the like pertain to engineering —a standpoint that would have to brush aside the entire history of computability theory in order to stand— while the latter is particularly prone to link memory resources to specific applications of *merge* within his, in my opinion misbegotten, “reinterpretation” of the Chomsky Hierarchy (see chapter 7, loc. cit.).

[33]LC was firstly introduced by C. Collins, as referenced in C&S, p. 17.

[34]A and C are SOs, the $\in$ symbol is the usual membership relation of set theory and the indices in W refer to different stages of a derivation; in the conception of C&S, each stage of the derivation is a combination of a specific lexical array (or N) and a specific W.

ively the result of properties within LIs —that is, of lexical features; or in other words, that no new features are introduced in the course of the derivation.[35] In the words of C&S, 'the only elements contained in W_n are the lexical tokens [from N] and sets containing them' (p. 16).

As for the other two computational properties, LC establishes that 'whether or not an operation… applies to elements of W_i… is determined completely by W_i and the syntactic objects it contains' (ibid., p. 17), while the PIC imposes that '[i]n phase α with head H that has [a] sister… Merge cannot apply to the sister of H or anything contained in the sister of H' (p. 26). These two principles, then, delineate the domains in which *merge* can and cannot apply,[36] and as a consequence they are closely related to a *phase* and the *edge* feature.[37]

Note that all these definitions are appropriate and clear, but they do not provide functional "cost units" for the atomic steps that a derivation is surely composed of; therefore, the *theory of the computation* so far constructed by linguists falls well short of what mathematical logic mandates a computational-level analysis ought to explicate.[38] One could perhaps protest that no more than what Chomsky and colleagues have put forward is in fact needed, but surely to focus on the intrinsically formal effects that computational principles produce is to say something about how they interact in a computational process qua a sequence of stages. It is more reasonable to believe, I think, that linguistics is now at a stage in which this sort of issues can barely be formulated, let alone approached in an entirely satisfactory manner. Still, it is no small consolation that C&S have been able to formulate the pertinent properties in a clear and concise way.[39]

[35]Chomsky (2001a) has indices, traces and bar levels in mind, and so the IN automatically eliminates any possible interpretation of syntactic structure as "being there" before the SO is constructed (I am particularly referring to bar levels here).

[36]Note that the five computational principles I have described relate to the applications of *merge*, and not to those of *select*, *minimal search* or *transfer*. Following C&S, *select* and *minimal search* apply in N and/or W, while *transfer* is clearly an interface operation.

[37]There are some other features of linguistic derivations that I am leaving out of my description, such as the operations *last resort* and *agree*. I will not say anything about the former here, but I will come back to the latter in the next section.

[38]Perhaps we would do well to stop short of proposing an analysis of the "computational complexity" of derivations, by which it is meant, following Aho, Hopcroft & Ullman (1974, p. 2), the determination of the 'rate of growth of the time and space required to solve larger and larger instances of a problem'. That is, it is not clear that such features are applicable to linguistic derivations. Cf., though, the relationship between computational complexity and specific grammatical formalisms, as discussed by Pratt-Hartmann (2010).

[39]In relation to something I briefly mentioned in section 1.1, C&S state in the first page of their manuscript that their aim in writing their paper was to advance a formalisation of *minimalist* syntax for the field so that new proposals could be formulated and evaluated —both conceptually and empirically. In addition to this, I believe that a more important aspect of their contribution is the role their paper can play in a future "foundation" for linguistics, as Dean and others understand this term (see ft. 27 supra). This is a topic of great importance and breadth, but one that I will only broach in the conclusion of this thesis.

3.1.4 Interface conditions

Regarding the conditions the interfaces impose, it will be recalled that in section 1.3 I described these impositions as being broadly speaking of two kinds. The SM interface requires a structure that can be linearised, and that means that the PHON representation that narrow syntax feeds to the phonological component must be such that the output of the latter results in an appropriate set of instructions for the articulatory/perceptual systems. The C/I interface, for its part, necessitates structures that exhibit hierarchical relations in terms of dependencies and containment, and it can readily be apperceived that the latter requisites are satisfied by the manner in which *merge* is constrained by lexical features and the basic computational principles described above.

Linearisation is admittedly a more complicated phenomenon to account for, and even though it has engaged the field a significant amount, a common perspective is yet to surface; or so it seems. Nevertheless, a rather influential view, building on Kayne (1994), is provided by Moro (2000). According to Moro, *merge* builds unordered symmetric structures (p. 22) of various kinds, such as multiple specifiers and head-head sequences (p. 32).[40] The C/I interface would have no problem interpreting such representations —unordered symmetric structures are still hierarchical—, but the SM interface, Moro contends, requires an input that represents order somehow (pp. 2–3), given that order is plausibly the sine qua non condition for anything to be linearised. As a solution to this conundrum, Moro proposes that the narrow syntax component makes asymmetric structures out of points of symmetry via movement operations in order to meet the condition imposed by the SM interface. In this sense, movement is a property of the geometry of phrase structure.[41] The type of structure that the SM interface receives constitutes a collection of [Specifier-[Head-Complement(s)]] phrases —the result of movement—, that can easily be linearised.[42]

[40]In a similar vein, Citko (2005) discusses a type of *merge* that also generates symmetries. She calls this operation *parallel merge*, which is posited to build multi-dominant structures; that is, structures in which a given LI can be concurrently dominated by two different nodes.

[41]Taking his heed from Moro, Richards (2010) puts forward a simplified explanation for these facts in terms of a *distinctness* principle according to which linearisation of α and β is only possible if these two elements are distinct. Building upon this, Richards reduces Case theory to a special manifestation of the *distinctness* condition, a perspective that might relate to the contention of Di Sciullo & Isac (2008) that Case theory is computed in another "space".

[42]According to Kayne (1994), precedence and order within Specifier-Head-Complement(s) structures can be established by computing the asymmetric c-command relations among syntactic nodes. Very roughly, *c-command* is a relationship between the nodes of a syntactic tree in terms of dominance, similar to the relation between the siblings and descendants of a family tree (the actual technical details are not important for the subject-matter of this thesis, though). It may be worth pointing out that Kayne argued that asymmetric structures were the norm for every level of syntactic representation; thus, Moro's modification involves allowing for the generation of symmetric structures at certain points of the derivation. A complete different take on linearisation is to be found in Kremers (2009). According to this author, all that is needed in order to achieve linearisation is a recursive algorithm that searches through the binary tree that narrow syntax sends over to the SM interface. Less roughly, Kremers is proposing a *depth-first* search algorithm that applies itself to each sub-node as follows: if

3.1.5 The actual shape of derivations

Now that I have described the main features of linguistic computations, the overall structure of the derivations the language faculty effects ought to be clearly perceivable. Note, then, that the applications *merge* carries out on the workspace proceed, strictly speaking, in a sequential, linear manner; that is, iteratively. This should not be seen as surprising at all, or incompatible with the recursive generation I claimed underlies *merge*. The latter is a characteristic of (an aspect of) what *merge* does, while its iterative application describes how *merge* proceeds in an implementation. In this sense, the analogy to the "iterative conception of set" turns out to be very apt indeed. That is, what *merge* does at heart is generating a K_{n+1} syntactic object from K_n, and it does so by reapplying this recursive generation in an iterative manner. In the conception of C&S, this can be described in terms of how the different stages of a derivation are sequenced. In a simplified form, we can state that if a derivation is a finite sequence of stages $S_1, \ldots, S_n$, then for all the lexical items of a numeration, S_i derives S_{i+1} (see C&S, p. 11 for a complete definition of a derivation and its stages).

The iterative manner in which *merge* proceeds has actually been explicitly pointed out by various scholars; to provide but three quotations, Chomsky (2008) describes Merge as 'iterable', given that it can 'iterate without limit' (p. 139); Di Sciullo & Isac (2008) emphasise that *merge* 'iteratively select[s] items from the numeration' (p. 261); and Jackendoff (2006) correctly describes the computations of *minimalist* syntax as 'Turing-machine-style' (p. 4) —and the latter *is* an iterator.[43]

This is not to say that a derivation could not proceed recursively, and in this sense it may be useful to compare the architecture described so far with a proposal from the literature to re-orient the direction of linguistic computations from a bottom-up to a top-down perspective; doing so, it has been claimed, would result in clearly recursive derivations (Zwart, 2011a). As far as I have been able to determine, though, *merge*-based top-down derivations were first argued for in C. Phillips (2003), but his reasons for putting forward such a proposal had nothing to do with recursion. Indeed, Phillips argued that top-down derivations gave a better account of a number of linguistic phenomena and the corresponding grammatical judgements; that is, it was an entirely linguistic argument that had no relation to design

the order is to be A-B instead of B-A, then A must be linearised before B is, which naturally results in a hierarchy of sub-routines. Whilst this is an interesting proposal —and an unequivocal example of a recursive operation—, as stated at the beginning of this chapter, I am only going to focus on the operations of narrow syntax here, and not on what happens at the interfaces.

[43]Recall that I claimed in chapter 2 that Chomsky has always focused on the underlying recursive generation I outlined therein, while Di Sciullo & Isac have defined the recursiveness of *merge* in terms of how it applies over its own output, a trivially correct but incomplete and imprecise characterisation. Jackendoff is a different case altogether, as I do not think he has so far recognised the distinction between a recursive generation and a recursive implementation. Similarly, both Bickerton (2009) and Luuk & Luuk (2011) fail to recognise the recursive nature of linguistic generation because they exclusively focus on the iterative shape of the derivations. That is, they fail to distinguish between *generation* and a *theory of the computation*.

features of the overall architecture. In addition, Phillips argued that top-down derivations were better-suited to account for the competence/performance dichotomy, perhaps reflecting a fact about human cognition: the only sort of computations the mind carries out are left-to-right.[44]

Naturally, a top-down derivational theory faces the very same pre-eminent questions than a bottom-up perspective does. Among others: what starts and drives the derivation? How are nesting and order achieved? And in particular, why would a derivation start with the subject position instead of the underlying argument structure? Unsurprisingly, Zwart (2009) readily admits that argument structure plays no role in top-down derivations, and this may well be a significant shortcoming. Nevertheless, he does outline the manner in which hierarchy and order are produced, which is closely related to the manner in which he envisions recursion to be implicated in this type of structure building. He specifies a *split-merge* combinatory operation that takes a numeration and "splits" it into a pair consisting of a LI and the remainder of the numeration (p. 163). The derivation proceeds with the splitting of each residue into another pair of objects until the numeration —which is reduced by one after each operation of *split-merge*— is empty.[45]

Note that what this process implies is that every stage (that is, every "split-up") generates a two-object set that can be graphically captured with the binary nodes so common of linguists' syntactic trees; namely, a head (the lexical item) and the remaining non-head material. The labelling of each syntactic phrase simply follows from the head/non-head distinction (ibid., p. 165), and given that every stage of the derivation involves a further splitting of the numeration, the process proceeds downwards until the bottom of the tree is reached. Moreover, and much like Di Sciullo & Isac (2008), Zwart (2011a) defends the idea that specifiers and adjuncts must be generated in different derivations, which are then employed in the main derivation, a take on the geometry of derivations he naturally links to *split-merge*. Contra Di Sciullo & Isac (2008), though, Zwart is not postulating parallel derivations that are then merged somehow (according to Di Sciullo & Isac, the result of the (parallel) derivation of a specifier would simply be adjoined into the SpecTP position of the main derivation).[46] Rather, Zwart (2011b) proposes that the output of a derivation for, as it might be, a specifier (for instance, the subject

[44]In C. Phillips & Lewis (2009) these architectural issues are discussed with a bit more care and the connection between the direction of derivations and performance is played down. Chesi (2005), on the other hand, draws such an extremely close connection between a top-down derivation and performance that the *phase* concept of a competence-level analysis becomes a 'complete process' in both parsing and production (p. 67; he confusingly uses the word generation for what many in the literature would term, like I do here, production). I suppose that such a standpoint is a clear violation of the process neutrality demanded by Sag et al. (2003).

[45]Note that Zwart's *split-merge* only generates asymmetric structures, and it is therefore not clear how it would account for Moro's points of symmetry. Zwart (2011b) points to Fortuny (2008) as a demonstration that there are only asymmetric structures in linguistic derivations (see reference therein).

[46]I am not at all sure if the account in Di Sciullo & Isac (2008) is compatible with the generally-held view that the subject NP is initially merged in SpecvP as the external argument of the verb and later in the derivation copied to SpecTP in order to check its *epp* feature.

of a sentence), constitutes an input for the main derivation. In this sense, he sees evidence for "layered derivations" in which the procedure of the main derivation involves running the procedure of an internal derivation anew (pp. 46–8).

Consider a simple example from Zwart (2011a, p. 48) as an illustration. A numeration such as {[the man], kicked, the ball} includes [the man] as a single and therefore independent unit, suggesting that this unit is the output of a derivation of a different numeration, namely {the, man}. According to Zwart, the very first stage of the derivation for {the, man} involves splitting the numeration into *[the]* and *[man]*, which is then followed by another application of *split-merge* that returns the pair *[the man]* and the empty set ($\emptyset$) as output.[47] This product is then inputted into the main derivation and the first stage of the latter involves the splitting of the numeration {[the man], kicked, the ball} into *[the man]* and {kicked, the ball}. It is this layering of derivations that Zwart links to a recursive type of computation, as the procedure for deriving [the man] is ran again in the first stage of the main derivation.

Let us accept, arguendo, that this layering of derivations is indeed recursive.[48] It must be stressed, however, that the whole approach is predicated on whether a top-down derivation is in fact shown to be the best way to characterise linguistic competence. According to this perspective, then, the place of recursion within the language faculty is established, quite literally, by fiat, its central role in language no more than a mere coincidence stemming from the specific manner in which top-down derivations are conceived. More importantly, Zwart appears to be completely oblivious of the concept *recursive generation*, including its introduction into linguistics by Chomsky in the 1950s and its posterior development into the set-operator *merge*. Like many others in the literature, his starting assumption is that recursion means, roughly, self-embedded sentences, and from this he proceeds to restate its place within the language faculty in terms of a property of a specific type of derivation. Considering what has been discussed here, such a re-conceptualisation is not only misguided, but rather disingenuous.[49]

To sum up, this section was devoted to an analysis of the factors at play within the construction of PHON-SEM pairs in order to ascertain the type of structure (or shape) a derivation exhibits. In general terms, it was here shown that a) lexical items have a rather rich internal structure, b) *merge* operates over lexical items in conjunction with other mechanisms (*select*, *minimal search*, *transfer*, ecc.), and c) the behaviour of these mechanisms is constrained by both general computational properties and interfaces conditions. In conjunction, all these factors result in a process that does not contain any deferred operations, the hallmark of recursive pro-

[47]The empty set is postulated so that further material can be added to the output of this derivation.

[48]I suppose the layering can be construed as being recursive in the sense that the derivation of X (the whole sentence) calls itself during its course in order to derive x (a sub-part of X). That is, the derivation of X can only be completed once the derivation of x has been resolved.

[49]Zwart is furthermore confused about the difference between recursive structures and recursive procedures, a problematic that boils down to a failure to distinguish between the description of a "state" and the description of a "process", as I will argue in section 3.3 below.

cesses, suggesting that a syntactic derivation is an iterative succession of recursive generations, much like "the iterative conception of set" described in chapter 2.

3.2 The generativity of the Language Faculty

The previous section outlined how *merge* derives linguistic structures in a rather abstract manner; here, I turn to a more concrete characterisation in order to offer a glance of the structural richness of some of the expressions the language faculty generates. The data to be described in this section will be of great relevance for chapter 5 below, as they relate to a seemingly widely-held belief in cognitive science; namely, that linguistic structure can be reduced to general features of cognition, the very topic of that chapter.

Admittedly, the concept of a *recursive generation* —that is, the derivation of a syntactic object K_{n+1} from K_n, another syntactic object— is rather abstract and prima facie not transparently attested in the actual expressions linguists study. In actual fact, however, and following Fukui (2011), the syntax of Japanese does exhibit a "pure" version of *recursive generation* in the manner in which clauses can be expanded by adjoining material at the edge.[50] Consider the following sentence, taken from Fukui (2011, p. 89):

(3.5)	*Dare-mo*	*(sono*	*gakkai-ni)*	*ko-nakat-ta.*
	anybody	that	conference	come-NOT-PAST

'Nobody came (to that conference)'.

According to Fukui, further noun phrases (with the relevant nominative marker) can be added at the left edge of these structures indefinitely and without affecting their grammaticality (ibid.):[51]

(3.6)	*Daigakuinsei-ga*	*[dare-mo (sono gakkai-ni) konakatta].*
	graduate students-NOM	

'As for graduate students, none of them came (to that conference)'.

(3.7)	*Seisuuron-ga*	*[daigakuinsei-ga [dare-mo (sono gakkai-ni)*
	number theory-NOM	
	konakatta]].	

[50]Adjoining material to the edge of SOs is probably the closest that language comes to transparently instantiating the $+1$ relation subsuming the successor function and recursive generation. Such an instantiation of recursive generation is very similar to the *adjunction* operation of TAG, and insofar as TAG allocates the recursive property of language therein, *minimalism* and TAG are equivalent. Adjunction thus understood is not exactly the same as the adjunction structure that is usually differentiated from arguments, even if the two sometimes coincide (see infra in relation to some Spanish data). In this sense, the belief of Hornstein & Pietroski's (2009, p. 131) that 'adjunction is the truly recursive part of natural language grammar' is not quite correct.

[51]For the most part, I adhere to the notation of Fukui in these examples. Parentheses include optional material, while square brackets indicate embedded structure. In each gloss, I only include a description of the new material, as the rest remains invariant from the previous example.

'As for number theory, none of the graduate students (in that field) came (to that conference)'.

(3.8) *Suugakuka-ga* *[seisuuron-ga* *[daigakuinsei-ga*
mathematics department-NOM
[dare-mo (sono gakkai-ni) konakatta]]].

'As for the mathematics department, in the area of number theory, none of the graduate students (in that field) came (to that conference)'.

(3.9) *Harvard-ga* *[suugakuka-ga* *[seisuuron-ga* *[daigakuinsei-ga*
Harvard-NOM
[dare-mo (sono gakkai-ni) konakatta]]]].

'As for Harvard, none of the graduate students in the mathematics department in the area of number theory came (to that conference)'.

To be sure, Fukui does point out that the allocation of meaning to these structures may be affected by whether an "interpretative relation" can be constructed between the noun phrase and the sentence it is adjoined to, but he assures us that formal properties are not violated (ibid., p. 90); that is, these sentences remain grammatical, a phenomenon that apparently also applies to the expansion of noun phrases (ibid., p. 88).

Note, then, that *merge* derives these structures by simply adjoining a syntactic object XP (namely, a noun phrase) to a more complex syntactic object (a CP, a full sentence), thereby creating another, more intricate, syntactic object (a new CP).[52] Fukui explains this phenomenon in terms of an "unbounded" *merge* that is in full force in the syntax of Japanese (ibid., p. 90), and while I do not wish to manifest dissent with this view, a more congruous interpretation would simply state that in the case of these Japanese structures, *recursive generation* applies in a "pure" fashion, free of constraints. In this sense, Fukui may be right to suggest that these Japanese constructions are the most natural or genuine structures in language (ibid., pp. 91–2); if this is indeed so, it would mean that what in fact needs to be explained is why other languages do not exhibit them.

Indeed, when *recursive generation* is constrained by other factors —such as the specific lexical features that can enter a derivation—, the language faculty derives rather different structures. In the case of Spanish, for instance, even simple, declarative sentences evidence a rather intricate structure, one that is prima facie particular to language. Consider the following pair:

(3.10)

El	*candidato*	*del*	*partido*	*se*	*preparó*	*el*	*próximo*	*discurso.*
The	candidate	of the	party		prepared	the	next	speech.

[52]Note the asymmetry —that is, hierarchy— between this adjoined XP object and the full CP sentence. I take it that this is what Chomsky (2008) means when he states that hierarchy is automatic for recursive operations (ft. 12, p. 137).

'The party's candidate prepared his next speech'.

(3.11)	*El*	*candidato*	*ha*	*preparado*	*un*	*discurso*	*sobre*	*la*	*sanidad.*
	The	candidate	has	prepared	a	speech	about	the	health service.

'The candidate has prepared a speech about the health service'.

Whilst these two subject-verb-object sentences exhibit prima facie similar structures, it is worth discussing their peculiarities in some detail. In the case of (3.10), the derivation would start by pair-merging the adjunct *próximo* to *discurso*. The resultant SO would then be set-merged with the determiner *el* —that is, the determiner selects, in the parlance of Stabler (2011), the head of the complex [próximo discurso]. At this stage, the XP so far generated would be selected by the verbal phrase *se preparó* as its complement, thereby forming a head-XP complex; call this *first-merge* (or the VP stage). Following upon this, the specifier of the head-XP object would be constructed and pair-merged, a process that would involve either a parallel derivation (as Di Sciullo & Isac 2008 posit; that is, *partido* and *del* would be merged first, and so on until the complex [el candidato del partido] is derived) or simply an assembling within the main derivation; in any case, call this stage *second-merge*. Note that the last stage specifies the argument structure of the sentence and constitutes a *phase* —the vP stage, as stated above—, to be followed by whatever internal applications of *merge* complete the derivation.[53]

Regarding (3.11), the derivation exhibits a succession of applications of *set-merge* until the specifier *el candidato* is pair-merged. That is, *la* selects, and is therefore merged with, *sanidad*; the complex [la sanidad] is then selected by *sobre*, and the resultant SO [sobre la sanidad] is in turn merged with *discurso*; *un* is merged with the SO so far derived and the whole structure is selected by the verb *ha preparado* as its complement (*first-merge*). It is at this stage that [el candidato] is pair-merged as the specifier of the SO in W, and the rest of the derivation would proceed, roughly as for (3.10).

Recall that all applications of *merge* are asymmetrical in the sense that for every SO, one of its elements projects, thereby becoming the head and selectee for the next LI to be introduced in the derivation. The distinction between *set-merge* and *pair-merge*, however, involves a sort of asymmetry that is perhaps easier to perceive in a different "plane". Consider the trees for these two sentences in order to clarify.

As the two figures show, there is a clear asymmetry between the subject and the object positions in regards to how they stand towards the verb. That is, while the object stands immediately next to the verb as its complement —thereby forming a unit—, the subject is altogether in a different plane, as it appears to simply be

[53]Some of these include the verb attaining the appropriate tense, which involves being copied to the head position of the TP phrase, and the satisfaction of the *epp* feature of the noun (that is, being copied from SpecvP to SpecTP)

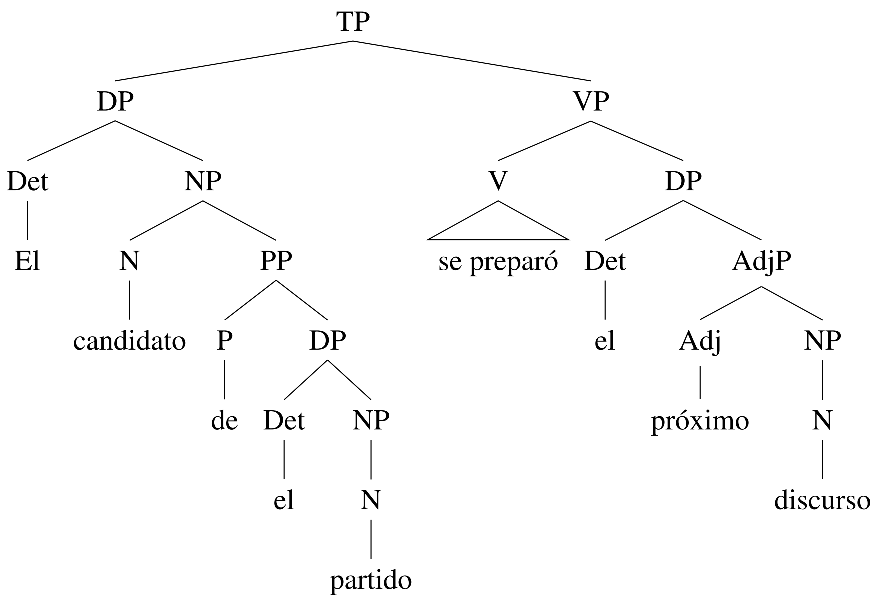

Figure 3.2: A diagram of ‘el candidato del partido se preparó el próximo discurso’

adjoined to the rest of the structure in a hierarchically more prominent position. Note, further, that a similar geometry holds for how the adjective *próximo* relates to the noun *discurso* in (3.10); that is, the adjective is an adjunct of the noun. While adjuncts and specifiers are the direct products of *pair-merge*, *set-merge* compiles verb-complement and NP-PP configurations.

In the derivations just described, I have ignored the fact that the morphology of a verb indicates a number of features —mainly, *number* and *person*— that require the establishment of an agreement relation with a noun. These properties of lexical items are usually called ϕ-features and the operation computing the interaction between a noun and a verb in these terms simply *agree*.[54] According to Chomsky (2001a), the agreement relation between nouns and verbs is asymmetrical, in the sense that the verb's ϕ-features depend on the noun's ϕ-features; or in other words, the ϕ-features of verbs are unvalued in the lexicon and become valued in the course of the derivation by entering in an agreement relation with a noun. This is supposed to be so because the interpretation of a verb does not rest on these features, while that of a noun demonstrably does. Consequently, the *agree* operation applies just in those cases in which the noun and the verb enter into an appropriate configurational relation —namely, in a specifier-head configuration.

In the case of (3.10) and (3.11), a step of the derivation would involve bringing about such a configuration so that the ϕ-features *3rd person* and *singular* of

[54]I am ignoring the *gender* feature for reasons that will be obvious soon enough.

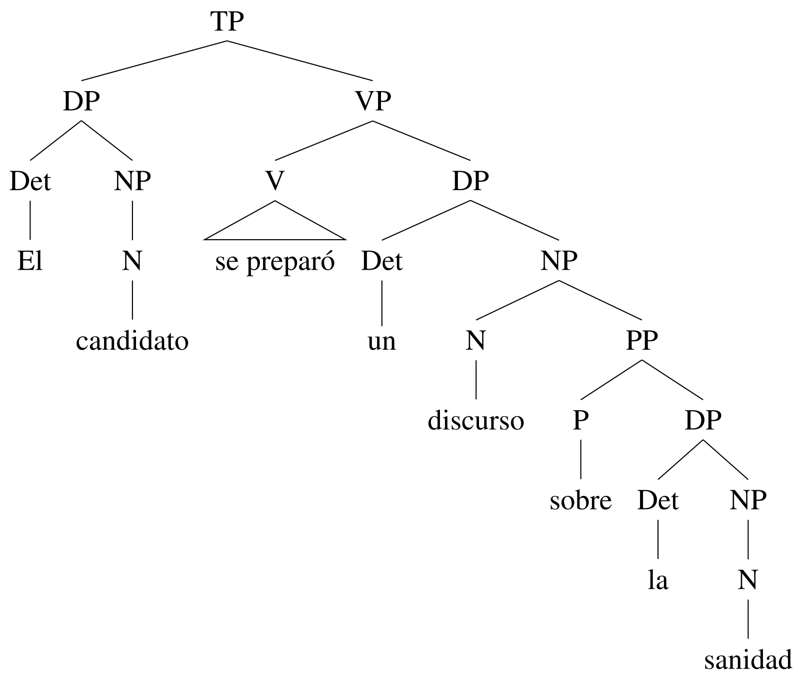

Figure 3.3: A diagram of ‘el candidato se preparó un discurso sobre la sanidad’

el candidato value those of the verb, and in both (3.10) and (3.11), this would come about in a pretty straightforward manner. Other Spanish sentences, however, involve much more intricate agreement relations. A particularly curious phenomenon goes by the name of “unagreement” (see Ackema & Neeleman 2011 for a recent take), as exemplified in sentences that prima facie manifest a mismatch of ϕ-features between a subject noun and a verb. Consider the sentence below as an example:

(3.12)	*Los*	*lingüistas*	*escribimos*	*un*	*gran*
	The	linguists-3rdperson-plural	wrote-1stperson-plural	a	great
	artículo.				
	paper.				

‘We the linguists wrote a great paper’.

In (3.12), while the subject noun appears in the third person plural and the verb in the first person plural, the sentence remains grammatical nonetheless. At first sight, a native speaker of the Spanish language might remonstrate that *los lingüistas* is *not* the subject of the sentence at all, and that (3.12) is nothing more than another example of the ability of this language to *drop* the subject. That is, a correct representation for this sentence would be something like (3.13) below, in contrast to (3.14).

(3.13)

pro	*los*	*lingüistas*	*escribimos*	*un*	*gran*	*artículo.*
	The	linguists-3^{rd}person-plural	wrote-1^{st}person-plural	a	great	paper.

'We the linguists wrote a great paper'.

(3.14)

Nosotros	*los*	*lingüistas*	*escribimos*	*un*	*gran*	*artículo.*
We	the	linguists-3^{rd}person-plural	wrote-1^{st}person-plural	a	great	paper.

'We the linguists wrote a great paper'.

If so, the pair *pro los lingüistas/nosotros los lingüistas* would quite simply be a case of nouns in apposition. According to Ackema & Neeleman (2011), however, such a proposal is untenable for a number of reasons. First of all, the domain of the unagreement phenomenon is very constrained, as it only applies to specific configurations; namely, it only arises when the subject is in the third person plural and the verb is either in the first or second person plural (ibid., p. 11; see examples therein). Clearly, if all there was to unagreement was a case of apposition, we would expect this unagreement spectacle to be more widespread.

Further, and following Cardinaletti & Starke (1999), Ackema & Neeleman (2011) point to the distinction between *strong* and *weak* pronouns —such as the English *he* and *it*, respectively— which have different syntactic distributions.[55] Assuming, with Cardinaletti & Starke (1999), that null subjects should be classified as weak pronouns, Ackema & Neeleman (2011) conclude that the sentence in (3.12) cannot be a case of apposition, as weak pronouns cannot appear in such structures; *los lingüistas* must really be the subject of the sentence.[56]

Subsequently, and in contrast to the *minimalist* account of how *agree* operates,[57] Ackema & Neeleman (2011) put forward a proposal in which agreement relations are symmetrical in the sense that 'ϕ-features are generated on the verb independently of the ϕ-features of the argument' (p. 5). Naturally, the verb and

[55]Roughly, strong pronouns can a) appear in any position within a sentence (weak pronouns cannot); b) cannot have non-human referents (weak pronouns can); and, c) can be coordinated with other noun phrases (weak pronouns cannot); see Cardinaletti & Starke (1999, pp. 150–2) for further details.

[56]Cardinaletti & Starke (1999) provide enough evidence for both the existence of weak pronouns and for the classification of null subjects as such, while Ackema & Neeleman (2011) show that weak pronouns cannot appear in apposition in German and Dutch (the implication being that null subjects in Spanish should not be able to appear in apposition either, perhaps an unwarranted conclusion); see details therein.

[57]Mancini et al. (2011) also discuss the unagreement data and offer a *minimalist* explanation in which *agree* is simply posited to apply in reverse in those cases (that is, from the verb to the noun.

the noun still need to enter in an agreement relation in order for these features to be licensed at the semantic interface, but this, according to Ackema & Neeleman (2011), is brought about by a process that employs a number of operations adapted from auto-segmental phonology (see reference therein, loc. cit). Namely, root nodes are regarded as segments; features are represented in different tiers —ϕ-nodes; a single ϕ-feature can be associated with two of these ϕ-nodes; ϕ-features can "spread" from one segment to another; and some of these ϕ-nodes can be eliminated (ibid.). Accordingly, Spanish unagreement would be explainable in terms of a relation in which the subject noun phrase has fewer ϕ-features than the verb, and feature spreading, identification et alia conspire to construct a representation that is interpretable by the semantic component (see pp. 12–3, loc. cit.).

To be sure, the Spanish data Ackema & Neeleman (2011) discuss are just not enough to argue for a symmetrical theory of agreement, and I do not wish to endorse their account here. My intention in this section was merely to point to some of the particularities of linguistic structure, such as the adjunction of noun phrases to clauses, the asymmetry of subjects and adjuncts, the agreement between nouns and verbs, null subjects and unagreement, and many others. Admittedly, I have offered a very small sample of the sort of sentences Japanese and Spanish manifest, and an even smaller sample of the structures the language faculty can generate in toto. Nevertheless, this collection is very illustrative, I believe, of some of the general features of linguistic expressions, and as I will show in chapter 5, no other domain of human cognition exhibits anything remotely similar. That is, none of the very general features I have discussed here are transparently present in the structures of any other cognitive domain.[58]

Therefore, the main claim of this section is simply that a number of factors —lexical features, interface conditions and recursive generation— conspire to yield a specific "generativity", one that is unique to language. Such a perspective, I might add, is not so dissimilar to what I take M. D. Hauser et al. (2002) to have meant by their hypothesis that the faculty of language in the narrow sense (their term for the uniquely human features of language) 'comprises only the core computational mechanisms of recursion as they appear in narrow syntax and the mappings to the interfaces' (p. 1573).

3.3 State and Process Descriptions

It should be noted that two different types of hierarchical representations have so far featured in this chapter —one implicitly, the other explicitly. The more obvious of the two is the tree representations so typical in many a linguistic study. In early studies of generative grammar (e.g., Chomsky 1957, p. 40), this sort of representations was taken to codify the derivation history of a structure, but they

[58]Further, the simple declarative sentences I have described in this section will feature in the experimental data to be discussed in the next chapter, and therefore the material here constitutes an appropriate primer for what is to be there discussed.

should be more appropriately viewed as an illustration of the relationships that lexical items form within a sentence. After all, the tree structures I have employed in this chapter do not transparently specify the operations of the combinatory operation that constructs sentences. It is precisely the organisation of the operations that *merge* conducts during a derivation that constitutes the second type of hierarchical representation I have here discussed.

Relatedly, and in the context of discussing complex systems, Simon (1962) draws a distinction between the description of a state and the description of a process. The former, according to this scholar, focuses on an 'object as sensed', while the latter centres on 'the object as acted upon' (p. 479). Furthermore, it is by carrying out the instructions of the process description that you achieve what the state description specifies (ibid.), a statement that naturally resonates with some of the issues that have been discussed in this thesis.[59] Indeed, in chapter 1 it was clearly shown that you could describe as recursive both a process and an object in their own terms and independently of each other, a fact that seems to have escaped Zwart (2011a).

Indeed, the manner in which the latter discusses the general issues at hand is rather puzzling. Right after stating that

> 'the procedure that draws the famous Droste can picture [see Fig. 5.4 in chapter 5], at some point involves calling the very same procedure, in order to draw that can in the picture' (p. 43)

(a procedure that plausibly specifies a recursive process), he proceeds to tell us that not only 'one cannot tell that an object is recursive by simply looking at it' (ibid.), but that the Droste picture was probably generated in a non-recursive manner after all (all in the space of one single page). Admittedly, Zwart is simply confusing a state with a process description, and by a "recursive object" he simply means a recursively-derived object, an entirely different matter. Be that as it may, there is an important difference between these two hierarchical constructs, a distinction that is crucial for linguists and psycholinguists alike and that will feature extensively in the ensuing two chapters.

Stabler (2010) calls these two representations the "derived" and the "derivation" tree structures, for state and process descriptions, respectively. Fig 3.4, adapted from Stabler (2010), graphically represents the differences between the two tree structures more clearly.

The tree on the left-hand side represents the derived version of *who Juliet loves*, and every element is in the right and final geometrical position (ignore the *t's* and the numbers in brackets). The derivation tree on the right-hand side, on the other hand, codifies the operations of the computational system, with black dots standing for operations of *external merge* and white dots identifying instances of *internal*

[59]There is another point of Simon's study that is also apposite for my examination; namely, that in order to properly describe a complex system it is necessary to understand the properties of its parts and the laws of their interaction if we are to describe them properly (loc. cit., p. 468).

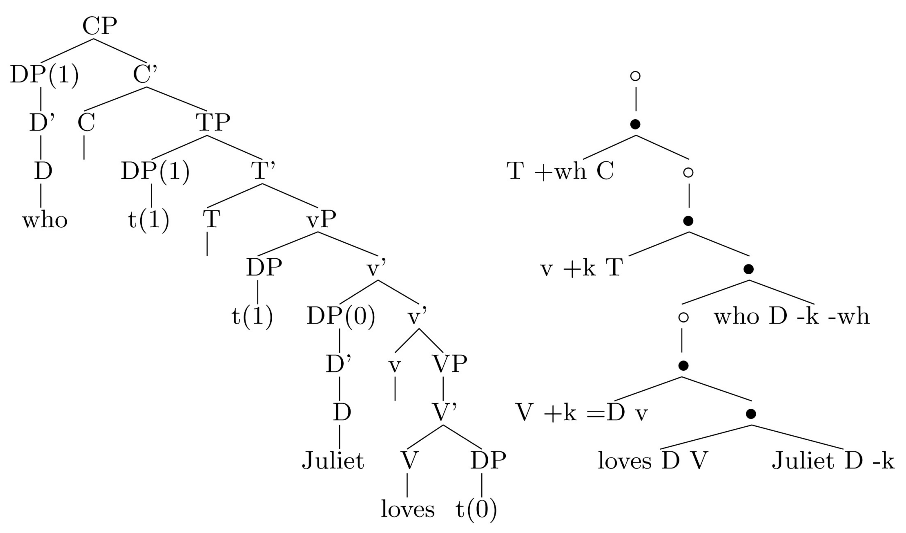

Figure 3.4: Derived and Derivation Tree Structures

merge. As the graphic clearly shows, each stage of the derivation is licensed by the features that apply therein.[60]

Clearly, it is the representation on the right-hand side that ought to focus the attention of linguists, as it is therein that derivations are explained. That is, an explanation of linguistic facts centres on the underlying properties and principles that license some structures and not others, and this is specifically codified on the right-hand side.[61] Naturally, the tree on the left-hand side represents the product, the result of running the operations the grammar effects, but such representations have perhaps received a disproportionate amount of attention among linguists compared to a derivation tree structure. It is one thing to map the geometry and landscape of linguistic structures, it is another thing completely to offer an explanation for the relevant facts.[62]

In a way, a derived tree structure constitutes a good nexus of contact with other

[60]Capital letters stand for categorial features (*T* for tense, *C* for complementiser, ecc.), lower-case symbols are either probes (+) or goals (−; *k* stands for case) and *=D* is a selector feature (in this case, the verbal head selects the D *Juliet*).

[61]P. H. Miller (1999) makes a very similar point while discussing issues related to formal language theory. Within that field of study, a grammar is said to *weakly generate* if it generates the strings of a language and to *strongly generate* if it generates the structural descriptions of such strings. Naturally, linguists are interested in the latter construct, but as Miller emphasises, the structural descriptions of a given formalism (he discusses TAG) are provided by the derivation tree, not the derived tree. This is a rather important point.

[62]This is not a criticism of representational approaches; such theories also postulate abstract properties that are not always transparently represented in a derived tree structure.

fields of study within cognitive science. After all, a linguist ought to be very keen to provide the set of representations the linguistic domain exhibits in order to compare and/or relate them to those of other domains of the mind. More specifically, it is plausible that derived tree structures participate in the operations of "thought systems" involved in planning, categorisation and belief fixation; if so, interdisciplinary work is a necessity rather than a luxury. There is little point, however, in relating the features of a derivation tree to the operations of other cognitive domains, as these are clearly inapplicable outside of language. Indeed, lexical features and (rather specific) interface conditions appear to be sui generis characteristics of language.

These characteristics give rise to rather remarkable structures, such as the fact that all syntactic phrases are headed and asymmetric, as specified in the [Specifier [Head - Complement(s)]] scheme. More remarkable is the very fact that clauses are nothing but configurations of such structures.[63] If Chomsky (1995a) is right, however, these configurations reduce to much simpler principles. In particular, both the specifier and the complement position should be regarded as nothing more than the manner in which local relations to a head are computed (ibid., p. 397). If such a reduction is at all feasible, then linguistic structure is just a collection of lexical elements and the sets constructed from them (p. 415).

Be that as it may, the resultant set of lexical items must still exhibit headedness and asymmetry, for these are unambiguous properties of language.[64] In a perhaps more daring manner, these very structures are argued to be a reflection of natural law by Medeiros (2008), Piattelli-Palmarini & Uriagereka (2008) and Soschen (2008). Specifically, these scholars draw an analogy between [Specifier [Head - Complement(s)]] structures and the manner in which Fibonacci sequences 'adapt to... [a] space of possibilities' (Piattelli-Palmarini & Uriagereka, 2008, p. 222). That is, it is an analogy between Fibonacci patterns qua an "efficient space filling" system (Soschen, 2008, p. 198) and tree building qua maximization of syntactic structure (say, by filling all specifiers). I cannot discuss this matter to any significant detail, but it does strike me as rather far-fetched, specially considering that in a certain sense, a derived tree structure is a sort of epiphenomenon. The real linguistic phenomena are those specified in the derivation trees of syntactic objects, and no such analogy between them and mathematical structures appears to be possible.[65]

[63]Remarkably, scholars who focus on representational constraints, such as what some of these perceive to be the finer structure of functional projections, can advance rather counter-intuitive claims. For instance, Boeckx (2006, p. 53) mentions that the linguist Cinque posits about 50 embedded phrases for simple, declarative sentences. Note that if the latter is right, isomorphic structures in other domains of the mind would have to exhibit a similar structure.

[64]In a sense, Chomsky (1995a) attempts to eliminate a number of, for him, spurious assumptions, such as the distinction between a category and a terminal, or bar levels and head projections. Still, the "bare phrase structures" he describes in pages 414–15 remains asymmetrical and endocentric in the relevant sense.

[65]As stated above, constructs such as TPs, CPs, ecc. ought to be seen as nothing more than labels for specific stages of the structure-building process. That is, even though they can be described as

*****

The present chapter has focused on the abstract implementation the language faculty effects —that is, the computational process (what linguists call the syntactic derivation) that generates PHON-SEM pairs from lexical items. In doing so, it was here established that the "shape" of a derivation is clearly iterative in nature. This is the case because *merge* resolves its particular operation at each step, rather than *deferring* it to some later point in the derivation.

Note that *deferred* operations should not be confused with *delayed* operations, as the latter actually abound in syntactic derivations. When a lexical item is introduced into a derivation, it will carry a specific set of features that need to be checked or valued, but it is not always the case for the entire set to be checked/valued at the very same time. For instance, a subject noun phrase is initially merged as the external argument of a verb, but it will feature in a later stage of the derivation once more in order to check its *epp* feature, thereby becoming the "subject" of the sentence being constructed. Be that as it may, every single application of *merge* is complete at any particular step, in the sense that none of its applications involve a resolution of simpler, internal instances. That is, the evaluation of an *epp* feature does not involve the precedent evaluation of an internal, simpler *epp* feature. Naturally, a number of operations must take place before the *epp* feature can be checked, and there is certainly some delay between the moment the lexical item bearing this feature is entered into the derivation and its eventual evaluation. That is certainly the main point here, this is nothing more than a delayed operation.

In any case, it is not too hard to imagine a recursive process similar to what was discussed above for top-down derivations. As Postma & Rooryck (2007) discuss in an unpublished paper, it could well be the case that the completion of a given *phase* necessitates the resolution of an internal *phase*. In this case, a *phase* would be calling itself, creating a deferred operation; that is, the completion of the macro *phase* would only be possible once the internal *phase* had been resolved. This would clearly be a recursive sub-operation, but the evidence for such layering of *phases* is pretty thin at present.

I must say, however, that I do not consider this result to be of much substance, at least not for the subject matter of this thesis. After all, the eventual right description of a derivation will be what will be, and there is very little point in relating it to the role of recursion within the language faculty in the critical sense that has been maintained throughout here. The two phenomena are clearly distinct. That is, even if it were the case for derivations to apply recursively, this would be a fact about the abstract implementations *merge* effects, but recursive generation would nonetheless remain the main property underlying the language faculty.

structures, this is only so under a specific perspective —that of a state description.

Nevertheless, the material presented here has advanced on the explanatory plan laid out in section 1.1 and I now turn to the last stage; that is, to an analysis of the real-time implementations that must subsume linguistic comprehension. Within the greater scheme of things that this doctoral thesis is, this chapter constitutes the pertinent background for the discussion to be carried out in chapter 4.

Chapter 4

Recursive parsing

4.1 The problem of Language Comprehension

The previous chapter described how *merge* compiles sound-meaning pairs from lexical items, a construction driven by the intrinsic and formal properties of lexical items in combination with the interface conditions the derivations must meet. These sound-meaning pairs determine the manner in which sentences are understood —they are, after all, the products of the function that is computed from atoms to structures— but they do not specify the way in which the underlying function is in fact calculated in real time.[1]

This general point alludes to one of the main problems facing the psycholinguist: how to relate competence and performance. In a sense, however, the dichotomy should not stress the psycholinguist too much if Fodor et al. (1974) are right. That is, the linguist's theory of grammar provides a description of the structural properties the parser decodes and encodes, which must be reflected, somehow, in processing operations.[2]

Naturally enough, the literature contains many examples of scholars who have attempted to link grammatical principles to specific operations of the parser, more often than not unsuccessfully. G. A. Miller & Chomsky (1963) were perhaps too bold to propose the *derivational theory of complexity* (DTC) in the terms they employed, but there must be some truth to the general idea. That is, whilst it is perhaps unreasonable to suggest that derivational cycles go hand-in-hand with matching parsing operations, general features of structural complexity must surely be re-

[1]Cf. Bever (1970): 'the form in which sentences are understood and memorized corresponds closely to the internal syntactic structure internal to them' (p. 287).

[2]As I will discuss soon enough, and in consonance with what I stated in the previous chapter, "the structural properties" the parser decodes and encodes are necessarily those specified by a derivation tree, not those of a derived tree. At the time Fodor et al. was published, however, the internal constitution of both derivation and derived trees seemed to be exactly the same, and so this issue did not arise (in my opinion, this close match was the result of the specific manner in which linguists employed systems of rewriting rules, but this is not the place to discuss this point).

flected in processing load.[3] The problem is to find out what structural properties precisely are reflected in parsing.

In a related note, Moro (2008) is certainly simplifying these issues a great deal when he offers the following formula as a description of the general state of affairs: $Performance = Competence + Time$. Whilst it is true that unparsable, self-embedded sentences such as *bulls bulls bulls fight fight fight* can eventually be regarded as grammatical with enough time and space (say, by employing pen and paper to analyse the sentence), this seems to be a fact about the *recogniser* —the mental component that identifies specific strings as being grammatical. The relation between competence and performance, however, is clearly qualitatively different.

At a minimum, then, performance must be composed of a *grammar*, a *parser* and a *recogniser* (Wolf & Gibson, 2003). The last component is plausibly not directly involved in online syntactic parsing, it must instead reflect the (subsequent) ability to recognise grammatical/acceptable sentences (cf. Stich's faculty of grammatical judgement, mentioned in chapter 1). It consequently operates on the output of the parser, but it is not involved in establishing whether the parses are licit or not; as stated earlier, the latter is established by the grammar.

How the grammar applies in real-time processing is without a doubt the most contentious factor. A linguist's grammar is certainly too rich and specific as a model for parsing (think of Cinque's multiple functional projections, for instance),[4] a point that has long been recognised by those computer scientists working within the field of *natural language processing* (NLP; Steedman 1994). It is common ground for a NLP parser to be endowed with a grammar that is very different from the *competence* grammar linguists postulate. Such a grammar —usually called a *cover grammar*— effects a 'homomorphism mapping [its] structures... onto the structures of the competence grammar' (Steedman, 1994, pp. 4); that is, the two grammars would be equivalent in their "productions", but not in the derivations they effect to generate these productions or in the parsing strategies that might be applied in language processing. Unsurprisingly, using a less complex cover grammar results in a more efficient parse compared to a processor that proceeds according to a full-blown linguistic theory.

Be that as it may, some of the properties unearthed by generative grammars have featured extensively in NLP studies and the overall psycholinguistics literature. Perhaps the most conspicuous case is the widely-held assumption that the parser must recover the tree structure underlying the strings it receives, and three main perspectives as to how the parser "approaches" this problem have been considered. In a bottom-up procedure, the parser attempts to combine words into constituents as they come, while in a top-down approach the processor assumes *there to be* a tree —that is, a sentence— and then proceeds to travel down its branches

[3]Wagers & Phillips (2009) revive the DTC in these very terms.

[4]Cf. Kremers's belief (2009) that the parser recovers a tree that is identical to the tree created by the grammar (see infra).

and nodes all the way to the words themselves (Crocker, 1996). Both approaches are psychologically implausible (ibid.), but a combination of the two into the so-called *left-corner* parser results in a more realistic model. According to the latter, the parser *looks* at the left-hand side corner of a phrase structure rule and projects the category, followed by a chain of predictions regarding the material to follow.[5]

Note that this viewpoint focuses on how the parser operates over what was above termed, following Stabler (2010), the derived tree structure of a sentence, and it does so by directly applying phrase structure rules. Naturally, a derived tree structure is the representational scheme that appears in many a linguistic treatise, a graphic that attempts to codify (most of) the structural relations among the different elements of a sentence. Crucially, however, a derived structure does not exhibit the properties and principles that apply during the derivation; that is, those grammatical principles that *actually* explain how the derivation proceeds into an eventually licit structure —the derivation tree structure that was the focus of the previous chapter. Fig 3.4 is repeated below as (4.1).

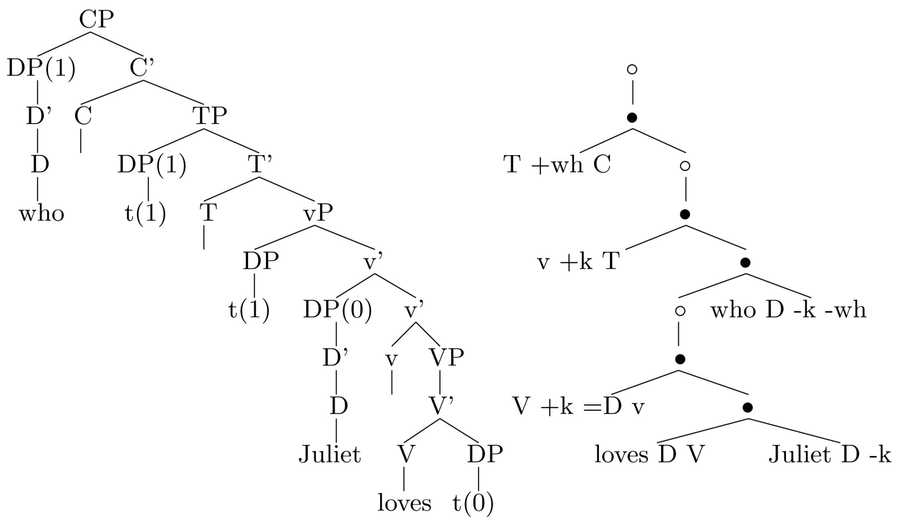

Figure 4.1: Derived and Derivation Tree Structures (repeated)

As stated in chapter 3, the tree on the left-hand side represents the derived version of *who Juliet loves* (recall that CP stands for complementiser phrase, DP is a determiner phrase, and TP is the tense phrase). On the right-hand side, we have the operations of the CS, with black dots standing for operations of *external merge*,

[5]In a *head-corner* model, a modification of the *left-corner* approach, the parser looks for the *head* of the phrase instead of the left-hand side of a rewriting rule (see references in Crocker 1996). This is a strange take on things, however, given that every single lexical item is the head of a unique phrase. To wit, the phrase *the car went into...* is composed of: *the*, the head of a determiner phrase; *car*, the head of a noun phrase; *went*, the head of a verbal phrase; *into*, the head of a prepositional phrase, *e così via*.

while white dots identify instances of *internal merge*; finally, the different letters stand for those features that drive the overall process (recall that the features lexical items bear need to be checked against the appropriate functional nodes).

Note, then, that a psycholinguistic model that employs a derived rather than a derivation tree betrays the very principle that a parser must interact with the grammar in order for its products to be properly constrained. Indeed, it is the operations that apply during a derivation that establish what structures are licit, but these are not (transparently) encoded in the derived tree structure; rather, the latter is the direct result of the former. Therefore, and as Stabler (2010, 2011) argues, psycholinguists need to focus their studies on the derivations *merge* implements, and not on the rather different, and now long-defunct, chains of phrase structure rules that previous theories postulated, as the latter constitute an unrealistic model of linguistic knowledge.

Historically, however, the 1970s saw a proliferation of parsing models that incorporated principles of *government and binding* (GB) Theory —the generative perspective then in vogue— piecemeal. Pritchett's theta-driven parser (a "fill theta roles as you proceed" model; see discussion in Crocker 1996) is a case in point, while Gorrell (1995) combines basic operations of the so-called garden-path model (to be treated in the next section) with a principled-based parser that builds the dominance and precedence relations a tree structure codifies (where "principled-based" makes reference to the structural constraints GB Theory postulated).[6]

This is also the case for processing theories closely connected to much more recent developments, such as the *minimalist program*. A rather recent example is found in Hornstein's (2009) proposal that all grammatical principles and operations ought to meet the *reversability* condition; that is, all derivational principles must be able to apply in the other direction (viz., from bottom-up derivations to left-to-right parsing). Weinberg (2001) seems to have a similar point in mind, as she attempts to apply precedence, c-command relations and even number of *merge* applications to the attachment preferences of the parser. Whilst it is true that grammatical principles must be operative in parsing somehow, it is less clear that their application will be so direct as these two scholars suggest; in fact, both Hornstein and Weinberg appear to assume that they are, but such a position needs to be argued for, it is certainly not axiomatic.[7]

[6]The name "garden-path" makes reference to those sentences that, while grammatical, start in such a way as to lead the interpreter towards an incorrect interpretation, a reference to the saying "being led down the garden path". A famous example is *the horse raced past the barn fell*, a perfectly grammatical example of a reduced relative that is systematically misinterpreted (and usually judged to be ungrammatical). The full version ought to dissipate the reader's puzzlement: *the horse that was raced past the barn, fell*. It is perhaps worth pointing out that such phenomena only ever occur during *reading* (which usually includes the withdrawal of the relevant commas) and it is hard to imagine a hearer noticing the particularity of sentences like these when they are properly intoned in real-life interaction.

[7]It is also worth mentioning that Weinberg navigates through the different parsing models of the literature in a somewhat misleading manner when she divides the different proposals into the following three groups: those that focus on extra-linguistic considerations (such as the garden-path model

In more general terms (that is, less theory-laden), many other models directly implement phrase structure rules and/or a type of derived tree structure. For instance, both the "phrase structure rules plus constraints" model Gibson & Pearlmutter (1998) defend (where the constraints are lexical, contextual, computational resources and phrase-level frequencies), or those parsing models that employ treelets (or pre-compiled phrase structure rules), as defended by Jackendoff (2006), are a case in point. Relatedly, a recent model by Hale (2011) proposes a synthesis of two themes: a) language comprehension involves the application of phrase structure rules and b) broad rationality principles impose what rules are applied at each stage of processing. Much like Hornstein and Weinberg, however, all these authors offer very little in the way of a well-argued defence of their starting assumptions, with the result that the evaluation of these models turns on how well they capture the psychological data. Perhaps this might be judged to be good enough by many in the literature, but it has been an overarching point of this thesis that we need to resolve those "conceptual issues" that enter into the study of cognition if we are to understand the rather intricate properties of mental architecture; in other words, we ought to strive towards an explanation of psychological facts, rather than proposing mere approximations to the data.[8]

The crux of the problem is to figure out what the right balance between grammatical principles and parsing operations is. At this point, it is worth mentioning some of the quintessential tasks the parser must conduct, and I will briefly focus on the very first task I list in order to bring out the core question of the problem of language comprehension. Some of these tasks include: segmenting the string into units (words, clauses, ecc.), assigning syntactic roles to those units (verbal phrase, ecc.; and also, subject, object, ecc.), establishing relations and dependencies between elements; setting up a correspondence between syntactic and thematic roles (agent, patient, ecc.), interacting with the semantics/pragmatic component; et alia.

As defended in section 1.3 and chapter 3, linguistic form is the result of rather intricate computational principles and properties that are internal to the language

or Gibson's 1998 theory; I will come to the latter soon enough); those that defend a constraint-based approach; and those that derive its principles from competence (such as the implementation of the theta-criterion). This is a bit misleading because whatever one thinks of the nexus between parsing operations and grammatical principles, language comprehension remains an intrinsically perceptual problem that is somehow related to the language faculty, and therefore extra-linguistic considerations are a given. It is certainly not the case that the garden-path model, for example, reduces parsing to non-linguistic properties in toto; rather, such models propose an interaction between grammar and processing as much as Weinberg's own proposal does.

[8]I should also point out, given its relation to the work presented here, that Hale (2011) applies Marr's levels of explanation in a rather peculiar way. Surprisingly, Hale only allows information-theoretical complexity metrics and Bayesian probability as operative factors at the computational level (ibid., p. 400), while the algorithmic level is composed of the different sources of information that may affect human performance (lexical bias, prosody, ecc.). The grammar, then, appears to play no role whatsoever at the computational level, apart from its reduction to Bayesian inference; according to this approach, sentence comprehension is but a revision of beliefs regarding possible sequences of words.

faculty, principles and properties that do not emerge from processing phenomena (at least not transparently), and that certainly do not come pre-compiled in the perceptual systems. In this sense, the parser is the result of the connection between the grammar and the perceptual systems, a specific element of the mind that needs to be investigated in its own terms (whatever they might turn out to be). The task of the psycholinguist, then, is to work out how the perceptual systems manipulate linguistic structures —i.e., what strategies and operations they effect for this task— something for which they were intuitively not designed.

Let us focus on the segmentation of the linguistic input into units in order to clarify this point. Plausibly, perception starts when the sensory organs are stimulated by certain physical properties (light, sound, or else), resulting in a process —transduction— in which this input energy is transformed into neural activity. It is the job of specific mental capacities to then a) recognise that some of the shallow representations the transducers produce fall under their domain, and to b) convert this information into appropriate mental representations —the labour of *functional* transduction (Pylyshyn, 1984)— so that these can then be employed by other systems of the mind (as, for example, in the fixation of belief). Naturally, different modalities will engage different systems of the mind, but we are here focused on the linguistic parser that is in charge of constructing the structure of the strings it receives.

That the perceptual systems receive "strings" points to a fundamental distinction between the "language module" and the language faculty. Given that what the parser is inputted is a succession of elements, it must combine them into a complex representation if it is to make sense of the meaning that is being conveyed. That is, the parser does not receive a structured representation; it must recover the underlying structure from the signal. For the parser, then, it is certainly true that language is highly ambiguous at all levels of representation (Crocker, 1996, p. 36). However, it is also important to keep in mind that ambiguity does not exist within the language faculty: every single structure receives a single interpretation; after all, an ambiguous structure is an impossibility.

Be that as it may, situations of ambiguity very rarely arise in real-life interaction, as the context usually helps hearers resolve all potential ambiguities. What I mean by this is that the mechanism underlying ambiguity resolution hardly ever fails in real life. That is, a hearer does not ask a speaker to specify the meaning he intended while using a potentially ambiguous utterance; quite simply, such a situation does not arise. In any case, whatever mechanism is involved in ambiguity resolution, it surely remains in operation regardless of what the extraneous conditions are like, and that is why ambiguity resolution is such a central topic in psycholinguistics (see Pickering & van Gompel 2006 and van Gompel & Pickering 2009 for details). My intention here is to merely point to the rather artificial settings usually created by psycholinguistics, a set of conditions in which such variables as contextual information are in fact usually explicitly eliminated. In these circumstances, the parser is literally receiving isolated strings of elements, a situation that is bound to create a constant state of uncertainty for the processor.

In a much less dramatic sense, the parser is in a state of uncertainty even in real-life interactions, as the first task of the linguistic module is to recognise that some external auditory information is linguistic in nature. As Fernández & Smith Cairns (2011, p. 170) put it, words have to be extracted from a signal that is continuous (there are no spaces between consonants or vowels, or between words), unsegmented and highly articulated (i.e., there is parallel transmission). As they go on to illustrate (pp. 174 et seq.), speech perception relies on the phonemic inventory of the language that has been internalised, and as a result supra-segmental information such as the duration, pitch and amplitude of segments can be employed to identify external noise as linguistic material, including where word boundaries are to be found, paving the way for, and in combination with, lexical retrieval. Naturally, this short description is not an account of how this process pans out, but it does point to the information the parsing system uses; an important property of the nexus between perception and grammar.

A much more careful analysis of this phenomenon is provided by Poeppel, Idsardi & van Wassenhove (2008), and the solution they propose is very illustrative of what the problem of language comprehension involves. Poeppel et al. adopt an analysis-by-synthesis (AxS) approach, a model for speech perception that was initially proposed by Halle & Stevens (1959, 1962). The AxS is based on the observation that in linguistic communication the receiver must recover the intended message from a signal for which he knows the "coding function" —that is, the receiver is perfectly capable of generating the signal himself (Halle & Stevens, 1959, p. 2). A good strategy from the receiver would be to 'guess at the argument [i.e., the structure of the signal, DJL]...and then compare [this guess] with the signal under analysis' (ibid.). In general terms, the model internally generates patterns, and these are matched to the input signal by employing a number of rules until the final analysis 'is achieved through active internal synthesis of comparison signals' (Halle & Stevens, 1962, p. 155).

One way to output patterns would be to provide the system with a repository of structures (templates), but this would not accommodate the open-endedness of language. Rather, an AxS system must be endowed with a set of generative rules —the coding function aforementioned—, plausibly provided by the language faculty.[9] Naturally, if all the available generative rules were to be applied ab initio, the computations would take a very long time to converge to the right interpretation of the signal, but this is clearly not what happens in language comprehension. Instead, the perceptual systems must surely carry out a *preliminary analysis* in order to eliminate a large number of comparisons, a step that would make available but a small subset of possible representations. The subsequent comparisons among internally-generated representations would have to be ordered somehow (this would be the role of the *control* component) and the whole *analysis-comparison-control* sequence (what Halle & Stevens 1962 call a *strategy*) constitutes the right nexus,

[9]Recall that I have defined linguistic knowledge as a function in intension that generates sound-meaning pairs, and it is this function that must be computed in real-time comprehension.

or so I will assume here, between the parser and the grammar.

The AxS model, then, can be viewed as encompassing two main stages, the first of which involves the generation of a candidate representation of the input (the preliminary analysis), followed by a comparison of this candidate representation as it is being synthesised. According to Poeppel et al. (2008, pp. 1072–3), in the first stage of speech segmentation a primal sketch is built in terms of minimal sensory information. The second stage then carries out a number of calculations of the perceptual candidates until they are synthesized into the correct representation.[10,11] The primal sketch mediates between spectro-temporal configurations and lexical entries, and its generation is driven by the distinctive features by which words are stored in the mind/brain (ibid., p. 1072). These linguistic features are active during the preliminary analysis stage itself, but they are accessed according to a minimal amount of signal information (p. 1074). As mentioned, the information is accessed in ordered stages, yielding a hierarchical process (a hallmark of a *control*-driven operation), as larger representations are built on the basis of simpler ones. Thus, segmental and sub-segmental cues are processed within 20–80 milliseconds (ms.), while supra-segmental and syllabic (the duration, pitch and amplitude aforementioned) phenomena surface around 150–300 ms. Note that it is only once this information has been processed and adopted that phrasal segmentation can proceed at all (as in the strategy proposed by Bever, Sanz & Townsend 1998: start a new phrase at every function word).[12]

Note, then, that the solution these scholars propose is not one in which linguistic operations are literally employed in speech recognition. Rather, *some* linguistic information is used in order to derive specific hypotheses about the input signal, but the strategy in use (viz., the analysis) is perceptual in nature. It seems to me that this is a reasonable solution for the problem of how the perceptual systems manipulate information that is prima facie unrelatable to their operations, given that linguistic structure is subsumed under a completely different domain of the

[10]Halle & Stevens (1962) talk of a different type of "stage" of AxS operations, one in which different sets of generative rules apply, in conjunction with the the so-called strategies, at different times after the preliminary analysis. This is likely to be the case for many parsing phenomena, but it is not clear it applies to the (sub)operation that will engage me here.

[11]Poeppel et al. (2008) link the AxS model to a hypothesis-and-testing method, as "guesses" are initially generated and then recoded into a format that allows comparison (ibid., p. 1079). They also note its similarities to a Bayesian inference, wherein $P(H\backslash E)$ (the probability of the hypothesis given some evidence) stands for the likelihood of the analysis and $P(E\backslash H)$ (the probability of the evidence given some hypothesis) for the likelihood of the synthesis. One may well point out that anything can be modelled with a Bayesian inference, but it does not follow that this is the (cognitive) mechanism in operation.

[12]Furthermore, note that the model would not succeed in segmenting the input if it did not have access to specific linguistic information, as Yang (2004) shows in his discussion of the results reported in Saffran, Aslin & Newport (1996). There is a subtlety to be mentioned here, though. Segmental, sub-segmental and supra-segmental features are crucial in the acquisition of language, but lexical access in language comprehension operates concurrently with these sources of information; upon hearing the sound *choc*, for example, a number of lexical items are immediately retrieved, to wit: *chocoholic*, *chocolate*, ecc.

mind. This was already evident to Bever (1970) —cf. his belief that the perceptual and grammatical systems are independent (p. 312) or his statement that while linguistic structures may be reflected in speech behaviour, they do not appear in specific behavioural processes (p. 342)— but such a corollary clearly follows from the epistemological discussion advanced in chapter 1.

As Halle & Stevens (1959) anticipated, the AxS approach can be applied in many other domains, including syntactic parsing. In fact, an AxS model of parsing was already defended in G. A. Miller & Chomsky (1963), a suggestion that was perhaps based on the realisation that intensive pattern recognition can extract different types of skeletons, be words, phrases, intonational units, et alia (Bever & Poeppel, 2010, p. 177). This view was then adopted and developed in Bever (1970), expanded by Fodor et al. (1974), and further polished by Townsend & Bever (2001).[13] Bever (1970) proposed that the preliminary analysis involves the application of a Noun-Verb-Noun (NVN) template onto the input, a reflection of the statistical distribution of sentences (i.e., NVN is the most frequent word order, at least for English, as exemplified in the canonical subject-verb-object structure).[14,15] The primal sketch this stage generates is then further expanded, or indeed revised if it turns out to be mistaken, when the proposed candidates are synthesised. The latter stage involves, according to Townsend & Bever (2001), an application of the derivational rules of the grammar upon the sketch created by the first stage. The overall model, then, starts with the extraction of a skeleton (a template), and is then followed by a syntactic derivation that fills the missing parts of the scheme (i.e., the first step is matched to the second; Bever & Poeppel 2010); a syntax-last parser, to borrow Townsend & Bever's phrase.

This take on things is not so dissimilar to the model proposed by Stabler (2010): in a "first-pass", the parser attempts to conjoin the elements of the string, while the "second-pass" builds the relevant hierarchies. There seems to be much in favour of this general take on things, as it accommodates the fact that language comprehension is primarily a perceptual phenomenon with the observation that the representations the parser builds are linguistically structured, and hence generatively constructed. The AxS model captures these desiderata, given that the first component operates on the input using an intrinsically perceptual strategy —indeed, the imposition of a template— which is then analysed by staged applications of a generative grammar that is stored in the second component.

Naturally, it is a matter of research to find out what operations exactly participate in this model (including their timing and memory requirements), and a number of questions immediately arise regarding how Stabler's proposal meshes, in detail, with the AxS model. The second-pass stage clearly pertains to the sequences of

[13]G. A. Miller (1967) also discusses the AxS model to some length. Therein, he gathers that a AxS listener recognises what he hears by comparing it with some internal representation (p. 75).

[14]This frequency information is clearly coarse-grained; that is, it is the frequency of a particular structure rather than of particular words in different structures.

[15]In a related note, Ferreira & Patson (2007) propose a similar strategy: assume that the first noun phrase is an agent and the second a patient.

generative rules, but whether the conjunctive operations of the first-pass participate in the application of the NVN template or right after it is not so clear. The issue that will engage the rest of this chapter is whether complex noun phrases on either side of the NVN template result in operations that are temporally prior to the build-up of higher-order hierarchies the second-pass undertakes, and whether these operations come with their own memory load and complexity. Such a possibility suggests a modification of the NVN template into a much more basic type of structure, one that furthermore suggests a possible recursive sub-operation of the parser. The model I am envisioning would start with the perceptual strategy of the preliminary analysis (the template),[16] it would then be followed by (or is combined with) Stabler's first-pass (an operation of the parser proper), and the final analysis of the signal would be the result of the set of stages encompassing the synthesis of the initial representation (roughly similar to Stabler's second-pass of the parser).[17]

4.2 Atomic Operations: building (S)-H-C(s) structures

The AxS model encompasses a rather complex mechanism for language comprehension, but I will here exclusively focus on the preliminary analysis, in combination (if they do indeed relate) with Stabler's first-pass stage.[18] This viewpoint

[16]The template is similar to the "primal sketch" aforementioned. The latter is a rough representation that is constructed from very basic sensory information, while the template attempts to organise the input into a structural representation.

[17]To be more accurate, the whole language comprehension process should be viewed as a succession of AxS applications, starting from speech segmentation all the way to ambiguity resolution. The crucial aspect is that all these applications are structured.

[18]Keeping to the theme defended in section 1.2, I am proposing a "classical" model for language comprehension; that is, a system in which specific computational operations manipulate symbolic representations and some information (linguistic, in this case) is built-in. I will not, then, engage or discuss connectionist models, but it is worth adding some words here, given that some connectionist studies (especially, Christiansen & Chater 1999 and Christiansen & MacDonald 2009) have treated issues that are related to recursion. First of all, it is worth emphasizing that a central tenet of generative linguistics has it that language is primarily a psychological phenomenon, by which it is meant that there is a cognitive capacity devoted to it, and therefore natural languages exhibit properties that reflect basic principles of mental organisation. In this sense, humans are prepared to acquire natural languages because they are pre-programmed to attain those systems that fit mental structure; hence, the Leibnizian dictum that language is a mirror of the mind. As Christiansen & Chater (2003) show, however, some connectionist models (the ones they favour, in fact) resist the proposal that constructs such as constituency and self-embedding reflect mental structure and are therefore in-built in the linguistic system, preferring to believe instead that neural networks simply come to learn linguistic structure, and so languages are literally built from scratch via the interrelations of neural units (which might to boot include statistical processes). It is not clear to me why these scholars hold this belief, but the consequences are clear: psychological phenomena are merely emergent properties of interconnected networks of units, placing learning (conceptualised as different configurations of populations of units) at the very centre of the system. Indeed, both Christiansen & Chater (1999) and Christiansen & MacDonald (2009) feed their networks literally thousands of examples of self-embedded sentences (what they call "recursion") until they come to exhibit the behaviour of human subjects in the processing of these structures (that is, they purport to show that these networks correctly model the psycholinguistic data), even yielding new predictions. At best,

contrasts with much of the psycholinguistics literature, as most studies are centred on how the material the parser receives is added to the already-built structure (i.e., where in the tree structure it is attached, as in ambiguity resolution), a phenomenon that is plausibly the domain of higher-order operations —that is, part of the staged sequences of the synthesis component. This is, in my opinion, a bit unfortunate, given that whatever operational variables operate in the first component, their result (such as memory load) must surely be inherited by the second component, perhaps affecting the operations of the latter in intricate but as-yet-unknown ways. Thus, it seems a reasonable strategy to study the first stage as thoroughly as possible before venturing into the operations and properties of the second component. A strategy that involves, I will argue, employing much simpler structures than those usually employed in psycholinguistic studies.

Let us assume, arguendo, that the preliminary analysis really involves the application of a NVN template. Even if this were to be the case, the conjunction of elements the parser receives must necessarily be underlain by some sort of mechanism,[19] but the literature has been rather unspecific on these matters. Grodzinsky & Friederici (2006), an expository article on the neuro-imaging data regarding syntactic parsing, is quite illustrative of the overall field; therein, they divide the different stages of language comprehension in such broad terms that it is hard to evaluate the presented data;[20] to wit: local phrase structures are initially built based on lexical information, dependencies are then formed and integration finally takes place (p. 243).

The garden-path model, an initial version of which was put forward by Frazier & Fodor (1978), is a clear exception. Therein, these authors also divide the parser into two stages: the preliminary phrase packager (PPP) builds phrase structures of roughly six words (7 ± 2, to be more exact, in reference to G. A. Miller 1956), while the sentence structure supervisor (SSS), the second stage, adds higher nodes into a complete phrase structure (Frazier & Fodor, 1978, pp. 291–3). The application of the PPP recalls some of the properties of the preliminary analysis, as its operations can be insensitive to some aspects of well-formedness (ibid., p. 292). However, much like Stabler's first-pass, the PPP closes and shunts phrases as soon as these are formed (p. 298), with the result that its "viewing window" shifts throughout the sentence (p. 305). It is furthermore composed of two building operations: "minimal attachment" (MA, which incorporates a word into a structure using the fewest syntactic nodes) and "late closure" (LC, which attaches new material to the node currently being processed). The SSS, on the other hand, carries out a "reanalysis"

however, these studies achieve *weak equivalence*: they model the input-output behaviour of humans, but they do not inform us about the actual cognitive capacity at play, unless we accept their premise that grammatical knowledge reduces to processing/learning phenomena (and the latter to networks of interconnected units).

[19]Recall that a mechanism is composed of operational variables and the data structures these operate on.

[20]That is, most of the psycholinguistic data seem to inform us of the very broad effects the processing of language seems to have on the parser, but we seem to be on the dark regarding the properties and behaviour of the operations that must surely underlie the processor.

of the interpretation the PPP returns if the latter is incorrect, eventually closing the whole structure under the *Sentence* node (giving a bottom-up flavour to the overall process; pp. 314–7). In a nutshell, the PPP creates local parses and interpretations that are then put together by the SSS.

In further developments of this theory (e.g., Frazier & Clifton Jr. 1996), the parser is augmented with an "active gap filler" in order to reconstruct displacement chains (the combination of a moved element and its trace). The resultant language comprehension system, then, starts with syntactic processing and this stage is then followed by the operations of a thematic processor. Regarding the operations of the PPP (MA and LC), these now apply to primary syntactic relations only —that is, an argument-over-adjuncts take on things— which is meant to capture some cross-linguistic differences in the application of MA and LC (see Frazier & Clifton Jr. 1996 for details).

The garden-path has come to epitomise an instance of a so-called modular, autonomous model, perhaps a corollary of an old-held belief that the organisation of grammatical sub-systems (as in GB Theory) may map onto the same number of processing modules (Frazier, 1988). A model of parsing such as this is usually contrasted with interactionist approaches that defend the idea that the parser has access to diverse bodies of lexically-stored information (syntactic, lexical, semantic, contextual, ecc.) at any stage of the comprehension process (see, for example, MacDonald, Pearlmutter & Seidenberg 1994).[21] An interactionist model is nonetheless able to explain a diverse set of experimental results (even the purely structural), as the theorist has great leverage in postulating how the different constraints are "ranked", not only ab initio, but during different stages of processing. This perhaps offers too libertine a check on theory-construction and interactionist scholars have certainly not shied away from explaining all sorts of experimental results by fidgeting with the ever-changing rankings —a strange property to ascribe the parser, certainly. Nevertheless, these same scholars have been very unspecific regarding the underlying mechanisms and operations that are surely at play in language comprehension.[22]

Be that as it may, and whilst it is certainly true that some ambiguity resolution tasks do in fact employ diverse bodies of information, it is also the case, as Pickering & van Gompel (2006) and van Gompel & Pickering (2009) show, that semantics seems to have a limited effect on syntax, especially on unambiguous

[21]Despite the modular tag, models such as the garden-path should not be too closely related to Fodor's modularity thesis; unfortunately too common a (mis)identification, in my opinion. Fodor (1983) does allow (see p. 78 and 135) what Steedman (1994) calls weak interactionism, which certainly covers the effect lexically-stored (that is, *linguistic*) information may have on syntactic interpretation; it is the influence of other parts of cognition that are blocked, according to Fodor.

[22]Gibson (1998) is an exception. According to this model, structure building involves looking up the lexical content of the word being processed, drawing predictions based on this information and then matching the predictions to the following words. The overall model is based on an analysis of the structural costs incurred in the processing of complex structures, as manifested in two components: the memory cost of storing a partial input sentence and the integration cost of incorporating new material. I will come back to this model presently.

and simple sentences (which is the type we will employ here). Furthermore, electroencephalography techniques employed in the measurement of electrophysiological responses to external stimuli (i.e., event-related potentials) have been able to identify a number of components that seem devoted to syntactic operations, such as ELAN (early left anterior negativity, which appears at around 150–200 ms. after the presence of phrase structure violations) and LAN (left anterior negativity, which surfaces at 400 ms. in relation to subject-verb agreement). These components precede both the N400 so characteristic of semantic violations and the P600 operative in reanalysis (as in garden-path sentences; see supra), providing some evidence for the existence of the syntactic properties I will be focusing here: those that play a role in an early stage of language comprehension.[23,24]

The overall approach defended here follows the spirit of a remark of Turing's in his epoch-making (1936) paper. In the context of discussing the structural properties of his abstract machine, Turing invites us to 'imagine the operations performed by the computer to be split up into "simple operations" which are so elementary that it is not easy to imagine them further divided' (ibid., p. 136). As applied to parsing, one could protest that it is hard to see any further sub-division to either the NVN template or Stabler's conjunction strategy. On the contrary, we propose that such a further sub-division is not only possible, but probable.

As established in early chapters, one of the most robust results of generative grammar —from a purely structural point of view, at least— has been the discovery that all phrases respect a general hierarchical structure, an asymmetric (S)-H-C(s) geometry (SHC henceforth). It is therefore reasonable to suggest that at some level of abstraction the parser must recover/build a (macro) SHC structure and its internal phrases, an operation that is much more basic and atomic than either the conjunction of NVN sequences or the applications of the PPP component (not to mention whatever mechanisms effect attaching and ambiguity resolution). Note, furthermore, that a SHC operation of this sort would introduce hierarchical structure (of operations, at the very least) at this very early stage, as further, internal SHC phrases may appear in either S or C. As the parser would build these phrases in succession —a recurrence— the Church-Turing Thesis results: the underlying computations may proceed recursively or iteratively; it is this very possibility that this chapter attempts to explore.

Note that building a SHC structure is not an automatic effect of carrying out phrase structure rules, as they just do not create this geometry.[25] It is also quite clear that it cannot be the case that SHCs are constructed by higher-order operations as if it were some sort of side-effect, as this would assume that SHC-building comes

[23]It is important not to confuse these event-related potentials for instances of "syntactic" operations. Instead, they should be taken to indicate those structural properties the perceptual system is sensitive to at precise temporal points.

[24]This chapter will focus on comprehension only, but such a sequential approach seems to apply to linguistic production too; see Sahin et al. (2009) for a recent study.

[25]Furthermore, the possibility of employing phrase structure rules in such a direct manner is nevertheless disregarded for independent reasons, as argued above.

for free, without exerting any special memory load on the parser; this, however, needs to be demonstrated, not presumed. Rather, I aim to investigate how the perceptual systems deal with correctly allocating each incoming element to the right slot of the SHC tree structure, perhaps a sub-operation of the PPP. Note, however, that the allocation of the right SHC position does not directly bear on how the parser resolves ambiguities; that is, a given noun phrase has the same position in the macro SHC structure regardless of what it in fact modifies (that is, whatever follows the verb is its C in the SHC structure, it clearly does not matter whether an internal noun phrase modifies the verb or not; a *complement* in a SHC structure is defined in terms of its hierarchical position towards the *head* —it is its sister).

As a first approximation, consider some of the general features of this point of view. The H position will always be occupied by a terminal element, while both S and C may be empty or occupied by other SHCs; clearly, not all SHCs need to be fully specified, and in fact most SHCs will exhibit a "reduced" version. Furthermore, there is a distinction between a macro SHC (a subject-verb-object sequence, basically) and the internal instantiations of this general scheme. A sentence, then, is an asymmetric CP structure composed of other asymmetric structures (NPs, VPs, ecc.). In principle, this constitutes a complex problem (building a CP) that is reducible to simpler instances of the same problem (building NPs, VPs, PPs, in succession), making a recursive solution perfectly applicable.[26]

If the parser applies recursively in the construction of SHC structures, reflexive calls are expected to apply at the edges: at either S or C, the loci of possible deferred operations. These specific locations, then, ought to exhibit a memory load that would at least be greater than the memory load found at those locations where there *cannot* be any deferred operations; a fortiori, a recursive process postulates greater memory loads than a non-recursive implementation (as discussed in section 1.1). There are two relevant comparisons then: between different positions of a sentence and between differing processes.

The proposal advanced here, then, involves the substitution of the NVN template and the operations that link up its elements for a SHC template with its corresponding conjunctive operations. This is a reasonable starting-point given the level of analysis I am focusing on, and the materials and experimental conditions to be described in the next section keep to this theme. Note that I am assuming that building a structure involves a mechanism (a combination of elements and operations), and that this mechanism exerts a cost in working memory. In contrast, one could perhaps propose that a probabilistic operation could "recover" SHCs just as well. However, I take it for granted that predicting S-S-S-S-H-C sequences is not quite the same as *building* a [S[S[S[S[HC]]]]] structure; a fortiori, given that it is certainly true that syntactic parsing engages working memory, it is not at all

[26]In this description, I am assuming that the parser operates over a derived tree for expository reasons. The actual process is more accurately described as an assembling of an internal SHC representation that mimics the operations of *first-merge*, *second-merge*, ecc. underlying the structure of a derivation.

clear how the calculation of probabilities translates into such tangible properties as cognitive load.

Let us describe the mechanism I am envisioning. By analogy with a *search* procedure from computer science, recursive parsing would be something like this (note the two recursive calls; note also that in my nomenclature terminals are processed but structures are built; the former is explicitly mentioned only once below, but it is assumed to apply to every terminal):

Task: build SHC

- build S, then build [H-C]
- build S
  - if S is a terminal, build [H-C]
  - if S is not a terminal, build internal SHC
    (a deferred operation starts ("push-down"), followed by a "pop-up", i.e. the operation moves up a level)
- build [H-C]
  - process H, then build C
  - if C is a terminal, end task.
  - if C is not a terminal, build internal SHC
    (push-down followed by a pop-up)

Naturally, a recursive implementation can be contrasted with a non-recursive (iterative) process, but it would be too daring at this point to outline a non-recursive implementation, given that it is such an unconstrained exercise —one could, in fact, imagine an unlimited number of possibilities here. Suffice here to state that an iterative process would not in principle exert a great memory load at those junctures in which a recursive operation may be operative. Moreover, if the process is iterative and there are no deferred operations, there should not be any significant differences in memory load within different positions of a sentence (barring a proviso I will treat below). That is, there are different predictions in memory load regarding what recursive and iterative processes postulate at specific points in a sentence, in addition to how memory load changes *during* the processing of a sentence. In general, this is because the state (and therefore the memory status) of an iterative implementation is exhausted, at any given time of the process, by the building operation and the variables it manipulates, while a recursive process involves a building operation, a set of the variables and the deferred operations that are being stored in working memory (resulting in a hierarchy of sub-operations).

The method employed here to figure out which implementation is in fact operative will consist in probing the memory load of the processor by constructing an experiment in which concurrent tasks are at play. Parsing a sentence is naturally the primary task of the processor, but an experimental situation can be devised in which subjects have to pay attention to a secondary task, creating a conflict in memory resources. The assumption here is that performance on the secondary task will correlate with the complexity involved in the primary task.

Before proceeding, it is important to note that I am not suggesting a top-down process; that is, the discussion so far conducted should not be construed as an outline of a process in which the parser "assumes" there is a macro SHC tree structure that it then proceeds to expand. Rather, in my view the role of the parser is to organise the input it receives into an SHC configuration that is compatible with the structure of the input. That is, the parser applies a template, it does not build a tree structure. If this so, top-down, bottom-up and even left-corner approaches to tree construction are simply inapplicable in the study of parsing. Following from this, there will be many situations in which the parser will, of necessity, reorganise the SHC conglomerate is assembling. For example, it is not possible for the parser to open a complex internal SHC node upon processing the S of a macro SHC, as it just cannot anticipate that this S is indeed complex.

In order to clarify, consider the operations of the parser as it processes a structure of the following type: $[_S[SHC][H-C]]$, where the first H is the noun of a complex subject noun phrase and the second H is the main predicate of the overall sentence. Clearly, the parser would combine the first three elements into a SHC compound as it receives them, but it would only be able to interpret this SHC as the complex S of a greater SHC structure once it encounters the verb. That is, it is only at the moment the parser starts processing the verb that the internal representation is constructing can be reorganised into a more adequate and intricate representation of the input. If this is the case, then the distinction between a recursive and an iterative process lies on whether the former reorganises the SHC configuration into a structure that contains a deferred operation. In any case, the memory predictions I will postulate at specific junctures remain the same.

Naturally, this is a conscious attempt at determining the appropriate representation of the input and output, and the implementation that relates them —an explicit study of Marr's algorithmic level. We do know a fair amount about the memory capacity and general cognitive architecture of human beings, and manipulating the memory load is a reasonable strategy to employ.[27]

This situation is not so dissimilar to a scenario in which a computer scientist is observing a given machine calculating the factorial of a number. It is certainly impossible to establish what method the computer is employing (recursive or iterative) by just looking at it (that is, without access to the underlying code). We would need to know more about its memory capacity, the form of the computational scheme it is employing, and perhaps many other factors. Or in the words

[27]There are many textbooks in cognitive psychology that do a good job of describing basic properties of mental architecture and memory capacity. I will not cite any of these here; instead, I will reference specific psycholinguistic data as they apply to this work.

of Fitch (2010), we would have to 'prob[e] registers and logic gates with measurement devices' (p. 78) in order to discover the underlying function being executed. This is exactly the approach that is undertaken here.[28]

4.3 Experimental Data

In order to manipulate the memory load of the parser, we employed what is known within the psycholinguistics literature as the "click" paradigm, an experimental technique first used in syntactic parsing studies by Fodor & Bever (1965). As these authors point out, the click paradigm was used by phoneticians at the time as a means of probing perceptual consistency (ibid., p. 415), a phenomenon in which some processing units can be found to resist interruption (and hence known as "perceptual units"). Now, whilst the issue of whether any such perceptual units could so be identified within syntactic parsing was one which would engage some scholars, the click paradigm was also perceived to be of relevance for the study of parsing for much broader reasons. Specifically, it was felt to provide a useful means for investigating whether the clausal hierarchies postulated by generative grammarians reflected how people in fact conceptualise them. That is, this experimental technique was regarded as an appropriate tool for probing the extent to which the segmentation of the parser matched the classifications of sentences proposed by the linguist.

The click paradigm itself consists in superimposing a short, extraneous sound —a click, a tone, or else— over some linguistic material, which is then played to subjects over headphones. In the version Fodor & Bever (1965) ran, the subjects would be asked, first, to write down the piece of auditory material they had just heard (in this case, a sentence) and then mark where they thought the click was placed. It was not a matter of probing subjects' accuracy in the task —they are indeed very inaccurate— it was instead an endeavour to map the errors subjects make, so that a comparison could be drawn between the objective position of the click and the position in which subjects subjectively perceive it.

Fodor & Bever (1965) reported that even though subjects had a tendency to perceive a click before its objective position (a left bias, which was also reported by the developers of the click paradigm; ibid., p. 419), the overall majority of clicks, as subjectively perceived, were displaced towards clausal boundaries. Thus, in a sentence such as *that he was happy was evident from the way he smiled*, the click was perceived to be between the main and the subordinate clause; that is, between *happy* and the *was* to its right. To be sure, a biclausal sentence of these characteristics exhibits a certain complexity, as it contains various internal phrases and boundaries. Nevertheless, the results reported in Garrett, Bever & Fodor (1966) suggest that clicks only ever migrate to the deepest constituent boundary —the

[28]The three experiments reported in the next section were carried out in collaboration with José E. García-Albea and Marc Guasch. We also benefited from the input of the members of the Psycholinguistics Research Group at the University Rovira i Virgili, where these experiments took place.

frontier between clauses. Similarly, Bever, Lackner & Kirk (1969) concluded that within-clause boundaries do not appear to affect segmentation strategies.[29] These results were taken as evidence that the clause is an important unit of syntactic processing, perhaps constituting a perceptual unit (see Fodor et al. 1974 for details).[30] Furthermore, the clause-by-clause process these scholars believed to have unearthed appeared to be solely the effect of syntactic properties,[31] as other factors were controlled for and did not seem to affect the results.[32]

The latter point was contested by Reber & Anderson (1970), though; by employing much simpler sentences (monoclausals such as *open roadside markets display tasty products*), they found evidence that a) a right bias was actually operative (which they claimed to be present in the results of the aforementioned papers as well), and b) extra-linguistic factors were clearly responsible for the errors subjects made. They also reported a tendency for subjects to mislocate some clicks to the boundary between the subject and the verb, which might suggest that this break is also important for the processor, something that was explicitly denied in Fodor et al. (1974, p. 336).[33]

These classic publications employed what is now known as the location version of the click paradigm, an offline experimental technique that is currently largely out of favour. Abrams & Bever (1969) developed a detection version, an online tech-

[29]The latter study also employed complex biclausal sentences, such as *when he stood up, my son's book fell from the low and small table*.

[30]It is worth pointing out that by "clause" these scholars had what used to be called the "deep structure" of a sentence in mind; namely, the underlying argument structure (the predicate with its arguments). Thus, it was the deep structure of a sentence that Fodor et al. postulated to be the perceptual unit of syntactic parsing; in retrospect, perhaps too daring a proposal. Carroll & Tanenhaus (1978) advanced a modification in terms of what they called a "functional clause", which is much closer to the surface representation. Being closer to the sentence that is explicitly perceived, Carroll & Tanenhaus (1978) were able to draw a distinction between complete and incomplete clauses, the former being a functional clause in which all the arguments are present, which is much more suitable for segmentation, according to their results.

[31]Fodor et al. (1974) described the clause-by-clause strategy in such a way as to suggest that semantic and pragmatic operations would only become operative once a clause had been completed, a view that was vigorously contested by, e.g., Tyler & Marslen-Wilson (1977). According to the data reported by the latter, the processor attempts to provide a meaningful representation to the material it receives *as soon as it receives it*, a phenomenon that has come to be known as *incremental* parsing. Because of its apparent incompatibility with incrementality, the clause-by-clause strategy is sometimes described with derision in some textbooks (e.g., Harley 2001, p. 228), but incremental parsing and the importance of the clause in processing are entirely compatible phenomena once we abandon the view that semantics and pragmatics await the completion of a clause. So-called "wrap-up" effects are very robust in parsing studies, and they are plausibly the result of the exigencies of a clause (see infra, though). Further, it is only the clause that codifies the underlying "proposition" of a sentence, and human thought does appear to be largely propositional (see the Introduction supra for some comments regarding this point).

[32]Some of these include pitch, intonation, response bias, the effect of writing down the sentence, the effect of extracting the sentence from memory before writing it down, ecc. (Garrett et al., 1966; Bever, 1973)

[33]Chapin, Smith & Abrahamson (1972) reported a similar result with biclausal sentences, but there are some problems with the materials they employed (see Fodor et al. 1974, pp. 336 et seq. for relevant discussion).

nique that is much more reliable —and the one employed here.[34] In this version of the paradigm, subjects are required to press (or depress) a button as soon as they hear the click, and so the analysis centres on the fluctuations in reaction times. The idea is that processing a sentence and monitoring a click compete for attentional resources, and so reaction times ought to be greater at those junctures of a sentence that require more working memory; a correlation between reaction times (RTs) and complexity, as it were. Keeping to biclausal sentences, Abrams & Bever (1969) found that clicks before the major break were reacted to more slowly than clicks *at* the major break or just *after* it.[35] Similarly, Holmes & Forster (1970) found that RTs in the first half of a biclausal sentence are greater than in the second half. However, the latter also found that subjects reacted faster when the click was placed at a major boundary, which is not entirely compatible with the data reported in Abrams & Bever (1969); I will come back to this eventually.

There are two other publications from the 1970s that are relevant for our investigation. Firstly, Bond (1972) found evidence that suprasegmental factors influence subjects' performance; namely: a) subjects react faster when the click is placed on a vowel that is unstressed, and b) intonational phrases appear to drive the segmentation process to a significant extent. According to this author, then, the first step in speech perception involves segmenting the string into phonologically-defined units. Green (1977), on the other hand, demonstrated that performance is greatly influenced by the nature of the task that subjects have to carry out. The experiment he designed contained a "continuation" condition in which the presented material would be stopped right after the click appeared and the subjects reacted to it, and subjects would subsequently be required to complete the sentence themselves. RTs in this task turned out to be much higher than in a "memorisation" condition that merely required participants to recall the segment that had just been presented.[36]

Surprisingly, the click-detection has not been employed as much as it perhaps deserves, given its sensitivity to the different cognitive loads the parser goes through within and between clauses in complex sentences. After Flores d'Arcais (1978) successfully used it to show that main clauses are usually easier to process than subordinates (and that the main/subordinate order exerts less memory resources than the subordinate/main order), the 1980s and 90s hardly exhibit any other study employing this technique. It is not surprising, then, that Cohen & Mehler (1996) considered their work a "revisit" to the paradigm when they reported that RTs to clicks at the boundary of reversible object relatives were greater

[34]Furthermore, as Abrams & Bever (1969) plausibly advanced, the click-location and the click-detection tasks do not appear to share the same operational mechanisms.

[35]They employed sentences like *in addition to his wives, the prince brought the court's only dwarf.*

[36]It should also be pointed out that Green employed monoclausal sentences that codified three propositions in an attempt to figure out if processing involves the combination of all underlying propositions into one common representation or the construction of each individual proposition into separate representations. As an illustration, he employed sentences like *the sleek sinewy leopard attacked the woman*, which is composed of the following propositions: the leopard is sleek, the leopard is sinewy and the leopard attacked the woman. The evidence suggests, to Green at least, that one single proposition is constructed; I will come back to this.

than at structurally-identical subject relatives (or in other positions of a normal object relative; they also reported similar results with semantically reversible and irreversible sentences). Recently, though, the click-detection paradigm has been usefully employed in a word segmentation study (Gómez, Bion & Mehler, 2011), and it is hoped that the results we report here are further evidence for its usefulness in the study of language comprehension.

As stated earlier, we decided to employ much simpler sentences than is usually the case in the literature. This is a rather natural choice considering that we aimed to study the most basic/foundational operations of the parser. Naturally, such core operations will be as operative in the processing of simple sentences as they are in parsing complex structures. Still, it is probable that the nature and behaviour of these atomic operations will be more easily discernible in simple sentences, given that various other factors are known to be present in the processing of complex structures, therefore obscuring the presence and effect of the underlying operations. This is not to say that the second-stage of the parser we identified above does not operate during the processing of simple sentences, but it is certainly true that an SHC (or NVN) template and a set of conjunctive operations takes you very far into the right interpretation of such simple structures. Relatedly, processing a complex sentence may well stretch the memory resources of the parser to the limits of its capacity, and this is unlikely to be the case in the processing of simple sentences. At the same time, any effect we might unearth with simple sentences will plausibly be part of the processing of complex structures, but in combination with other operations and properties. Thus, it seems like a reasonable strategy to study language comprehension from scratch, as it were.

Monoclausal, subject-verb-object Spanish sentences were constructed for the purposes of this investigation. Starting from a matrix *proposition* —that is, a predicate and its arguments— two different types of sentences were created. *Type A* sentences exhibited a complex subject but a simple object, while the reverse was the case for *Type B* phrases (Type A and B constitute two experimental conditions). By a complex subject or object is meant a noun phrase (composed of a determiner and a noun) which is modified by either another noun phrase (also composed of a determiner and a noun, but introduced by a preposition) or by a long adjective (that is, more than two-syllables long). A simple subject or object, on the other hand, would simply be composed of a determiner and a noun, the latter sometimes modified by a short adjective (this is usually the case for the object position, though; I will come back to the reason why this is so presently).

Take the proposition

(4.1) preparar (discurso, candidato)

where *preparar* (to prepare) is the predicate, *discurso* (a speech) is the object and *candidato* (the candidate) is the subject. Naturally, a simple sentence like

(4.2) El candidato se preparó un discurso.
'The candidate prepared a speech'.

could easily be analysed with an NVN template, but our intention here is to construct sentences with slightly more complex Ns on either side of the verb, thereby probing for possible recursive sub-routines. If these sub-routines turn out to be operative, we would have to conclude that the NVN template is too rough a pattern, and therefore worth replacing with a macro SHC template, including the application of internal SHC templates. Consider the sentences below, which exemplify *Type A* and *B*, respectively.

(4.3) El candidato del partido se preparó el próximo discurso.
'The party's candidate prepared his next speech'.

(4.4) El candidato ha preparado un discurso sobre la sanidad.
'The candidate has prepared a speech about the health service'.

While both sentences are macro SHC structures,[37] they exhibit rather different internal SHC configurations. Thus, *Type A* sentences are [NP-[PP-NP]-[VP-NP]] sequences, while *Type B* phrases exhibit a different form, namely [NP-[VP-NP-[PP-NP]]]. As mentioned earlier, there are further SHCs to be constructed in these two types of sentences, but at different locations: either on the left-hand side of the VP (in Type A sentences) or on its right-hand side (for Type B). Clearly, if there is a specific mental effort involved in the construction of further SHCs in the critical areas I have just identified, its properties may well be beyond the scope of what a NVN strategy may be able to scrutinise.

At a certain level of abstraction, of course, any string is but a succession of heads. Thus, the phrase *el candidato del partido se preparó el próximo discurso* is ultimately just a sequence of the following type (note that I take *se* not to be an independent head, but part of the verb; nothing results from this particular choice):

(4.5) $[head[head[head[head]]]][head[head[head[head]]]]$

In another sense, however, some of these elements stand in a head-complement configuration within the macro SHC structure. Thus, *del* (viz., de+el) is the head of the prepositional phrase that introduces the noun phrase *el partido*, but the whole structure (i.e., *del partido*) is the complement of *el candidato* (and similarly for *sobre la sanidad* and *discurso* in Type B sentences). At yet another structural level, the whole *el candidato del partido*, itself a complex SHC, stands in a specifier-head relation with the rest of the sentence; that is, *el candidato del partido* is the S of the macro SHC (and similarly for *el candidato* in Type B sentences).[38]

Most lexical items, therefore, have various structural roles to play in the greater hierarchy of elements that a sentence is, and it is imperative to establish what roles

[37]In a somewhat simplified description, the subject is in the specifier position, the verb is the head of the overall structure, and the object stands as its complement.

[38]When I talk of specifier-head relations, I have Brody's analysis in mind (see p. 66 supra). That is, syntactic objects are really HC structures, but they can stand in specifier-head relations to each other; hence, I will keep employing the term SHC to capture these two points.

precisely will be operative in the parser. Regarding the roles and sentences that interest us here, the parser will only have to construct internal SHCs once it reaches the first preposition of our materials, as shown below (note that this applies to both Type A and B):

(4.6) El candidato del...

(4.7) El candidato ha preparado un discurso sobre...

It is at these locations, then, where the "push-down" operation of a recursive sub-routine would apply. That is, in the process of constructing a complex SHC, the parser "moves down" a level in order to build the internal SHC, an operation that must be completed if the whole SHC is to be successfully assembled. Note that while the recursive sub-routine is in operation, the parser is exclusively focused on building the internal SHC phrase (that is, this is the only operation in course), while the uncompleted material is being kept in (some part of working) memory. Once the internal SHC has been built, the parser closes it and "moves up" back a level in order to complete the entire SHC; the latter operation is what computer scientists call the "pop-up". If parsing really does proceed recursively, then, it is these push-down/pop-up chains —chains of deferred operations— that ought to result in a significant load on working memory.[39,40]

So far, then, we have identified two critical boundaries (one internal to the subject and one internal to the object), but the sentences we are manipulating here exhibit other breaks that might affect segmentation.[41] There are two other frontiers that might be of relevance here: the boundary between subject and verb and the verb-object juncture. According to the macro SHC structure, there is certainly an asymmetry between the subject and the verb-object compound, but it is not clear that this is directly related to a possible recursive process, as such a geometry is a general feature of linguistic structure and therefore likely to be a factor in both recursive and iterative procedures. Nevertheless, the subject-verb boundary in Type A sentences coincides with the end of the postulated recursive operation, and so this break may be the locus of various operations; to wit: a pop-up, a wrap-up (i.e., putting everything together; or branching off the different open nodes), the very transition from S to HC, and perhaps others.[42] Crucially, the S-HC boundary in

[39]To be more accurate, and as stated in the previous section, this experimental work is meant to probe the memory load involved in reorganising the internally-built structure in order to see if these reorganisations suggest deferred operations. For expository reasons, however, I will keep using the more straightforward language of the computer sciences.

[40]If interested, Hofstadter (1979) explains where these computer terms come from —apparently, Anglo-Saxon university cafeterias provide more than just ghastly coffee served in plastic cups.

[41]Recall that click location and click detection are not underlain by the same mechanisms. Consequently, Bever et al. (1969) may have been too hasty in concluding that internal boundaries are not operative in segmentation, given the offline character of the click-location task they employed. In any case, our experiments were meant to probe this too.

[42]It would be a mistake, however, to postulate that this frontier witnesses an accumulation of operations, something like a "push-down+pop-up+wrap-up+S-HC transition" sort of chain. If the

Type B sentences does not exhibit such a population of operations, and thus a direct comparison can be drawn. Regarding the transition from the verb to the object (the SH-C boundary), this is also postulated to remain constant in both recursive and iterative processes, but in this case there cannot be any confusion with other operations.

Further, the present work was also intended as an exploratory study of how syntactic phrases are in general assembled and it was therefore decided that clicks ought to be introduced at all principal boundaries. This allows us to draw a direct comparison between Type A and B sentences purely in terms of length; that is, the RTs to a click in a Type A sentence can be compared to the RTs to a click at the same location (measured in syllables) in a Type B sentence. Such comparisons can be very illustrative, given that any two locations across sentence type will be mired in differing operations.

Consider, then, the three different boundaries wherein clicks were to be placed (marked with the | symbol):

(4.8) El candidato | del partido | se preparó | el próximo discurso. (Type A)

(4.9) El candidato | ha preparado | un discurso | sobre la sanidad. (Type B)

According to the discussion so far advanced, the following considerations and predictions follow. In view of past results with the click-detection paradigm, RTs should be slower towards the end of the clause, which in this case coincides with the end of the sentence itself. This is a sensible prediction indeed, as the end of a sentence involves a "wrapping-up" (Just, Carpenter & Woolley, 1982); or in terms of Frazier & Fodor (1978), the closing off of the various open nodes the parser opens up during processing. Also, it is towards the end of a clause that the underlying proposition can be assembled, and this is also likely to result in significant memory load.

Regarding the fluctuations in RTs within sentence type, we have two different progressions to track if the processor applies recursively. In the case of Type A sentences, the recursive operation would apply between the first and second boundaries, and so RTs to a click on the second border ought to be higher than at either the first boundary or the third.[43] In the case of Type B sentences, though, the first boundary is a S-HC frontier and therefore RTs here should be higher than at the next boundary (a SH-C juncture) but lower than at the third frontier, given that the latter marks the beginning of a possible recursive operation (i.e., the push-down).[44]

demise of the DTC showed anything is that there is no reason to believe that rule application involves a linear increase in complexity.

[43]This is supposed to be the case independently of the fact that the second boundary is a S-HC break. Recall, nevertheless, that the third boundary is a SH-C frontier, which ought to be less significant than S-HC

[44]Furthermore, the last boundary is closer to the end of the sentence (where pop-up and wrap-up operations would be operative).

Comparisons across sentence type are also highly relevant. Even though by the time the processor reaches the first frontier it has seen the same material in both sentence types (viz., *el candidato)*, in Type B this position is a S-HC boundary and therefore RTs here should be greater than at the same position in Type A (where the processor is yet to reach the S-HC frontier). This last prediction is perhaps a bit tricky, though, as we are comparing the load the S-HC frontier exerts with that of the beginning of a recursive operation (the push-down). In this sense, the terms for comparison are rather uncertain —to be sure, a sign of the exploratory nature of this experiment. The state at the second position should be clearer, though, as for Type A sentences this would involve the end of a recursive operation at the S-HC border, and higher RTs than at the corresponding second position in Type B sentences should therefore be expected. Naturally, the reverse ought to be the case regarding the third boundary.

If the process proceeds iteratively, however, the state of the parser at any stage ought to be exhausted by the assembling operation it carries out plus the variables it operates upon in combination with whatever memory cost the S-HC frontier and the wrap-up operation produce. That is, there should not be such noticeable differences in RTs within a sentence apart from those stemming from general features of linguistic geometry. A fortiori, no differences in terms of chains of deferred operations, push-downs or pop-ups. A much more regular pattern in RTs for both sentences is therefore predicted if the process is iterative.

Naturally, the experiments and materials were designed so that only these considerations in fact apply. As such, all sentences were composed of high-frequency words, there were no structural ambiguities and length was controlled for so that RTs to clicks placed in the same position across sentence type could be compared. The second point was relevant for Type B sentences especially, as it was crucial that the preposition introducing the modifying noun phrase did not create a local ambiguity; that is, we needed to construct sentences in which the preposition could not be interpreted as introducing a noun phrase that could modify either the verb or the preceding noun phrase. This was achieved by making sure that the elimination of the principal object noun phrase rendered the sentence ungrammatical. Thus, if we eliminate *un discurso* from the example above, the resultant sentence (*el candidato ha preparado sobre la sanidad*) is not grammatical, as the phrase *sobre la sanidad* cannot modify the verb *ha preparado* —that is, the preposition *sobre* can only point to the preceding noun phrase.

I now turn to the description of the three experiments we carried out. All three manipulated the same critical items, but these items were presented to different sets of subjects in different circumstances. In this sense, the experiments clearly varied in design complexity, which brought about an interesting set of data.

4.3.1 Experiment 1

Method

Materials. Two variants of monoclausal, active, declarative, subject-verb-object sentences were constructed for 60 *propositions*. Type A sentences exhibited an NP-NP(or AdjP)-VP-NP pattern in which a) the subject was a complex structure composed of a noun phrase (determiner + noun) modified by either another noun phrase (always introduced by a preposition) or a long adjective, b) the verb was either in perfect tense (with an auxiliary such as *ha*) or in reflexive form (and therefore always headed by the particle *se*) and c) the object was a simple noun phrase sometimes modified by a short adjective. Type B sentences manifested a NP-VP-NP-NP form in which a) the subject was a simple determiner-noun complex, b) the verb was of the same type as for Type A and c) the object was a complex noun phrase modified by another noun phrase that was always introduced by a preposition. Type A and B are the structural conditions of the experiment. All sentences were unambiguous, composed of high-frequency words (according to the corpora in Sebastián-Gallés et al. 2000 and Almela et al. 2005) and with a total length of 20 syllables. In order to keep the length constant, it was sometimes necessary to either change the tense of the main verb (some forms are significantly longer than others in Spanish) or add a short adjective in the simple versions of subjects and objects (this short adjective always precedes the noun and therefore cannot be identified as its C). This was particularly laborious in relation to the three critical boundaries of the experiment, as length had to be equalised across sentence type. On average, the first boundary appeared after 4.2 syllables (SD: 0.43), the second frontier after 9.1 (SD: 0.62) and the last juncture after 13.9 (SD: 0.72); there were only three values for each interval: 3-4-5, 8-9-10 and 13-14-15, respectively. The sentences were recorded in stereo with a normal but subdued intonation by a male native speaker of the Spanish language using the Praat software on a Windows-operated computer. Special care was employed so that the intonational break between the subject and verb was not too marked, thereby neutralising the phonological factors reported in Bond (1972).[45] The sentences were subsequently analysed with Praat in order to identify and eliminate any undesired noise, and to calculate amplitude, intensity and pitch values (average, minima and maxima). Three click positions per sentence were established, one for each of the boundaries. These are the three positional conditions of the experiment (1-2-3). It was decided that the clicks would be placed on the vowel of the second syllable following the relevant boundary, so that the processor could use the first syllable (usually a preposition, the beginning of a preposition, or the form heading the verb) to "disambiguate" the phrase it was processing. This was deemed to be necessary so that it was clear that the click was placed at a location where the parser had been able to work out that the preced-

[45]Prosody has been argued to play a rather central role in sentence comprehension, but this is usually framed in terms of syntactic attachment decisions, focus interpretation or the availability of contextual information in the resolution of lexical ambiguity (Schafer, 1997); that is, in terms of higher-order operations, which do not apply to the materials we are manipulating here.

ing phrase had been completed and therefore could be closed off. The software Cool Edit Pro was employed to generate and superimpose tones with a frequency of 1000 hertz, a duration of 25 milliseconds, and a peak amplitude similar to that of the most intense sound of the materials (around 80 decibels). Every sentence had one click only, and in order to make sure that every item went through every condition, three different copies of each experimental item were created, totalling 360 experimental sentences. A further 12 practice items were created, two items per experimental condition (see Appendix A for a full list of materials).

Procedure. The design of the experiment was a 2 (type of sentence factor) by 3 (click position factor) within-subjects, within-items factorial, and therefore six versions of the task were created, corresponding to six experimental groups. Every version of the experiment was composed of 60 experimental items, with a distribution of 10 items per experimental condition. Each version was arranged from a pool of 360 experimental items according to a Latin square (blocking) method. The items were randomised within each block (a block per experimental condition) and between blocks. Subjects were randomly assigned to each experimental group. Consequently, every subject underwent every condition, and every item also underwent every condition. The experiment was designed with the DMDX software (K. I. Forster & Forster, 2003) and administered in a sound-proof laboratory with low to normal illumination in which a maximum of four subjects at a time would be tested. After being instructed to switch off mobile phones and to leave all personal belongings on an adjacent table, subjects were seated in front of a table containing a computer screen, a keyboard, a keypad and a set of headphones. A list of instructions was placed on top of the keyboard for subjects to read before the start of the practice session. Once they had finished reading the instructions, the experimenter explained the task and answered any possible questions. As soon as this was done, subjects were asked to put on the headphones in order to carry out a practice session while the experimenter was still in the room. The sentences were presented over the headphones binaurally and participants were instructed to hold the keypad with their dominant hand in order to press a button as soon as they heard the tone. They were told to be as quick as possible, but to avoid guessing (low and high cut-off points were set up). Once a sentence had finished, an instruction on the computer screen stated that the next sentence would be presented upon pressing the space bar, giving subjects control over the rate at which the sentences were presented. Each sentence would be played 500 milliseconds after pressing the space bar and would not last more than 5 seconds. This set-up ensured that subjects had the dominant hand on the keypad and the other on the keyboard. Once the practice session was over, the experimenter clarified any final questions before leaving the experimental room. An experimental session of 60 items started immediately after and the DMDX software was used to measure and record reaction times. The whole session lasted around 20 minutes. The experiment was administered at the psychology department of the University Rovira i Virgili, of Tarragona, in November 2010.

Participants. 88 year-2 psychology students (20 male, 68 female) participated in the experiment. They were awarded a 0.25 credit (on a scale of 0 to 10) towards the final grade on a core-module course for their participation. The mean age was 20 years, all participants had normal or corrected-to-normal sight and not known hearing impairment. All were native speakers of Spanish.

Results

The responses of 8 subjects had to be eliminated for a variety of reasons. Six of these were due to technical problems with the coding of the computer program and/or the equipment, while the other two did not meet reasonable expectations regarding average performance. One of these two subjects failed to record a single response (most likely because the task was carried out incorrectly), while the responses of the other suggests that this subject was not paying attention at all (the standard deviation of this participant more than doubled his/her right high average response time).

The reaction times of the remaining 80 subjects were collected and polished with the DMDX programme. A response that occurred before the tone or 3 seconds after the tone was not recorded at all (in some cases, 3 seconds after the tone meant that the sentence had long finished), while responses deviating 2.0 SDs from the average were eliminated (this affected at 4.3% of the data). The resultant measures were then organised according to experimental condition. The analysis of reaction times was carried out with the SPSS package. The following two tables collate all the RTs (measured in milliseconds) per condition and analysis.

Type	*Position*		
	1	2	3
A	257.22 (SD: 59.1)	222.51 (41.0)	206.78 (40.1)
B	252.40 (52.0)	217.33 (43.9)	205.26 (44.3)

Table 4.1: Experiment 1 Subjects analysis, N=80

Type	*Position*		
	1	2	3
A	257.04 (SD: 24.4)	222.78 (20.2)	207.05 (12.9)
B	253.23 (24.3)	217.32 (19.9)	205.00 (17.7)

Table 4.2: Experiment 1 Items analysis, N=60

A repeated-measures analysis of variance showed that the *click position factor* was significant for both the subjects and items analyses ($F_1(2,158) = 144$, $p < .001$; $F_2(2,118) = 295$, $p < .001$; $minF'(2,265) = 96.76, p < .001$), while the *sentence type factor* was only significant for the subjects analysis ($F_1(1,79) = 4.66$, $p < .05$;

$F_2(1,59) = 2.48$, n.s.; $minF'(1,114) = 1.61$, n.s.). There was no interaction effect in either analysis ($F < 1$).

Nevertheless, a two-tailed t test was carried out to compare pairs within (A1 vs. A2, ecc.) and across sentence type (A1 vs. B1, ecc.). All within-sentence type comparisons were significant ($p < .001$ in every case), while the A2-B2 pair proved to be significant in the subjects analysis only ($t_1(79) = 2$, $p < .05$; $t_2(59) = 1.3$, n.s.).

4.3.2 Experiment 2

In order to make sure that the results of the previous experiment did not reflect a possible "habituation" to the task, we ran a modified version that included filler sentences and a comprehension task. It was hoped that the variety of the fillers (mixed with the experimental sentences) would force the processor to parse each sentence anew (as it were), while the inclusion of the comprehension task was intended to probe whether subjects were in fact paying attention to the meaning of the sentences they were hearing.

Method

Materials. The experimental items were the same as in the previous task. 60 new sentences were now constructed to act as fillers. 24 of these fillers were long, biclausal sentences. Another 24 were monoclausal sentences with a different word order from the canonical subject-verb-object. Given that Spanish allows much freedom in syntactic construction —all word orders are legitimate in certain constructions— we were able to construct 24 sentences of great variation. The remaining 12 fillers were exactly like the experimental items (six of Type A and six of Type B) but did not carry a tone at all. This was also the case for 12 other fillers; namely, six biclausal and six non-canonical sentences did not contain a click (see Appendix B for the full list of materials). A further six sentences were constructed in order to be included in the practice phase. All these sentences were recorded by the same speaker and using the same equipment. However, the quality of the new recording was significantly different from that of the experimental items, and it was therefore necessary to edit both the new and the old recordings so that the difference was unperceivable. This was achieved with immodest success by first eliminating *DC offset* from the new recording with Praat, and then eliminating background noise in both recordings with Cool Edit Pro. A short test with various colleagues showed that any difference in quality could no longer be perceived. Regarding the comprehension task, 26 questions were constructed, 12 for the fillers, 12 for the experimental items and 2 for the practice session. The questions were rather simple in formulation and would query an uncomplicated aspect of either the subject, the object or the verb of the corresponding items. The answer required was either a *yes* or a *no* (the questions and the required answers can also be found in Appendix B). All other significant details (generation and introduction of tones, ecc.) remained unchanged from the previous experiment.

Procedure. The same as in the previous experiment, but with the addition of the fillers and the comprehension task. The practice phase was still composed of 12 items, but these now included at least an example of each new condition, including two sentences without a tone and two questions. The fillers and the experimental sentences were randomised together for this version, which naturally included the questions some of these items were associated with. Regarding the comprehension task, each question appeared on the computer screen and the participants recorded their answers by pressing either the S key (for *sí*, that is, yes) or the N key (for, well, *no*). Upon entering the answer, a message on the computer screen would instruct the subject to press the space bar in order to hear the next sentence. Participants were also advised not to press a button if a sentence did not have a tone. The experimenter made sure that every subject understood the procedure during the practice session. In all other significant respects, the new task remained exactly the same as in the previous experiment. The session was now significantly longer, however, taking close to 40 minutes to complete. The experiment took place at the psychology department of the University Rovira i Virgili in April 2011.

Participants. 76 year-4 psychology students (15 male, 61 female) participated in the experiment. This was a different set of participants from Experiment 1. They were awarded a 0.25 credit towards the final grade on a core-module course for their participation. The mean age was 22 years, all subjects had normal or corrected-to-normal sight and not known hearing impairment. All were native speakers of Spanish.

Results

The responses of two participants were eliminated as they belonged to a different age demographic in respect to the other subjects (there was a difference of at least thirty years between these subjects and the average age of the group). A further subject was eliminated as his/her average response time was close to a full second and furthermore failed to respond to 29 clicks. An analysis of the comprehension task showed that subjects hardly made any errors, and apart from a subject who erred in 29% of the questions, everyone else was well under that figure. We had settled on a 30% cut-off and therefore decided not to eliminate anyone else from the final analysis. The responses of the remaining subjects were collected and polished following the same procedure of the previous experiment. As before, responses deviating 2.0 SD from the mean were eliminated, affecting 4.3% of the data.

The reaction times, summarised in the following two tables, were again analysed with SPSS.

The analyses of variance with subjects and items as a random factor once again showed that the *click position factor* was significant ($F_1(2,144) = 69.5$, $p < .001$; $F_2(2,118) = 43.8$, $p < .001$; $minF'(2,237) = 26.86, p. < 001$), while the *sentence type factor* did not prove to be significant in either analysis ($F < 1$). Naturally, the

Type	*Position*		
	1	2	3
A	395.69 (SD: 94.2)	355.16 (92.6)	345.16 (93.5)
B	394.84 (96.8)	356.61 (98.0)	337.77 (93.4)

Table 4.3: Experiment 2 Subjects analysis, N=73

Type	*Position*		
	1	2	3
A	396.31 (SD: 45.1)	354.31 (43.3)	344.17 (34.7)
B	394.97 (44.3)	356.73 (45.5)	338.35 (36.5)

Table 4.4: Experiment 2 Items analysis, N=60

interaction effect was also not significant ($F < 1$, for both analyses).

Despite the fact that the sentence type factor was not significant in either analysis, planned comparisons were carried out anyway. All within-sentence type pairs proved to be significant ($p < .01$, in every case) except for the A2-A3 pair in the items analysis ($t_2(59) = 1.3$, n.s.; this pair was also borderline significant in the subjects analysis: $t_1(72) = 2$, $p = .048$). No across-sentence type pairs were significant ($p > .05$ in every case).

4.3.3 Experiment 3

Given that the RTs of the previous experiment were significantly higher than those of Experiment 1 (and also higher than those reported in the literature), we decided to run a modified version of Experiment 2 to make sure that the overall length was not affecting subjects' performance (and by extension the statistical significance of the results). In the new version, the overall task was divided into three sessions in order to provide participants with a short resting period between presentations.

Method

Materials. The same as in the previous experiment.
Procedure. The same as in the previous experiment, but with the addition of two breaks. The overall task was consequently divided into three even blocks. The practice session was now composed of 9 items and a practice sentence was placed at the beginning of each block. During the break, the computer screen would turn white and subjects would be instructed to rest and relax, but to not disturb the others. The break would last two minutes, and at the end the screen would turn

black in order to signal that the break had finished. The session would restart as soon as the subjects pressed the space bar. A third and final white screen indicated that the overall session had finished. The experiment took place at the psychology department of the University Rovira i Virgili in December 2011 and lasted around 30 minutes.

Participants. 77 year-2 psychology students (8 male, 69 female) participated in the experiment. This was a different set of participants from Experiments 1 and 2.They were awarded a 0.25 credit towards the final grade on a core-module course for their participation. The mean age was 22 years, all subjects had normal or corrected-to-normal sight and not known hearing impairment. All were native speakers of Spanish.

Results

The responses of two participants were eliminated as they belonged to a different age demographic with respect to the other subjects. A further ten subjects were also eliminated as they did not meet reasonable expectations regarding average performance. In particular, two subjects had an average response time that was close to 2 seconds while another subject failed to record a single response. An analysis of the comprehension task showed that subjects hardly made any errors, and apart from a participant who erred in 40% of the questions, everyone else was well under that figure. As we had settled on a 30% cut-off, only a further single subject was eliminated. The responses of the remaining participants were collected and polished following the same procedure of the previous experiments. Again, responses deviating 2.0 SD from the mean were eliminated; in this case, 4.0% of the data were affected.

As before, the reaction times, summarised in the following two tables, were analysed with SPSS.

Type	*Position*		
	1	2	3
A	340.71 (SD: 89.8)	290.86 (79.1)	283.00 (67.4)
B	335.42 (88.9)	296.54 (96.5)	291.25 (81.5)

Table 4.5: Experiment 3 Subjects analysis, N=66

Type	*Position*		
	1	2	3
A	340.71 (SD: 51.3)	290.86 (47.3)	283.00 (35.3)
B	335.42 (52.9)	296.54 (44.3)	291.25 (48.1)

Table 4.6: Experiment 3 Items analysis, N=60

The analyses of variance with subjects and items as a random factor once again showed that the *click position factor* was significant ($F_1(2,130) = 70.21$, $p < .001$; $F_2(2,118) = 36.61$, $p < .001$; $minF'(2,218) = 23.77, p. < 001$), while the *sentence type factor* did not prove to be significant in either analysis ($F < 1$). The interaction effect was also not significant ($F_1(2,130) = 1.5$, n.s.; $F_2 < 1$).

As before, pair comparisons were carried out. All within-sentence type pairs proved to be significant ($p < .01$)except for A2-A3 ($t_1(65) = 1.5$, n.s.; $t_2 < 1$) and B2-B3 ($t < 1$ in both analyses). No across-sentence type pairs were significant ($p > .05$ for every comparison).

4.4 Discussion

Rather surprisingly, the most general of predictions was not confirmed at all. As mentioned above, a number of factors and some experimental evidence suggested that RTs towards the end of a clause (which in this case was also the end of a sentence) would be higher. An "end-of-clause" effect had been obtained by Abrams & Bever (1969) and Bever & Hurtig (1975) using the click detection paradigm, while a similar "wrap-up" effect had proven to be very robust in self-paced reading paradigms, where reading times tend to be much higher towards the end of a sentence (Just et al., 1982). These are prima facie very natural results to obtain; they would be predicted to follow from the very general point that the more material the parser is inputted, the more strained working memory is likely to be. Or in other words, the ceaseless inflow of material would result in a ever-greater number of open nodes, and these would only be "branched off" towards the end of a sentence (or of a clause). Ambiguity resolution, the closing of the sentence node and the construction of the underlying proposition are some of the other well-established phenomena that further suggested an end-of-clause or wrap-up effect —or at least this is what is assumed in a number of models of structural complexity and parsing.[46]

In our experiments, however, RTs were in fact greater at the very beginning and shortest at the very end. Moreover, there was a regular decrease in RTs from the first to the last position in both sentence types and in all three experiments. This decreasing progression is rather robust, and the high significance of the (click) position factor is further confirmation. As a whole, this datum might be taken to suggest that subjects are progressively better prepared to respond to the tone the more settled they are, perhaps the reflection of some sort of "uncertainty" factor that does not appear to be related to either the end-of-clause or the wrap-up effect (recall that these two effects conflate in the experimental materials we employed).

Given the high significance of the click position factor, we conflated the two

[46]This point applies to a variety of models: Gibson's (1998) *storage-and-integration-costs* theory, Hawkins's (2004) *immediate constituents* account, or Frazier & Fodor's (1978) *sausage machine* (the latter postulates a syntactic process in which incoming elements are kept in (working) memory until they can be packaged into phrases).

sentence types into one single condition in order to assess the significance of the click position when treated as a simple effect. That is, the RTs to position 1 in both sentence types were merged, an average calculated and the same process was applied to the other two positions. The overall averages for positions 1, 2 and 3 in the three experiments are represented in the boxes below.

Random variable	*Position*		
	1	2	3
Subjects	254.81 (SD: 5.9)	219.92 (4.5)	206.02 (4.6)
Items	255.14 (1.6)	220.05 (1.5)	206.02 (1.2)

Table 4.7: Position factor Experiment 1

Random variable	*Position*		
	1	2	3
Subjects	395.26 (SD: 10.7)	355.89 (10.8)	341.46 (10.7)
Items	395.64 (4.2)	355.52 (4.1)	341.26 (3.3)

Table 4.8: Position factor Experiment 2

Random variable	*Position*		
	1	2	3
Subjects	338.06 (SD: 10.7)	293.70 (10.3)	287.13 (8.8)
Items	338.06 (4.5)	293.70 (4.0)	287.13 (3.5)

Table 4.9: Position factor Experiment 3

Pair comparisons show that the differences in RTs among the three positions were significant in the first two experiments ($p < .01$, in every case), while the 2-3 pair did not prove to be significant in the third experiment ($t_1(65) = 1.5$, n.s.; $t_2(59) = 1.0$, n.s.). I will come back to this below; now I turn to an analysis of the RTs to the filler sentences in Experiments 2 and 3 in order to check if the uncertainty effect played any role in the response patterns.

Given that the clicks were introduced in a somewhat random manner in the construction of the fillers, a correlation analysis was conducted in which x stood for the number of syllables after which the click would appear in each item and y was the reaction time to the click. The Pearson's correlation was $r_{xy} = -.681$, $p < .01$ in Experiment 2 and $r_{xy} = -.633$, $p < .01$ in Experiment 3, indicating that the greater the number of syllables (that is, the deeper into the sentence the click is), the lower the reaction time to it. Plausibly, then, the uncertainty factor is having a great effect in subjects' performance in the overall task.

Upon closer inspection, in fact, the data reported in Abrams & Bever (1969) also exhibit a position effect that is very similar to what was obtained here. It will be recalled that these authors established three different click positions in sentences such as *since she was free that day, her friends asked her to come*; namely, on the last word *before* the main clause break, *in* the clause break, and on the first word *after* the clause break.[47] The following are the RTs they obtained, placed in the right order: 243 ms., 230 and 216. Similarly, Holmes & Forster (1970) found that RTs to clicks placed on the first half of a biclausal sentence were shorter than the RTs to clicks introduced in the second half. If what I am suggesting here is at all correct, the higher RTs are not the result of the click being placed before the end of the clause; rather, it is the consequence of being placed at an earlier position and subsequent positions are reacted to faster because of a position effect.

As a matter of fact, a decrease in RTs across a sentence has been reported in other detection tasks such as phoneme- and word-monitoring, as discussed by Cutler & Norris (1979). According to these authors, however, the click detection data are not comparable to these other monitoring tasks, a rather surprising statement, given that Cutler & Norris (1979) focus on the very same studies I am discussing. As it happens, their conclusion is based on a mistaken analysis of the Abrams & Bever (1969) study, as they ignore the idiosyncrasies of that work. Indeed, Abrams & Bever (1969) exposed their subjects to repeated presentations of the same material, and naturally, subjects' performance progressively improved. The response pattern also changed a bit in successive presentations —nevertheless, the first position was always reacted to slower than the successive positions—, but given that subjects were reacting to now-familiar sentences (and familiar click positions), these responses are not comparable to those of our experiment (or to those of the monitoring tasks Cutler & Norris (1979) discuss). Moreover, Abrams & Bever (1969) also ran a version of their experiment in which the materials were artificially reconstructed (that is, each word was recorded separately and then spliced together into coherent sentences) and the responses they obtained on the first presentation exhibited exactly the same pattern as in the first presentation of the "normal" exposure. Clearly, Cutler & Norris (1979, p. 129) are too brusque in drawing a line between the memory resources involved in reacting to sentence-internal targets (such as those of phoneme- and word-monitoring tasks) and those at work in targeting sentence-external elements (as in a click-detection task). Rather, monitoring tasks of all types should be viewed as "divided attention tasks" (p. 130) in terms of the competition between reacting to the extraneous material the perceptual systems gather and the internal representations the parser (at some stage) analyses.[48]

Holmes & Forster (1970) briefly discuss the position effect and reasonably suggest that subjects must be experiencing "maximal uncertainty" at the beginning

[47]The main clause break for this sentence is located between *day* and *her*.

[48]Nevertheless, it is true that RTs to a click are significantly faster than RTs to a phoneme or a word, and this certainly requires an explanation (see infra).

of a sentence, something that is plausibly reflected in the memory load involved. This maximal uncertainty makes reference to the expectations and predictions of the parser during processing and it is not quite the same notion I have described above. According to Holmes & Forster (1970), then, the processing load at the end of a clause ought to be minimal, given that 'structural information conveyed by the last few words would tend to be highly predictable' (p. 299). In other words, the cognitive resources exerted by the primary task (parsing a string) are much greater at the beginning of a sentence, while towards the end the attentional mechanisms at work in the perception of a click have access to more resources —i.e., there is less competition for resources between the primary and the secondary tasks— and hence reactions to a click placed late in a sentence are much faster.[49]

There are, then, two aspects to the uncertainty factor I am identifying. One has purely to do with the position effect of the click —i.e., the deeper into the sentence the click is, the more prepared the subjects are to respond to it—, while the other aspect has to do with the linguistic expectations and predictions of the parser (such as predicting the verb, its complements, ecc.), which are much greater at the beginning of a sentence than at the end. In rough outline, both aspects ought to conspire into producing greater RTs for early click positions, but a comparison of the materials of the last two studies I have discussed and those of Cohen & Mehler (1996) will show that the two aspects can indeed diverge, suggesting that a number of different factors are at play in click detection tasks.

In the first three experiments reported in Cohen & Mehler (1996), length was controlled for across two types of sentences and different RTs were recorded in the same position across these sentence types, which naturally suggests purely structural effects. Tellingly, though, Cohen & Mehler (1996) used relative clauses, which are certainly more complex than the non-relative structures that both Abrams & Bever (1969) and Holmes & Forster (1970) employed. That the position effect appears to have been nullified in Cohen & Mehler (1996) may be the result of a point I made earlier; namely, that structural properties are likely to have a greater effect on cognitive load when memory resources are pushed to the limit, which is probably the case vis-à-vis relative and non-relative phrases. A closer look at their materials will further illustrate.

In the first experiment, Cohen & Mehler (1996) compared reversible subject and transposed object relatives in French, a highly relevant pair to compare given that in this condition the complementiser is the only differing element between the two sentences (*qui* in subject relatives, *que* in the object constructions).[50]

[49]It is hard to work out how robust this effect actually is in the experiment Holmes & Forster (1970) ran, as we are not provided with exact RTs and length was not controlled for. Recall that these authors also found that RTs in a clause break were shorter than in positions wherein the click would interrupt a large number of constituents, but in some sentences the clause break would precede the no-clause break, and in others it was the other way around.

[50]Much as before, I employ the | symbol to mark where the click was placed, while the numbers within brackets indicate the RTs. The translation for these French sentences can be found in the original paper.

(4.10) Le savant (qui connait le docteur) t|ravaille. . . (218 ms.)

(4.11) Le savant (que connait le docteur) t|ravaille. . . (234)

Note that the RTs to a click placed right after the embedded clause indicates, in agreement with the literature (especially regarding the data amassed with self-paced reading paradigms, see Gibson op. cit.), that object relatives are harder to process than subject relatives. In a second experiment, these results were replicated (the RTs were 248 and 272, respectively) and then compared to a new click position: right before the end of the embedded clause.

(4.12) Le savant (qui connait le d|octeur) travaille. . . (249 ms.)

(4.13) Le savant (que connait le d|octeur) travaille. . . (250)

Interestingly, RTs to clicks before the end of a clause are not different across sentence type, suggesting a position effect once more. In this case, it seems that the cognitive load an object relative exerts is in fact operative *after* the embedded clause has been processed, but not during it. In the third experiment, though, the object relative was presented in its natural canonical order (i.e., it was not transposed) and the differences disappeared altogether:

(4.14) Le savant (qui connait le docteur) t|raivalle. . . (262 ms.)

(4.15) Le savant (que le docteur connait) t|raivalle. . . (264)

The last datum is very relevant, as it suggests that object relatives in their normal manifestation are not more difficult to process than subject relatives —at least in the case of speech perception. Putting all these data together, it seems reasonable to postulate that the position factor has a greater role to play in the explanation of the response patterns in click detection tasks than has usually been recognised. More precisely, the design of a monitoring experiment appears to influence whether the position effect of the uncertainty factor is operative or not (and to what extent it is).

Furthermore, it is quite unlikely that the end of a clause (or of a sentence) exerts much memory load in speech perception. In fact, the results of our experiments, combined with the data here discussed and the results of other monitoring tasks, point to the inexistence of end-of-clause or wrap-up effects in speech perception. Instead, the dual uncertainty factor is a much more central feature, and a re-evaluation of the click-monitoring technique in these terms may well be in order (I will come back to this in the postface).

The monitoring data clearly contrast with those obtained with the self-paced reading and click-location paradigms, but these offline techniques are plausibly not tapping the same mechanisms underlying the real-time working memory the click detection apparently tracks. After all, reading and listening are different phenomena, as are remembering a sentence to then mark the just-heard click and pressing a button in the middle of listening to some audio material. The wrap-up effect, in particular, might be a sui generis phenomenon of reading, and not of speech perception per se. Likewise, while the end-of-clause effect is clearly related to the construction of the underlying proposition, this phenomenon is likely to only be perceptible to the experimenter when it is brought out by the research technique employed.

Be that as it may, the RT patterns ought to tell us something about structural properties of the parsing process beyond purely perceptual properties such as the uncertainty factor. After all, these patterns cannot be the result of purely perceptual phenomena; as Holmes & Forster (1970) indicate, RTs to clicks in isolation are much faster. Not a lot faster, as Cohen & Mehler (1996) mention —indeed, these RTs are close to 'simple reaction times to elementary auditory stimuli' (p. 96)— but sufficiently so for us to believe that these response patterns point to early and basic operations of the parser.

Structural properties proved to be of some significance in the first experiment, but perhaps not all that compelling given the statistical analysis —the analysis of variance showed that the sentence type factor was only significant in the subjects analysis. Nevertheless, given the simplicity and structural similarity of the two conditions we employed, there is a case to be made for the suggestion that at this level of study any structural effects may be more substantial than what prima facie appears to be the case. After all, the two types of sentences we employed were very simple subject-verb-object structures with a single difference: whether the subject or the object was a complex structure. At this level of investigation, therefore, structural effects are not expected to surface unambiguously. Naturally, had we decided to compare cognitive load across very different types of sentences, say between right branching subjects (like those of Type A sentences) and numerations (of the type *the tiger, the lion and the giraffe were captured in the same savannah*), there would have not been any grounds for comparison. That is, parsing such different structures likely involves unrelatable computations, resulting in sui generis memory loads. Indeed, it is because we constructed such similar sentences, with embedded SHCs at different locations, that the investigation on possible recursive sub-routines was at all justified.

Moreover, in this first experiment the across-sentence type comparisons unearthed a surprising (albeit small) "verb effect". Indeed, the (small) difference in RTs between positions A2 and B2 —that is, before or after the verb— turned out to be statistically significant in the subjects analysis, which might be taken to suggest that the parser is more strained before processing the verb; a sort of *verb search* sub-operation, as it were. At a more general level, this result could be interpreted as suggesting that the parser is sensitive to the asymmetric macro SHC structure of sentences; that is, the difference between the subject and the verb-object(s) complex. This was already postulated to be the case regardless of whether the parser proceeds recursively or iteratively, given that it is a plain fact of language that sentences exhibit this type of geometry. Nevertheless, a verb effect such as this has not received much attention in the literature; indeed, its existence has been explicitly denied (see infra).

It is of interest to note at this point that the results of the last two experiments differ from the first experiment in various ways. First of all, the RTs in the latter were significantly higher than in Experiment 1 —they were moreover significantly higher than the RTs usually reported in the literature. Secondly, the structural effects of the first experiment all but disappeared, and only the position effect remained significant (which included the robust linear decrease in RTs). The differences in RTs among the three experiments are likely the result of the overall length of each session, and we note that while the average response time in Experiment 1 was 240.68 ms., increasing to 381.34 in Experiment 2, the breaks we introduced in Experiment 3 managed to bring the average to the in-between value of 313.67 ms. We also note that Experiments 2 and 3 appear to be more complicated (and last significantly longer) than the experiments reported in the literature. Indeed, while Cohen & Mehler (1996) did employ both a comprehension task and fillers —both of them being less onerous than ours—, their critical items did not even constitute a third of the total number of experimental sentences we used. Furthermore, the more telling results of the literature —arguably those of Abrams & Bever (1969) and Holmes & Forster (1970)— were obtained in experiments that in fact did not use either fillers or a comprehension task; this was also the case for our first experiment.[51]

In order to delve deeper into these issues, we decided to carry out further analyses of the data obtained in our three experiments. Taking our heed from the Marslen-Wilson's (1985) treatment of his speech shadowing data, we decided to form two groups of subjects for each experiment. By calculating the median value for each set of responses, we formed a group of "fast" respondents (those subjects whose average response time was lower than the median) and a group of "slow" respondents (the top half). The median values for each experiment were the following, in the right order: 238.85, 376.9 and 300.5. Apart from Experiment 2, where N=73, the subjects of the other two experiments were evenly divided into two groups. The following tables show the average RTs for each sub-group of subjects in the three experiments, by sentence type and click position.

According to an analysis of variance, the click position factor proved once again to be highly significant for each sub-group of subjects ($p < .001$ in every

[51] If there is one methodological feature that characterises most of the click-location and -detection experiments is the small set of experimental sentences employed, usually less than 20. This is possibly due to the instruments then available, but it is very likely that a short number of critical items may be a necessity in these tasks considering the factors we are discussing. Still, the most recent of the aforementioned studies (viz., Cohen & Mehler 1996) used 16 critical items in each experiment, while the number of fillers and comprehension questions varied across the four experiments therein reported (64–89 for fillers, 14–19 for questions) —in total, well below the number of items we employed. I might as well add that the length of the click these studies employed also varied significantly. Abrams & Bever (1969) used a 25 ms. tone; Bever & Hurtig's (1975) was 30 ms. long; Holmes & Forster (1970) employed a 80 ms. click; Green (1977) generated a 10 ms. tone; and Cohen & Mehler (1996) report that while a 12 ms. signal was used in the first experiment, the tone for the other three experiments was 35 ms. long. For what is worth, the majority of click-location studies made use of a 25 ms. tone, just like in our experiment.

Sub-group	*Type*	*Position*		
		1	2	3
Fast Respondents	A	217.14	194.16	183.78
	B	217.87	191.34	182.08
Slow Respondents	A	295.34	249.49	228.65
	B	285.24	242.05	227.31

Table 4.10: Sub-groups Experiment 1

Sub-group	*Type*	*Position*		
		1	2	3
Fast Respondents	A	336.02	288.29	273.46
	B	340.43	291.06	270.08
Slow Respondents	A	498.57	437.44	433.15
	B	486.65	445.71	419.76

Table 4.11: Sub-groups Experiment 2

Sub-group	*Type*	*Position*		
		1	2	3
Fast Respondents	A	273.79	233.38	234.50
	B	265.44	231.96	235.04
Slow Respondents	A	407.64	348.34	331.51
	B	405.39	361.11	347.47

Table 4.12: Sub-groups Experiment 3

analysis). In the case of the last two experiments, the sentence type was not significant in any sub-analysis ($p > .05$), but an interesting pattern emerged when we carried out pair comparisons. As it transpired, the A2-A3 and B2-B3 pairs were not significant in any of the sub-analyses of Experiment 3 ($p > .05$), and were only significant for the fast group in Experiment 2 ($p < .05$ for both the subjects and the items analyses; note that all the other within-sentence type comparisons were highly significant in every sub-analysis, $p < .001$). In more general terms, when the position factor was treated as a simple effect (that is, when we conflated the two types of sentences into one), the 2-3 pair failed to reach significance in all but the fast group of Experiment 2 ($p < .05$). This is a surprising result, as these positions interact with the presence of the verb (in Type A sentences, the verb appears after the second click position, while in Type B the verb has been fully processed by the time of the second click position).

As for Experiment 1, the sentence type factor was not significant for the fast group of subjects ($F < 1$), but it proved to be for the slow group ($F_1(1,40) = 4.07$, $p = .05$; $F_2(1,59) = 4.55$, $p < .05$). However, there was no interaction effect between sentence type and click position ($p > .05$, for every sub-analysis).[52] All within-sentence type pairs proved to be significant, but no across-sentence type pairs were statistically relevant. All the pairs within the position factor treated as a simple effect were significant for both the fast and the slow groups of the first experiment.

Overall, it is of interest to note that much like past click-detection studies, our first experiment, which was only composed of critical items, yielded the most interesting results. This may indicate that monitoring tasks do not in fact necessitate the inclusion of fillers and a comprehension task; instead, it is likely that their presence may in fact negatively affect subjects' performance. At the time of the first experiment, we decided that neither a comprehension task nor a set of fillers were in fact needed at all. Given that the click-detection task is a case of speech perception, we concluded that it was not possible for subjects to completely ignore the linguistic material and exclusively focus on responding to the click —what Fodor calls the mandatory property of modules, a well-supported feature, we believe. As for the fillers, and considering the postulated competition in attentional resources between parsing the sentence and reacting to a click, we thought it likely that a great number of items would in fact quickly bring about a deterioration in performance. It seems to us that these pre-conceptions proved to be correct.

It is a common practice in psycholinguistics to employ fillers so that subjects do not become accustomed to the task. The thought behind this strategy is that if you manage to vary the material the subjects are exposed to, the processor would not be able to anticipate either structure or click position; it would instead be forced to parse the incoming material anew, as it were. Such a strategy is supposed to guarantee that the results are as genuine and transparent as possible, but we found that too much variation, and too much of it, skewed our results instead. Clearly, it cannot be the case that structural effects would be more operative in Experiment 1 than in Experiments 2 or 3, considering that the parser could have easily adopted a "template" to analyse the data in the former but not in the latter —i.e., Experiment 1 exhibits much less variety than Experiments 2 and 3. Rather, the fact that structural effects were manifestly at play in Experiment 1 must be taken as specially significant given the setting we constructed. After all, had we initially run the second or third version of the experiment, we would have naturally concluded that the fillers and comprehension task were a) tiring the subjects too much, as evidenced in the much higher RTs attained (even the RTs of Experiment 3 are still high compared to those of the existing literature), resulting in b) ambiguous and

[52]I ought to mention that for every analysis and sub-analysis I have mentioned in this chapter, the Eta squared values indicate a medium effect size for any of the structural factors that have proved to be significant and a rather large effect size for the click position factor.

opaque data on the underlying mechanisms. The logical next step would have been to run a stripped-down version (as in Experiment 1) and the resultant data would have vindicated this choice.

All in all, then, we consider the results of Experiment 1 to be the most genuine,[53] and the distinction between fast and slow respondents a real one (cf. close and slow shadowers in Marslen-Wilson 1985). Fast subjects seem able to abstract themselves from the material to a certain extent, while the analysis of the response patterns of slower respondents indicate that these subjects carry out a careful parsing of the underlying structure of sentences, no matter how simple it might be (and the structure of our materials clearly was simple).

This brings us to briefly consider the "verb effect" I mentioned supra. This effect is systematically denied in the literature, more obviously in Gibson's (1998) parsing model. According to this account, the processing complexity of linguistic structures is to be calculated in terms of the storage and integration costs that incoming elements incur. That is, it is a combination of the cost of storing the structure built so far (what Gibson regards as a *partial* representation) and the cost of integrating new words into the already-built structure. To be more precise, integration is calculated according to the "unit costs" each new discourse referent the parser receives is allocated. Of special interest for us is Gibson's treatment of the cost the verbs incur in the relative-clause sentences he usually studies with the self-paced reading paradigm. While the verb of the subordinate phrase and various other elements such as adjectives and prepositions are postulated to introduce new discourse elements (therefore effecting certain costs), Gibson is rather adamant that processing the matrix predicate (that is, the verb of the main phrase) is "cost-free" (ibid., p. 15) and therefore allocated a cost of zero (p. 26). Support for this assumption seems to be twofold: a) the parser is said to expect a matrix predicate ab initio anyway, and b) some experimental evidence indicates that it does not matter whether a relative clause modifies the subject or the object of a matrix verb, as the cognitive load remains stable (pp. 26–27; relevant sources cited therein). Naturally, reason (a) is mere assumption, and perhaps an unreasonable one. After all, even if the parser does expect a matrix verb, this does not mean that this expectation has no effect on cognitive load. Moreover, even though verbs play a rather central role in linguistic analyses, written language and even in daily communication, it is nevertheless true that a large part of linguistic production and comprehension surely involves navigating through many verb-less chunks, and it is not clear that the parser behaves qualitatively different in these cases (or that this does not have a specific effect on cognitive load).[54] Regarding (b), the data Gibson references may

[53]This is further supported by the fact that the linear decrease in RTs we have reported is actually more robust in Experiment 1 than in the other two, as an analysis of the RTs subject by subject and item by item shows. Indeed, 50% of subjects and 35% of items exhibit the decreasing tendency in Experiment 1, for 27% of subjects and 13% of items in Experiment 2 and 22% of subjects and 8% of items in Experiment 3.

[54]This is another aspect underlying the suggestion that the NVN template ought to be replaced with an SHC template, which is as applicable to verb-less chunks as it is to full-fledged sentences.

well be specific to relative clauses in particular (or complex structures in general); more importantly, it is worth pointing out that these data were obtained by analysing reading times in a self-paced reading task and it is very likely that the resulting data inform us on those mechanisms underlying reading —that is, the data may not necessarily elucidate a property of speech perception. The self-paced reading paradigm, after all, is hardly the most ideal of techniques to probe online cognitive load.

In fact, given that we have obtained a conflicting result with rather simple, monoclausal sentences, it is likely that the verb effect would surface more clearly when the memory resources of the parser are pushed to the limit in more complicated sentences. Naturally, the effect reported here is too small to draw any drastic conclusions and we need to further investigate this matter as to elucidate its real significance (I will also come back to this in the postface).

Regarding the specific aims of our investigation, however, an analysis of the RTs at the critical areas we identified shows that there do not appear to be any deferred operations at all. That is, considering the regular decrease in RTs *within* sentence type and the similarity *across* sentence type, the distinctions we drew earlier do not appear to validate the presence of any recursive sub-routines.

Nevertheless, the take-home message of the experimental work here discussed is that reacting to a tone demands non-trivial attentional resources. Therefore, designing a click-location experiment requires much careful consideration if the right balance is to be struck between the quantity and complexity of the experimental materials and the memory threshold that is involved.

Take Green (1977) and Gómez et al. (2011) as two illustrative examples of this point. It is likely that the former failed to achieve the right balance, as evidenced in the rather high RTs obtained (these are reported as geometrical means, and so the arithmetical equivalents would in fact be even higher). Indeed, in the memorisation condition (see supra), RTs were close to 552 ms., while the continuation condition exhibited RTs of nearly 900 ms. These RTs are significantly higher than those reported by any other study, and they are especially surprising given that monoclausal sentences were employed in this study. In discussing these results, Green (1977) concludes that the different nature of the two conditions had a clear and particular effect on performance (as exhibited in the different RTs), but he fails to note that such high RTs suggest that the design of the experiment itself influenced subjects' performance in general. Indeed, compared to what we were trying to do here, Green's experiment is rather cluttered. Besides the two aforementioned conditions, the sentence materials (as will be recalled, sentences like *the sleek sinewy leopard attacked the woman*) were manipulated across various other dimensions: the head of the noun phrase could either be concrete or abstract, or it could be of a high or low frequency; furthermore, three different positions were determined for the clicks (within the subject noun phrase, right after it, or at the end of the sentence).[55] The combination of all these variables and the requirement that subjects

[55]Interestingly, there was a linear decrease from the first to the second position, but the RTs in-

had to either memorise or complete chunks of linguistic material may have pushed memory resources beyond what is reasonable to do for the data to be reliable and significant. As far as the RTs go, this click-detection study is in a league of its own, in fact.

A different situation seems to have obtained in Gómez et al. (2011). In this word segmentation study, it is reported that subjects' performance diminishes after 2 minutes of exposure to a continuous stream of artificial but language-like speech (that is, composed of pseudo-words). Crucially, though, the response patterns that emerge after these 2 minutes seem to correlate with relevant linguistic categories —i.e., clicks are reacted to slower if they are placed within pseudo-words than if they are placed between pseudo-words— suggesting that, for continuous speech at least, the right balance had been achieved. That is, during the first two minutes of exposure only the position effect and the uncertainty factor appear to play any role in the response patterns, but after two minutes structural effects start to surface. This state of affairs is precisely what all monitoring tasks need to achieve.

Let us now close this section with some comments regarding the general architecture of the parser. Putting it all together, we may conclude that the parser attempts to construct the underlying proposition of a sentence as soon as it can. This is not quite the same as stating that the processor tentatively allocates meaning to every segment it receives (including merging the information of the incoming elements to the representation that is being constructed). Incremental parsing, after all, does not mean that the incoming elements are automatically ordered in a subcategorisation frame. Indeed, at one stage of parsing, interpreting a noun may just involve deciding whether it is animate, abstract, ecc., but not that it is a subject or an object. It was precisely this distinction that salvaged the importance of the clause in sentence processing.[56] The data reported here, however, suggests that in fact there is no branching off of open nodes towards the end of a clause —at least not when there is no ambiguity to resolve. That is to say that the processor does not keep incoming elements in working memory until these can be appropriately packaged; there is no wrap-up effect. Instead, it is very possible that the first-pass of the parser "looks" for the verb and attempts to "merge" its arguments as promptly as possible; packaging, then, is carried out much sooner than currently posited.

The model of the syntactic parser here defended would roughly look something like this:

- preliminary analysis — first-pass — second-pass

The very first stage involves the imposition of a template upon the material the pro-

creased in the third and last position. Green argues that the fact that RTs after the noun phrase were not higher than in the prior position suggests that the propositions at that point witnessed (i.e., the leopard is sleek and sinewy) are not stored in individual representations. All things considered, though, it seems to me that the data are better explained in terms of the position effect and the uncertainty factor.

[56]This distinction allowed Townsend & Bever (2001) to keep the click-location data as part of their comprehension model, in fact.

cessor receives —a purely perceptual strategy. Despite being perceptual in character, we have suggested that the template the parser employs respects the asymmetric structure of sentences; that is, a template that is sensitive to the fact that the subject is hierarchically more prominent than the verb-objetc(s) bundle; an SHC template. If further experiments confirm this proposal, the "verb search" effect would tell us something about the actual connection between perception and the parser, given that whatever mechanisms underlie this effect are likely to operate in the nexus between the preliminary analysis and the first-pass of the parser; that is, the preliminary analysis and the first-pass of the parser may not be entirely independent components. Plausibly, in the first-pass the parser would attempt to compute, close and discharge the chunks it processes by carrying out two general operations:

- Shift: keep incoming elements in memory until they can be combined
- Reduce: "merge" words into phrases and close them from further operations

It was at this stage that deferred operations were postulated, but we found evidence for Stabler's conjunctive strategy instead. That is, in the assembling of SHCs structures, the parser successionally proceeds in an iterative manner. Nevertheless, the linear decrease in RTs argues against a conjunctive strategy in which all steps are uniform and equally costly. Rather, the *compute-close-discharge* chain takes place as soon as possible, which results in a greater cognitive load at the beginning of a sentence, and perhaps a particular cost in the search for the main verb.

Note that these results suggest that some sort of hierarchy is already being developed in the first-pass stage, as the process is not quite the result of a flat conjunctive operation. Nevertheless, the hierarchy so far constructed is rather minimal and skeletal, nowhere near the intricate structure that sentences in fact manifest. This may well point to the nexus between the first-stage and the second-stage of the parser, where the latter ought to be identified with the "synthesis" of the AxS approach. It is at this level that higher-order operations are likely to be operative, such as reanalysis in ambiguity resolution, lexical effects, and direct applications of generative rules. Of this part of parsing, we have had little to say here, and a fuller account of the parsing model we have tentatively outlined will have to await another publication.

*****

It will have been noticed that for a chapter on recursive parsing, I have not said anything about self-embedded sentences, the sort of structures that most people have in mind when talking about recursion. There are many reasons for this choice. For a start, this chapter was about the actual operations of the parser, and whether these applied recursively, which is a rather different matter to whether the internal hierarchy of self-embedded sentences is appropriately computed in real-life communication. This point was already alluded to before, in connection to the artificial grammar learning (AGL) paradigm. I will discuss this paradigm again in the next chapter, but at this point it is worth discussing some peculiar features of some AGL studies. Indeed, AGL scholars seem intent in blurring the two levels of analysis we have clearly kept apart here (viz., the computational and the algorithmic), and they also show an incessant focus on Marr's third level, the hardware implementation (i.e., the study of how the representation and the algorithm are physically realised in the human brain).

Let me restate the general state of affairs in AGL. Sets of production rules are used to generate sequences of nonsense syllables; some strings are generated by recursive rules, yielding centre-embedding in some cases, crossed dependencies in others; appropriate tests are used to check if subjects are able to process the strings (or learn the rules, depending on who you ask); it has been noted that the mere processing of these strings is not a test for recursion, as many non-recursive strategies may be employed; modifications are introduced to make sure subjects process the hierarchy of the strings; finally, brain scans identify the regions that were activated during the processing of hierarchical and non-hierarchical strings.

Friederici et al. (2006) is a fairly illustrative example of this manner of proceeding, but note the two extrapolations and the omission. *Extrapolation 1*: the ability to process the hierarchy of a string is taken to be synonymous with recursion (Friederici et al., 2006, p. 2458), but this is not correct, as non-recursive mechanisms are indeed capable of processing recursive structures such as the self-embedded. The correct processing of hierarchical structures means not hierarchical, processing building, let alone recursive, processing building. *Extrapolation 2*: we know humans possess a grammar in which recursion is a central property, a grammar that generates hierarchical structures of a certain type, but this is a property of the *grammar*, and not necessarily of the *parser*. Recursive specifications of algorithms may well be implemented iteratively (indeed, our data suggest that they are), and since the successful processing of hierarchical structures *are* performance data, much care must therefore be employed in the ascription of successful processing to properties of the grammar. One would naturally expect that linguistic knowledge makes processing and acquisition at all possible, but the relationship between competence and performance is rather tricky and not so straightforward. As Chomsky (1963, p. 390) pointed out, even though a finite automaton cannot capture the basic facts of linguistic *competence*, the system underlying linguistic *performance* is in fact such a thing, with the obvious corollaries. *Omission*: there is a leap from the correct processing of hierarchical structures, by extrapolations 1 and 2, to the neural basis of recursion, but nothing is being said about the second level of analysis. That is, we know nothing about the actual operations and mechanisms being executed in the processing of these sequences; the subject matter of cognitive psychology.

It was the very purpose of this chapter to at least offer a way to start probing these issues —that is, a way to discern the nature of the underlying operations of a computational process— even if ultimately much broader and wider factors were discussed. Be that as it may, our investigation compelled us to focus on much simpler structures whose processing nevertheless suggested possible recursive sub-routines.

Another reason for the omission of self-embedded sentences from this chapter has to do with the fact that these structures have been extensively studied by many scholars, and the evidence does not suggest any recursive sub-routines. Hudson (1996) provides a good review of the literature and a reasonable account for why some self-embedded sentences are hard, or indeed impossible, to process. According to Hudson (1996, p. 22), hearers cannot handle [N_1 [N_2 [N_3 - V_3] V_2] V_1] structures in which

1. a finite clause [$\ldots N_2 \ldots V_2 \ldots$] modifies N_1
2. N_2 is modified by a finite clause [$\ldots N_3 \ldots V_3 \ldots$]
3. N_3 is a common noun
4. upon processing N_1 - N_2 - N_3, it is hard to establish the meaning of V_2 and V_3

Given this classification, it follows that there would be many examples of self-embedded sentences that are in fact easy to process, in English or in other languages (these can be found in Hudson 1996). What is of interest here is the explanation Hudson provides for this phenomenon, an account that is centred on the fact that the hearer must keep concepts of different nature in memory during the processing of the difficult/impossible cases (i.e., finite vs. non-finite clauses, common nouns vs. pronouns and names, ecc.). This is a plausible explanation as far as it goes, but I will not delve deep into the literature to evaluate it properly. What seems pretty clear from the literature is that the difficulty/impossibility of certain self-embedded sentences is not the result of recursive sub-operations (such as keeping the uncompleted phrase in mind until the lower-level phrases are processed, ecc.).

Note, finally, that there is no question as to whether self-embedded structures can be at all understood. Given sufficient time and space (say, by employing pen and paper), even the difficult/impossible cases can be eventually understood. The problem, rather, is whether specific kinds of self-embedding can be processed in real-life interaction, a clearly distinct matter. Moreover, linguists have certainly not needed any experimental evidence to show that self-embedded structures can indeed be appropriately understood. That is, there has not been any need for the "empirical indicator" for recursion Fitch (2010) has defended, a point that ought to carry to the study of other cognitive domains. It is to this topic that I turn to in the last chapter.

Chapter 5

Recursion in general cognition

5.1 Universality claims

The previous three chapters have chronicled some of the diverse ways in which recursion is being treated in the linguistics literature. In particular, section 2.2 outlined the central role recursion has had in Chomsky's writings, a connotation that ought, I believe, to form the proper groundwork to linguistic theory overall. However, the last decade has witnessed a surprisingly heated debate on how central to linguistic cognition recursion really is, a dispute that was spurred when M. D. Hauser, Chomsky & Fitch (2002, HCF henceforth) hypothesised this property as perhaps the only feature unique to the language faculty.

Judging by the number of citations this publication has received over the last decade, its influence cannot be overstated —at least as a catalyst for the ensuing debate. This is unfortunate for a number of reasons. First of all, recursion did not in actual fact feature extensively in the article; rather, HCF constitutes an evolutionary and comparative conjecture regarding the possibility that the language faculty is the result of various systems and principles. Indeed, recursion is therein defined in very general terms as a neurally implemented (p. 1574) computational mechanism (p. 1573) that yields a potentially infinite array of discrete expressions (p. 1574) from a finite set of elements (p. 1571). Secondly, both Hauser and Fitch appear to have a different understanding from Chomsky as to what recursion actually constitutes. As is clear in M. D. Hauser (2009) and Fitch (2010), for example, these authors identify recursion with an embedding operation, whereas Chomsky's focus has always lied on the "finitary inductive definitions" at the heart of the CS of the language faculty. Remarkably, it is easier to connect Chomsky's interpretation to the rather vague characterisation in HCF than what appears in either M. D. Hauser (2009) or Fitch (2010).

Unfortunately, the subsequent debate, Pinker & Jackendoff on one side and Hauser, Chomsky & Fitch on the other, quickly deteriorated into a collection of verbose publications that were clearly not engaging each other (Pinker & Jackendoff 2005; Fitch, Hauser & Chomsky 2005; Jackendoff & Pinker 2005). Indeed, despite many statements to the contrary (as chronicled in section 2.3), the field has exclusively focused on a) whether self-embedded sentences are present in every human language, and b) whether similar structures appear in other species' systems of communication; that is, a matter of structures rather than mechanisms in the sense that this thesis has been at pains to clarify.

It is safe to say that this state of affairs is to some extent the result of the utter failure of the vast majority of scholars to engage with Chomsky's writings; clearly, a decent exegesis would quickly and easily identify his leitmotif. Heine & Kuteva (2007), for instance, while devoting a long chapter of their book on language evolution to a cross-linguistic quest for self-embedded sentences (and much else), frame the terms of the debate in a somewhat misleading manner with an astonishingly misrepresented quote from Chomsky. After framing their discussion of recursion in terms of self-embedding, so that a language is described as recursive if it exhibits self-embedded sentences, they quote Chomsky as stating that '[it is a] possibility that languages are nonrecursive', something that ought to 'be granted by everyone who has seriously discussed the subject'; further, even if languages turn out to be recursive, 'there is no reason to suppose that this must be so' (Chomsky 1980, pp. 120–2, cited in Heine & Kuteva 2007, p. 271). At face value, then, Chomsky's quote would appear to give credence to the overall approach —that is, to an investigation meant to establish if every language is "recursive" in this sense.

However, this quotation of Chomsky is entirely misapplied, betraying complete lack of understanding from the part of Heine & Kuteva (2007), if not dishonest citation. In pages 119 to 126, Chomsky (1980) is in fact discussing Dummett's contention that linguistic knowledge involves the capacity to recognise if a sentence is well-formed. In other terms, whether the language faculty is some sort of algorithm that correctly decides if a given element is part of the set of possible sentences after a finite amount of time; that is, if the set of possible sentences is a recursive set (as defined in section 1.1). This possibility was widely discussed by scholars such as Hilary Putnam and William Levelt in the 1960s and 70s, and Dummett's (and Chomsky's) worry centred on whether this factor would have any effect on the theory of the grammar the linguist devises.[1] It has, needless to say, absolutely nothing to do with self-embedded sentences; in this context, if a language is recursive, it merely means that its set of possible sentences is recursive —a technical connotation.

A more notorious example of scholarly work that utterly fails to engage Chomsky's treatment of recursion is to be found in Dan Everett's study of the Pirahã language (2005; 2009; 2010). Everett (2010), in particular, starts with the 'important book-keeping' matter of defining recursion, and he offers two interpretations, one that characterises recursion as an operation that applies over its own output, and another that is basically a definition of a recursive set (p. 1).[2] After stating that he

[1]Both Matthews (1979) and Chomsky (1980) conclude that it does not.

[2]In page 10, Everett claims that these definitions are what 'computer scientists mean by recursion', but no references are offered —unsurprisingly so.

will keep to the first definition, he moves on to a study of self-embedded sentences within Pirahã, surprisingly unaware that the connection between a mechanism that operates over its own output and self-embedded sentences is non-existent unless it is further assumed that the operation being carried out in succession is an embedding one. That aside, both recursion and iteration apply over their own output, but it is not in these terms that recursive generation should be understood, as has been repeated ad nauseum here.[3]

In reality, Everett focuses on configurational properties of sentences, but these do not directly provide a lot of information about the underlying generative system. Recall that in section 2.1 I insisted that self-embedded sentences did not constitute the evidential basis for recursion; rather, the unbounded novel linguistic behaviour did. Therefore, even if Everett is right about Pirahã and this language does not contain any type of self-embedding, their speakers still manifest an extensive range of expressions that exceeds memory of individual sentences learned by rote, making a computational system, and by extension recursion, a necessity.

Note that this is not to lessen the importance to his work on the structural properties of Pirahã, the validity of which seems to rest on two points. Firstly, Pirahã appears to lack "mental state" verbs such as *to think* and *to believe*, which would tell against the presence of self-embedded sentences. The second point revolves around the status of the verbal suffix *-sai*, a marker that Everett, in earlier work he now considers mistaken, argued could appear in two conditions: either as a nominaliser or as a clausal embedding indicator. In Everett (2005, 2009), he alternatively concludes that this suffix is a unique marker of semantic cohesion between parts of discourse, but Sauerland (2010) offers some experimental data that might cast some doubt on this. After carrying out a "maximum pitch" analysis on the two conditions the *-sai* marker appeared in Everetts earlier studies, Sauerland found that the pitch level in the nominaliser condition was indeed much greater than in the clausal condition, indicating that there are two versions of this suffix —and consequently, that one of them indicates embedding. On a related manner, chapter 4 offers another way to experimentally investigate this matter, as the click-paradigm could be employed to probe if reaction times are greater to clicks placed right after the boundary of the internal clause of the postulated self-embedded structure. It is of course to be seen if this is at all feasible, given the difficulty of carrying out experimental research with Pirahã speakers, but this remains an unexplored possibility.

Everett's work, then, is telling us something about what I called earlier, following Stabler (2010), the derived tree of a structure, but his work is not informative regarding the derivation tree, which is what should preoccupy the linguist and psy-

[3]I ignore Everett's response to the criticism of his work by Nevins, Pesetsky & Rodrigues (2007); in particular, I am disregarding the latter's statement that what HCF really meant by recursion was in fact *merge*. There is, of course, a sense in which they are right —namely, as sketched in chapters 2 and 3— but neither Nevins et al. nor Everett have been quite able to identify the recursive generation underlying *merge*; hence, my dismissal of Everett's discussion of this particular matter, as I consider it a red herring.

cholinguist (see Fig. 3.4 supra). Given that "finitary inductive definitions" are at the heart of the CS for language, and that all phrases are generated in such a manner, it is rather clear, in accordance with Tomalin (2011), that Everett's grandiloquent claims regarding the non-centrality of recursion are entirely fallacious. That is, *merge* 'recursively constructs' Pirahã objects as much as it constructs the syntactic objects of any other language.

In order to salvage his grand conclusions, Everett is reduced to a rather precarious situation; namely, to focus on the meagre treatment recursion receives in HCF —indeed, he makes no attempt to properly engage any of the publications I focused on in section 2.2. Rather amusingly, and even though he complains about the lack of definitions and clarity vis-à-vis recursion in HCF (Everett, 2010, p. 1), this does not stop him from selectively extracting those quotes that can be used to make his case. This is rather widespread in the literature, in fact. Many recent publications on the role of recursion in cognition (some of which have been reviewed above) have come up with rather outlandish definitions, which are then loosely related to HCF, even if on closer inspection, the actual work that is eventually presented has very little to do with it —or more importantly, with Chomsky's characterisation, which I have argued to be the correct one.

Regarding whether recursion is a species-specific feature, recall the discussion of some data from the artificial grammar learning paradigm in section 2.3 and chapter 4. In these studies, subjects were presented with regular patterns of strings to probe if they could extrapolate the underlying grammar. Fitch & Hauser (2004) compared the performance of humans with cotton-top tamarin monkeys, and showed that the latter were capable of learning a finite-state grammar, but not a context-free one.[4] Subsequent studies have attempted to investigate if non-human species would be capable of correctly parsing a^nb^n strings —demonstrating mastery of a context-free grammar.[5] Thus, Gentner et al. (2006) argued that European starlings were capable of learning context-free patterns, a conclusion that is doubted by van Heijningen et al. (2009), as the *ab* pairs were composed of either rattles (for $a's$) or warbles (for $b's$), and so a much simpler "counting" strategy could have been employed. Abe & Watanabe (2011), on the other hand, have shown that the Bengalese finch can master an a^nb^n pattern without such cues, as in this task the $a's$ and $b's$ were merely different syllables of their songs —that is, the *ab* pairs do not share phonological features as in the Gentner et al. (2006) study. This is an interesting result, but it is not quite true that '[h]umans are supposed to differ from other animals in their capacity to deal with hierarchically structured sequences'

[4]I gloss over whether the distinction between finite-state and context-free was well-modelled in Fitch & Hauser (2004); Rogers & Pullum (2011) argue that most AGL studies employ grammars and languages that are in fact below the context-free class.

[5]I am simplifying the exposition for the sake of the presentation. It is more accurate to say, as discussed above, that Fitch & Hauser (2004) were interested in probing the expressive power of the underlying grammar subjects had internalised, while the two studies I will now discuss attempted to discover if the rules of the grammar were literally operative in the parsing of these strings, a slightly different matter.

(Abe & Watanabe, 2011, p. 1072), and even less so that these results 'cast doubts on what is currently considered to be a unique characteristic of human language' (ibid.). As stated in section 2.3, a context-free rule like $S \rightarrow a(S)b$ guarantees that the *ab* pairs are matched, at least in the sense of being introduced at the same time in the derivation, but this constitutes a rather weak equivalence for the subject-verb configurations it is supposed to be modelling. Indeed, lexical items enter into structural relations of a fundamentally different kind, and the interaction of *merge* with the interfaces yields, as argued in chapter 3, the full generative extent of human language, a discrete infinity of sound-meaning pairs that appears to be completely unknown in the non-human animal kingdom.

Naturally, self-embedded structures are but a small part of the possible sound-meaning pairs; furthermore, it has never been claimed that all languages ought to exhibit them. The linguist is, after all, interested in the *potentialities* of behaviour, that is to say the universal features of the linguistic capacity.[6] This point has been repeated on numerous occasions (cf. Fitch et al. 2005 and the open peer commentary section in Evans & Levinson 2009), and there is no denying that a Pirahã child would acquire and exhibit productive use of self-embedded sentences if he/she was born in, say, an English-speaking country —and the same applies to any other feature of language. Linguistics must account for this eventuality, and in order to do so it appears necessary to postulate diverse and abstract principles that interact in particular ways upon specific contingencies from the environment. It does not follow, however, that for a linguistic feature to be universal, it must explicitly appear in every single language. Clearly, whatever turns out to be the universal set of properties of the language faculty, these will be rather abstract in nature and detail.

Some scholars have a curious way of opposing such a possibility, however. Evans & Levinson (2009), for instance, consider linguistic universals mythical, but only because they deny the existence of any underlying properties. Having done so, they then proceed to show that there is no explicit feature that is part of every language. There is a certain scent of superficial and infantile syntactic analysis in their paper, but more importantly, the overall argument completely fails to engage the position they attempt to critique.

In the case of Karlsson (2010), a similar situation obtains. Therein, he describes the details of a set of corpora analyses he conducted, which show that self-embedded sentences are hardly ever present in actual usage —and when they are, they do not exhibit many levels of embedding. He takes this fact to reflect quantitative and qualitative "constraints" on their use and structural depth, which he argues tells against the centrality of recursion and, moreover, indicates the finiteness of language (supporting Reich's 1969 contention, Karlsson (2010, p. 43) tells us). The latter is a clearly misplaced contention, however. First of all, Reich (1969) argued for the finiteness of language in terms of the author's intuition that you could not embed (*if* S_1, *then* S_2) into other (*if* S_1, *then* S_2) structures, in opposition to what Chomsky (1956) argued. This argument is somewhat dated, and

[6]The reference to Aristotle is intended; see ft. 21 below.

it would be interesting to know what Reich would make of the grammaticality of S_1 *and* S_2 *and* S_3... sentences, which take the system out of finiteness as much as self-embedding —the latter a puzzling fixation of scholars. As for the centrality of recursion, Karlsson can only conclude thus because he equates self-embedded structures with recursion —mistakenly, as it has been argued here. Nevertheless, this particular aspect of language use was already well-known more than forty years ago when G. A. Miller & Chomsky (1963) noted that these self-embedded sentences, 'being equally difficult for speaker and hearer' —because of memory constraints— 'simply are not used' (p. 471); further, they can simply be rephrased as left- or right-recursive expressions (p. 470). In any case, this is a fact of performance, not of competence.

Granted that much, Karlsson (2010), following Heine & Kuteva (2007), believes that self-embedding is not a real property of language, but merely a feature of the grammar the linguist proposes as a *model* for language —that is, an abstract and perhaps whimsical phenomenon. This is explicitly assumed by many contributions in van der Hulst (2011b; namely, by Mithun, Laury & Ono, Tiede & Stout and Harder), but it is a bizarre and irrational position to hold. First of all, such a stance would argue against all science, as the very possibility of postulating theoretical constructs in order to explain a given phenomenon is a necessary part of achieving understanding. Moreover, all these scholars engage in such exercises as much as anyone else; concepts like *nouns*, *verbs* and the like may well be closer to the surface, but they are as abstract as any other. Finally, this position presupposes that the term "language" has a clear and unambiguous reality (independently of grammar), ready-made to be studied by linguists, and perhaps genuinely manifested in corpora only. However, none of these scholars has provided a sensible and coherent account of the subject-matter of their studies in these very terms. Mithun (loc. cit., p. 39), for example, complains that language should not be viewed as a mathematical system,[7] but there is no indication whatsoever as to what phenomenon he is in fact purporting to be studying (mutatis mutandis for the other contributions).

The focus on corpora, incidentally, is clearly misplaced. Obviously, no linguist can ignore corpora studies, but it must surely be realised that the information they provide is rather limited. Clearly, scholars like Karlsson have too narrow a perspective, as it is not the case that they are interested in *any* type of corpus; rather, Karlsson seems to think that real-life language interaction, as in the sort of conversation one may engage in during a walk with a friend, constitute the one true carbon-copy of language. Other sorts of corpora, such as those related to academia or journalism, are disregarded as artificial, not reflecting real linguistic use. This is clearly disingenuous, however. The subject matter of linguistics, surely, is whatever mental capacity accounts for *all* types of linguistic behaviour. A university lecturer who enters a room and utters a rather complex, even unparsable, self-embedded sentence is exercising his linguistic capacity as much as the pedes-

[7]No-one really does; rather, language is considered a biological system that happens to me amenable to being studied mathematically, a different matter.

trian is in his inconsequential and evanescent chattering. The crucial notions for the linguist are grammar and its underlying properties; language and languages are at best derivative terms.[8]

Nevertheless, let us assume for the sake of the argument that there are languages that do not make use of self-embedded structures. Even if this were to be so, it is yet to be shown that the general, recursive SHC (or HC) structure is not a feature of every language —Pirahã clearly exhibits it, for instance. Indeed, take the title of Everett (2010): "You drink. You drive. You go to jail. Where's recursion?". The implication is that there is no self-embedding in these simple sentences, and furthermore, that Pirahã does not allow their combination into one complex sentence. However, a simple sentence like *you go to jail* is an asymmetric structure that is composed, on one side, by a determiner phrase, while the other side exhibits an embedded verbal phrase which in turn contains a prepositional phrase. That is, a structure of the type [...H...[...H...[...H...[...H...]]]]; a recursive structure. It is perhaps in terms of the discrete infinity of sound-meaning pairs of this specific kind —(S)HC— that "universality" claims regarding the language faculty ought to be understood.

Other scholars take a slightly different stance, however. Hurford (2004) proposes that human language is unique in respect to two features: the thousands of arbitrary symbols (words) we can acquire and self-embedding. Ludlow (2009), on the other hand, suggests that while the language of both humans and animals is some sort of "expressivist" language, the former is augmented with self-embedding. Some of these properties (namely, the large depository of lexical items and expressivism) may well be prerequisites for the discrete infinity language manifests, but surely it is the full generative extent of the language faculty that constitutes the unique feature of our species.[9]

In section 2.3, it was briefly mentioned that even if a language like Pirahã really lacked self-embedded expressions, its speakers would not be at an expressive loss, as the same sort of thoughts could still be articulated in alternative ways. That is, even if a self-embedded sentence were to be unavailable in a given language, the corresponding self-embedded thought could still be entertained in conceptual structure, a position that is defended in Everett (2010). Everett, then, does not want to take recursion out of cognition, just out of the grammars of particular languages. This is a particular take on the centrality of recursion within the language faculty, to be sure, as it centres the issue on particular grammars, rather on the primitive principles of *capacity* for language. Consequently, Everett's proposal turns out to

[8]This shortcoming afflicts most of the "functional" approaches to language, with their scorn for made-up examples and sentences, as if these were not the output of the same capacity that underlies normal linguistic usage. In short, language is primarily a mental phenomenon and it is to the study of the corresponding cognitive capacities and principles that linguists ought to direct their efforts.

[9]I ignore the possibility of whether linguistic structures other than syntactic ones exhibit self-embedding with no arbitrarily-imposed upper limit. Pinker & Jackendoff (2005) argued that phonological phrases are hierarchical but not recursive (cf. Neeleman & van de Koot 2006), a position that is contested by van der Hulst (2010a) and, in a slightly different manner, by Wagner (2010). See my Lobina (2011b) for discussion of other possibly recursive loci of the language faculty.

be based on a study of the grammar underlying "cultural cognition", as he seems to believe that it is cultural phenomena that establish properties of language (and perhaps of cognition overall). This would certainly appear to be a much more difficult enterprise, given how nebulous the very notion of culture is. Nevertheless, the general framework is rather straightforward in outline at least: to reduce linguistic phenomena to properties of general cognition. The field abounds with such proposals, and it is to this that the next section is devoted.

5.2 Non-linguistic cognition

As anticipated in the preface, it is possible to approach the study of general cognition by following a similar path to what has been provided here for the study of the linguistic capacity. As a start, we ought to draw a distinction between competence and performance. Thus, we would be interested, firstly, on the function in intension that generates the set of structures of the mental phenomenon under study, a *faculty*-like analysis that would focus on the underlying capacities and formal properties that enter into such a mapping. Naturally, the aim is to understand what sort of mechanisms and structures a given cognitive domain effects before embarking, and this is the second part, on a study of how these capacities and structures are put into use —the domain of performance.

In the preface, I also briefly defended the postulation of a "language of thought" (Fodor 1975, 2008; LoT), a domain-general, conceptual representational system in which thought is couched in terms of concepts and the way in which these combine into more complex structures. As will be recalled, assuming a LoT constituted an explanation for how the acquisition of natural language could be at all possible. That is, it was there observed that in order to acquire a representational system, an organism must be able to entertain, beforehand, whatever relations this to-be-acquired scheme exhibits. One way to describe this state of affairs, following Fodor (1979), is to imagine an organism that possesses something like a propositional calculus as its representational system. Clearly, it would be well-nigh impossible for such an organism to invent/acquire the predicate calculus, as the primitive operations of the system it has cannot represent the structural relations of predicate logic; mutatis mutandis for the acquisition of language (call this the "impossibility argument").

This is not to deny that acquiring a specific language often results in rather particular effects in the cognition of its speakers. However, it does not follow that an increase in "expressive power" takes place —i.e., that acquiring a language allows one to entertain thoughts that could not in principle be entertained in conceptual structure. That is, postulating a LoT does not mean defending a "uniform" cognition —whatever that means— it just means postulating a uniform starting point. This is in fact a very basic principle, but it is worth emphasising, as it is constantly misunderstood in the literature. Evans & Levinson (2009), for instance, advance that postulating a LoT is problematic because 'languages differ enormously in the concepts that they provide ready-coded in grammar and lexicon' (p. 435), but this statement betrays a complete lack of understanding of the actual argument. Indeed, these scholars go on to catalogue data that purport to show that diverse languages encode concepts (and much else) differently, the implication being that it cannot be the case that 'linguistic categories and structures are more or less straightforward mappings from a pre-existing conceptual space' (Li & Gleitman 2002, cited therein). The point they seem to completely miss is that there is nothing barring particular languages from making use of *some* parts of the pre-existing LoT. That is, it is quite probable that languages differ in the mappings they effect with the conceptual structure, and this would explain the language-specific phenomena; the LoT, however, remains a necessity.[10,11]

It is fair to say, however, that the LoT still awaits a detailed competence-level analysis. While it is true that some of its most elementary features have been delineated (such as the nature of simple and complex concepts, systematicity, productivity, ecc.), this is a far cry from what it is known about the manner in which the language faculty constructs sound-meaning pairs. Indeed, it is simply not known what sort of properties and principles drive the derivations of simple concepts into complex structures, but these must surely exist if human cognition is capable of forming new concepts out of prior, simpler ones (as it clearly is). It is beyond the scope of the present work to undertake such a massive study, but the literature does contain some tentative ideas that point to what may be regarded as preliminary work in that direction. Here I will initially focus on material that pertains, I will argue, to basic principles of the computational system underlying the LoT, and then I will move to the study of nested structures.

Much like human language, the LoT can be described as a finite number of primitives —concepts— and a finite number of formation mechanisms that merges them into more complex structures, what I will call here "propositions", the units of thought. These propositions are to be understood in a strictly technical, computational sense (Pylyshyn, 1984), and while they are clearly related to what philosophers understand by a proposition —for instance, both constructs encode argument structure— they should not be considered the very same idea. As units of thought, these propositions play different roles: they are the atomic elements of decision-making activities, interface with other systems of the mind and are part of long-term memory (which would include whatever systems account for the faculty of imagination, among others).[12]

[10]This very point was already explicitly defended in Fodor (1975, p. 85), but Evans & Levinson (2009) seem to have missed it, despite citing this work in their critique. In actual fact, this citation is rather gratuitous, as what they actually quote immediately after is an excerpt from Pinker (1994) —a popular science book. They persist with this mistake in Levinson & Evans (2010) when considering the possibility that the LoT may possess self-embedded structures (the discussion therein is in relation to self-embedded linguistic sentences, which they refer to as recursive; I come back to this presently).

[11]It also seems to escape the attention of proponents of a strong version of linguistic relativity that if their views were true, they would not be able, qua speakers of a specific language, to explain, or indeed to understand, the radically different cognition of speakers of other languages.

[12]Newell (1980) calls these propositions "expressions", but the same properties hold. Note that it

Further, and as argued in section 1.2, cognition is best thought of as a computational process in which the manner that propositions interconnect with each other does not follow associationist principles —that is, successive order need not determine causality. On the contrary, I am postulating local, structure-preserving operations that imply, naturally, an internal hierarchical organisation. It is also important not to forget that a LoT is supposed to account for the flexibility of human cognition, at least as manifested in the fact that while different bodies of information may well be processed in different modalities —that is, in a sui generis manner— combining them all in decision-making activities does not seem to be a problem. The LoT is, by definition, a modality-neutral representation scheme; therefore, when I talk of propositional interconnectivity, I am referring to the fixation of belief that encompasses the gathering of diverse percepts.

Whatever the actual details of the structural properties of the LoT, one of its mechanism must be able to read, write and transform structured representations (Pylyshyn, 1984, 1989). In addition to this, there must be a component in charge of ordering the operations that apply over the structures: call this the *control* unit. Such a mechanism would not only monitor sequencing action from point to point, it would also transfer control to a lower locus —self-reference— developing sub-routines. A sub-routine can in turn send control to other sub-routines, and a hierarchy of nested operations naturally develops. Once each sub-routine is completed, control is sent back up to where it was transferred from, and so on until it reaches the highest *control* operation.

The hierarchical nature of *control* operations has been the focus of two classic papers of cognitive science (Simon, 1962; Newell, 1980), but G. A. Miller, Galanter & Pribram (1960) was perhaps the first attempt to outline a detailed model of serially-ordered compositional systems that postulated a specific *control* operation: the TOTE (test-operate-test-exit) units (see Fig. 5.1).[13]

The cycle of operation of a TOTE is rather straightforward. *Test* obtains a representation of the problem, *operate* carries out some activity, effecting a change, and *test* then checks if the desired result has been obtained. If it has not, *operate* reignites the cycle again until the required output is returned, with *exit* terminating the overall process. TOTEs are, therefore, based on feedback loops (self-reference), and they can as a result be nested into other TOTEs, making it ideal for solving complex tasks divisible into functionally-equivalent but simpler subtasks. It has been recently described as an instantiation of the Standard Account in early cognitive science (Samuels, 2010) —a "plan, then execute" model of behaviour— and it is ideally suited to account for the hierarchical organization of the cognitive architecture. Furthermore, this model allows for MetaPlans (G. A. Miller et al., 1960, p. 169), that is, plans supervising other plans, resulting in a complex sys-

is the propositions that are retrieved from long-term memory, not the individual concepts themselves.

[13]Cf. Lashley's (1951) "logical and orderly arrangement of thought and action". I might also add that Chomsky (pers. comm. February 2007) considers G. A. Miller et al. (1960) to be the only example of a study of recursion in non-linguistic domains.

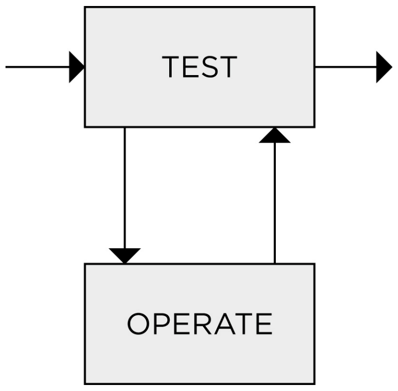

Figure 5.1: TOTE unit

tem that is analysable into successive sets of sub-systems (Simon, 1962, p. 468).[14] G. A. Miller et al. (1960) provide various examples of how this strategy can be employed in order to construct "plans of behaviour", and such analysis plausibly pertains to a competence-based analysis.[15,16]

As stated, it is the "plan" part of the Standard Account that would pertain to a competence-analysis, not the "executing" stage —the latter would effect a performance model.[17] This is not a spurious point, however. Earlier, I corrected Hunt's contention that most models of cognitive psychology are based on a computational level that is then tested experimentally. Instead, I claimed that to-be-tested computational models are in fact detailed descriptions of processing modules, and not a *theory of the computation*. A similar situation may be obtained in the application of the framework outlined in G. A. Miller et al. (1960). That is, it could be claimed that TOTE units are (fine-grained) descriptions of the actual online mechanisms that operate in performance, and it is certainly not clear at this point what, if any, could tell against this possibility.

[14]Clearly, MetaPlans subsume meta-cognition, the ability to monitor cognition itself (see Nelson 1999 for some relevant discussion). Further, note that *control* is clearly recursive, as pointed out by Newell (1980, p. 166).

[15]It is interesting to note that G. A. Miller & Chomsky (1963) already pointed out how analogous "plans" and syntactic trees were. In a forgotten but spot-on passage, they consider it not an 'accident that a theory of grammatical structure can be so readily and naturally generalized as a scheme for theories of other kinds of complicated human behavior' (p. 488).

[16]In a *New Yorker* report on his work (published on the 16th April 2007), Everett makes much of his "discovery" of Simon's (1962) paper on mental architecture and complexity, a work that suggests to him that recursion is not a distinctive feature of language, but of cognition —a rather conceited take on things, given that the analogy between plans and syntactic trees had been proposed more than 50 years ago.

[17]In this vein, G. A. Miller et al. (1960) remark that the execution of a plan must necessarily rely on working memory (p. 65), but this is not so for "planning".

In a related note, Rohrmeier (2011) offers one of the very few rule-based accounts for (a sub-part of) music, an approach that proposes a 'core set of grammatical rules to cover the fundamental features of the recursive structure of tonal harmony' (p. 36). The analogy with language is explicitly intended, and while Rohrmeier points to the "structural parallels" between linguistic and musical structure, he is well aware that dependency relations in music are strictly temporal in nature, which is not the case for the atemporal, abstract derivations of the faculty of language (ibid., p. 49). A difference between competence and performance to be sure; I will briefly come back to the specific case of music below.

Note, furthermore, that Fodor (2008) frames his theory of the LoT in similar terms; thus, there is a *thought* stage ("plans", roughly) that precedes *action* (the "execution"), but this take on things is clearly focused on "mental processes" —the real-time implementation.[18] As has been argued in this essay, however, a competence analysis explains why linguistic sentences are understood in a certain way and not another —that is, it attempts to unearth whatever properties and principles conspire to bring this about— but the mechanisms that are operative in understanding and production are quite clearly much different in detail. In spite of that, a much more careful study could provide a clear competence-level analysis for the LoT, and this would greatly benefit the field.

Such a study would not attempt to explain, for instance, the data that appear in too many a psychological study on rationality. Rather, it would follow a similar path to that which linguistics has undertaken for the last 50 or so years. That is, given the range of beliefs that our cognitive systems generate, what is the nature of the underlying structures and capacities that explain why these systems generate a specific structured set instead of another (viz., one that is unattested)? In the specific case of studies on rationality, the focus would not lie on why subjects behave irrationally —given an agreed-upon standard, be this logic, probability theory or else— but on how our cognitive systems generate the specific set of thoughts and beliefs that these experiments elicit —a slightly different take. Be that as it may, G. A. Miller et al. (1960) constitutes one of the clearest, and perhaps one of the very few, examples of a study of recursive mechanisms in general cognition.

Regarding whether recursion is a property of language or of general cognition, Fodor (2008) is quite certain of the 'recursive character of mental processes' (p. 106), and he is furthermore adamant that certain properties of language, such as productivity (discrete infinity, roughly), systematicity and compositionality, besides being parasitic on properties of the LoT, only require recursion in order to be explained (ibid., p. 105).[19]

There must be some truth to Fodor's position if the "impossibility argument" is at all sound. However, the relation between language and thought is probably

[18]Fodor (1975) has at times a peculiar view on the competence/performance dichotomy, one that seems to be based on a distinction between mental organisation and processing operations, but this is not entirely correct.

[19]L. Hauser (1995) and Jackendoff & Pinker (2005) make a similar point regarding a combinatorial and recursive LoT.

more subtle than what he suggests. Reinhart (2006) discusses some related issues when she divides the C/I interface into various systems: a repository of concepts, contextual information, and an inferential system. The latter carries out those operations so characteristic of decision-making activities, but it can only operate over propositions, it cannot manipulate concepts directly; that is, this system can only execute "rules of inference". Therefore, there must be a mechanism that mediates between the set of concepts and the inferential component: a computational system that combines concepts into propositions (this is graphically shown below in Fig. 5.2).

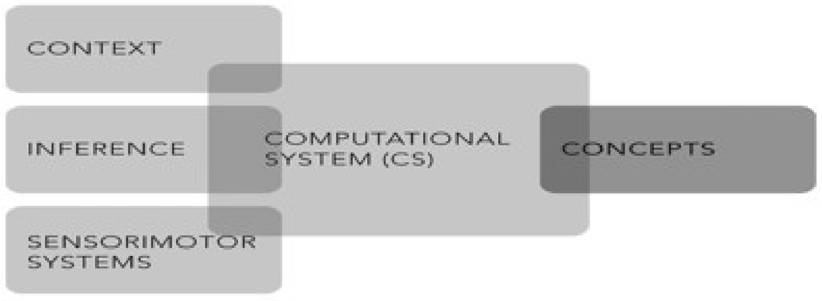

Figure 5.2: Mental Architecture

Reinhart (2006) proposes that it is the language faculty that effects this connection, but the evidence for this is rather thin. Linguists have certainly provided much evidence for how lexical items combine into sentences, but this does not transparently translate into an account of conceptual structure. In fact, Reinhart (2006) and Chomsky (2007a,b) seem to discuss the issues at hand in a way that collapses language and thought into one phenomenon, but it is one thing to state that the CS underlying language (i.e., *merge*) provides the mechanism for generating an unbounded number of structures, it is another thing completely to conflate lexical items and concepts into the atomic units this system operates over. Rather, it is very possible that the CS is in fact domain-general, but behaves differently according to the intrinsic properties of the elements it operates upon —lexical items and concepts *are* structurally different.[20] Thus, it is perhaps not entirely justifiable to reduce linguistic properties to general cognition; rather, the language faculty *shares* certain components with other systems of the mind, even if different cognitive domains may well be, ultimately, the result of a sui generis collection of systems and properties.[21]

[20]Indeed, lexical items are bundles of phonological, semantic and syntactic features (see Radford 2004 for some details) that are quite different from the intrinsic properties of concepts (see the introduction to Margolis & Laurence 1999 for a description of the latter).

[21]Hence, the analogy between "plans" and syntactic trees. This is closely related to Aristotle's

As mentioned in chapter 3, it is in this precise sense that I understand Chomsky's contention that the language faculty is ultimately a domain-specific system. In relation to this, it might be interesting at this point to discuss the views expressed in Brattico & Liikkanen (2009) and Brattico (2010), as these authors discuss these very issues. Three main points seem to underlie their discussion. Firstly, they define modularity as a proposal in which different (cognitive) domains are said to have their own "generative engines" (Brattico & Liikkanen, 2009, p. 251), an interpretation that they apply to the "modular" approaches of both Chomsky and Fodor. It is, however, not quite right to conflate the theories of these two scholars under one phenomenon, as both Chomsky and Fodor have pointed out in various places; Chomsky, for instance, has repeatedly remarked that Fodor's modules are "input systems", whereas his own work has instead focused on the initial and attained states of cognitive systems (e.g., in Chomsky 2000a, p. 20).[22] Secondly, Brattico & Liikkanen (2009) see the progression that the theory of generative grammar has undergone from production systems to *merge* as an example of the replacement of domain-specific rewriting rules for a domain-general *merge*. Specifically, they see the whole *generalized transformations* machinery as a domain-specific phenomenon, while the *bare phrase structure* that *merge* is said to output indicates, to them at least, the vestiges of a domain-general reality. The latter is a rather puzzling statement, as it is based on the belief that a *minimalist* account of language does not define 'syntactic configurations. . . on the basis of absolute positions', but in terms of the '*relative* positions emerging from. . . the properties of the lexical elements' (ibid., p. 271). Surely, though, these "properties of lexical elements" are specific to language; a fortiori, since it is these features that run linguistic derivations, the resultant computations cannot be anything but particular to the language faculty, as clearly evidenced in derivation trees.[23] On a similar note, Gallistel (2006) proposes a TM-like learning mechanism that remains unchanged from species to species, even if the calculations and outputs it effects differ depending on the contingencies of each species —namely, the representations each species has. That is, a domain-general CS that behaves in idiosyncratic ways as a result of manipulating particular symbols; a domain-specific productivity stemming from a domain-general mechanism. As a colophon, Brattico (2010) puts forward a *recur-*

potentialities, as mentioned supra. As Moravcsik (1975) remarks, according to Aristotle behaviour is explained by underlying "dispositions" and the latter must be the result of structural or constitutive differences (p. 628); that is, if two things have different potentialities, they must differ in their parts or arrangements (ibid.).

[22]Brattico & Liikkanen (2009, p. 252) link this definition of modularity to Fodor's language of thought hypothesis, and so the claim becomes that each cognitive domain is said to have its own language of thought. Whilst it is true that Fodor (1983) proposed that each processing module operates over a sui generis vocabulary, the language of thought is meant to be a unique, modality-neutral representation system. In fact, nowhere does Fodor suggest that there might be different *productivities* or generative engines.

[23]Furthermore, there is no reason to believe that rewriting rules are specifically linguistic in nature; after all, they can operate over any type of variables. That is, an S rule that is rewritten as NP+VP is not intrinsically linguistic; it is a just rule that is manipulating linguistic material, a different matter altogether.

sion hypothesis, the proposal that there is but one generativity capacity in human cognition.[24] He sees a connection between this proposal and Newell & Simon's general problem solver (GPS) model (ft. 2, p. 218) —evidence that the general theory of recursion/generative engine/productivity was never abandoned, he tells us— but fails to notice that the purview of the GPS bears no relation whatsoever to what Fodor's input systems or Chomsky's language faculty range over; a fortiori, there is certainly no "general theory of recursion" in these terms.

The main problem with their discussion is that they employ terms like productivity, generative engine and recursion interchangeably, but unfortunately these constructs go pretty much undefined in their discussion. If we take productivity to be the set of generable structures of a given domain (and if the set is infinite, it would constitute a discrete infinity), there is no conflict with the proposal that different domains employ the same generative engine (if the latter is understood as whatever mechanisms/operations generate structures from atoms). However, even if there are grounds to identify the generative engine of the language faculty with a recursor, recursion plays a rather different role in both Fodor's input systems and the GPS. Indeed, recursion would only be evident in the latter constructs in terms of the self-call operations of the *control* component, but whether recursive *control* is a general feature of various (or all) cognitive domains is not at all clear. Furthermore, we should not forget that faculties and modules *are* different mental realities, and so these very properties are likely to operate differently.

In a way, the general state of affairs outlined by these authors is not so different from what seems to have always been the case anyway —general CS, domain-specific computations— but I do not think their history of cognitive science in general and linguistics in particular (i.e., from domain-specific productivities to domain-general computational systems) stands up to scrutiny. In any case, the relevant facts about mental organisation remain unchanged, and the *Cartesian modularism* they defend seems to me an unnecessary terminological addition.[25]

Notwithstanding all that, I suppose that many scholars would contest my statement that G. A. Miller et al. (1960) constitutes one of the very few studies of recursion in general cognition. Michael Corballis is sure to protest that he *has* offered myriad examples of recursion in non-linguistic cognition. In actual fact, what he has done is point to some possible examples of non-linguistic self-embedded struc-

[24]Rather disappointingly, Brattico (2010, p. 216) defines recursion as a process that applies to its own output, a definition he claims applies to cognitive science in toto, even though no references are provided. In any case, there is in fact a lot of overlap between the two papers I am discussing, but while one focuses on productivity, the other is centred on recursion. One man's freedom fighter is another man's terrorist, they say.

[25]Very roughly, Cartesian modularism allows for a domain-general CS and non-generative "modules" such as the lexicon (a view of the lexicon that many will no doubt contest). It is rather startling, however, that they shy away from adding linguistic properties to the mix, but I fail to see how bare phrase structure or lexical features (the latter they call strong "lexicalism") can be anything other that particular properties of the language faculty. I also fail to comprehend their belief that the combination of all these systems 'lessens the pressure on language-specific productivity' (Brattico & Liikkanen, 2009, p. 275).

tures, but discussion of recursive generation or processing has been rather thin on the ground.

Corballis (2003, 2007b, 2011) chronicles some of the different self-embedded structures of the mind, Theory of Mind (i.e., belief ascription; ToM) perhaps being the most conspicuous case. Corballis (2007b) divides ToM abilities into two levels: *a*) zero-order theory of mind, i.e. mental processes such as thinking or knowing; and *b*) first-order, i.e. thinking or knowing what *others* are thinking or knowing, which involves recursion, according to him.[26] Self-embedding in ToM, then, involves the embedding of beliefs into other beliefs, such as in my ability to entertain the belief that Corballis seems to hold the belief that he understands recursion rather well.

There is some empirical evidence that children go through certain stages in the development of ToM structures (a chain of events that constitutes an alternative sub-division of ToM abilities, incidentally): contiguity, action, one-loop recursion, and two-loop recursion (P. H. Miller, Kessel & Flavell 1970, from which Fig. 5.3 below is adapted). Furthermore, the experiments reported in P. H. Miller et al. (1970) suggest that children's abilities in understanding self-embedded sentences and self-embedded beliefs/desires is almost concurrent, even if comprehension of "recursive" beliefs/desires appears to start a bit earlier (Oppenheimer 1986; see also Eliot et al. 1979). Still, at best these experiments tell us something about how children *represent* self-embedded beliefs/desires, but almost nothing about how they are generated or processed.[27]

Other loci of self-embedded structures, according to Corballis, are to be found in those capacities involving episodic memory —as exemplified in *I know I experienced X*— or in the apparent hierarchical conceptualization of tool making (viz., in the ability to use a tool to make another tool). Visual cognition is apparently yet another such domain, according to Jackendoff & Pinker (2005). Therein, they present an array of asterisks organised in columns and rows to make the point that for, say, a grouping of five asterisks, the ones in the middle could be interpreted as being embedded into the bigger series (p. 218). There is, however, no obvious internal hierarchy among the asterisks, apart from Jackendoff & Pinker telling us that we *could* entertain such an interpretation —a rather banal point, it seems. Still, this pertains to what Luuk & Luuk (2011, p. 5) call "parallel interpretation", an important feature of cognition (see Fig 5.4 for some examples).[28]

[26]Corballis (2011) allows for more levels of ToM embedding; the fifth is particularly important, as it may explain people's belief in God (pp. 137–8). It is perhaps without irony that he concludes this book by stating that he has 'primarily focused on human imagination' (p. 226).

[27]In relation to this, the reasoning Corballis follows is too widespread for comfort. He usually starts by describing recursion in language in terms of recursive rewriting rules; the resultant structures are then defined as recursive —conflating (inter alia) mechanisms and structures; and finally, he moves on to similar structures in other parts of cognition, but this time with no reference whatsoever to the mechanisms that generates them (or that operate over them).

[28]In order of exposition, Fig 5.4 shows: a Droste picture in which the overall picture is repeated on the cover of the cereal box, supposedly ad infinitum; dolls inside another dolls; and finally, triangles inside other triangles, all of them embedded into one big triangle. I will come back to this

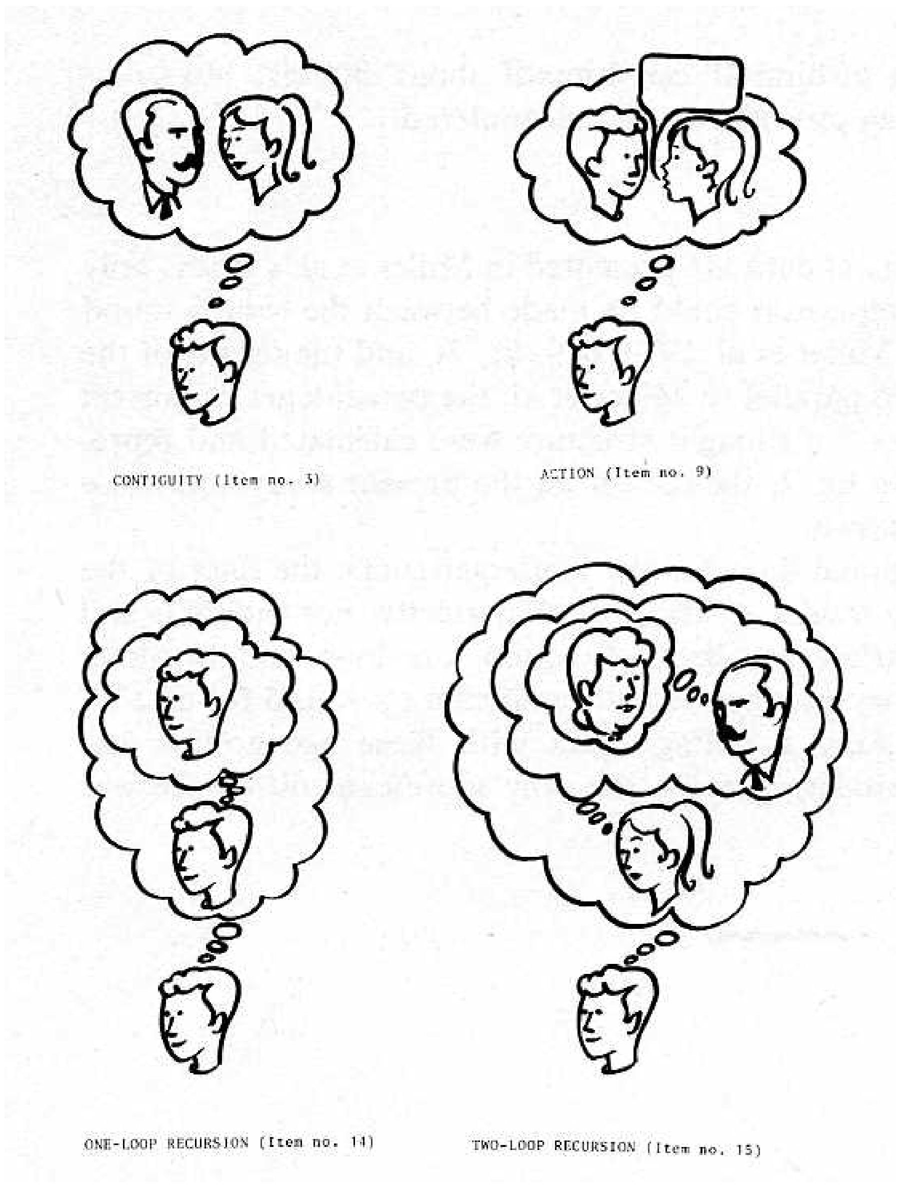
CONTIGUITY (Item no. 3)
ACTION (Item no. 9)
ONE-LOOP RECURSION (Item no. 14)
TWO-LOOP RECURSION (Item no. 15)

Figure 5.3: The development of Theory of Mind

The cognition underlying episodic memory introduces the possibility of a hierarchy of memory systems, and perhaps in a related manner, the question of mutual knowledge, joint attention and self-awareness. Peacocke (2005) advances an in-

phenomenon below when I briefly discuss whether we process music recursively.

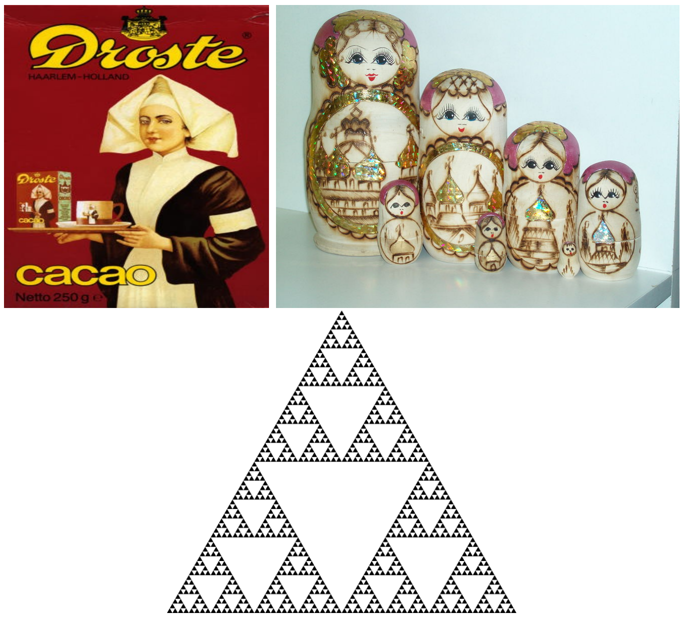

Figure 5.4: Self-embedding in Visual Figures

teresting discussion of some of these issues. Following the work of David Lewis and Stephen Schiffer, Peacocke defines mutual knowledge in terms of embedded *know that p* relations; to wit: *x knows that p*, *y knows that p*, *x knows that y knows that p*, ecc. Similarly for joint attention, *x perceives that x and y are attending to o*, *y perceives that x and y are attending to o*, ecc. (ibid., pp. 300–1). Despite the structural similarity, Peacocke notices, the mind cannot entertain an infinity of embeddings starting with *perceives that*, given the computational limitations of human psychology (p. 303). He consequently draws a distinction between the inferential case (mutual knowledge) and the perceptual awareness case (the joint attention), a dichotomy he connects to the relationship between the perception of language and linguistic competence (on the grounds, I gather, that an inference is isomorphic to a linguistic derivation). Perceiving that a sentence is grammatical is the result of (inter alia, we can presume) the unconscious operations that employ the information stated in the grammar (p. 311). Therefore, tacit knowledge and inference are entirely compatible with the fact, for Peacocke at least, that the final state is a perceptual state (ibid.).[29]

Naturally, all these examples make reference to a special case of hierarchically

[29]It is also interesting to note that Peacocke states that awareness of something is always different from what is an awareness of, which would discount the possibility of *awareness of itself* from our cognitive repertoire.

nested structures —self-embedding— a phenomenon that may perhaps be subsumed into a general class: simply put, thoughts can be embedded inside other thoughts. In a rather stronger strand, though, Corballis (2011) defends the idea that self-embedded linguistic structures may be reduced to domain-general structures; that is, hierarchical structures may be a domain-general feature. This is, however, unwarranted; linguistic and conceptual structures may share, in very general terms, the property of self-embedding or nesting, but they have a completely different organisation —i.e., they are not isomorphic in any meaningful sense.

The asymmetric geometry of phrases has featured extensively here, a property that it is most clearly manifested in the observation that the subject position of a sentence such as *the dog chased the cat* —viz., *the dog*— is hierarchically more prominent that the unit *chased the cat*. Suppose that this asymmetry is also present in belief-ascription structures such as *I believe that Corballis believes X. . .* , which is perhaps demonstrated by the fact that a propositional attitude verb like *to believe* takes scope over whatever it ranges over (in the case at hand, X), but clearly X does not take scope over *I believe that Corballis believes*. Still, the asymmetry of a sentence is also operative in its internal phrases in rather intricate ways, and it is not obvious that ToM structures manifest anything remotely similar. A self-embedded structure such as *[the mouse [the cat [the dog chased] bit] ran]* exhibits a level of interconnections among its internal constituents that ought to distil any belief that they could be reduced to superficially similar domain-general structures. Indeed, note that while *the cat* is the subject of *bit*, it is also the object of the internal phrase *the dog chased*; in turn, *the mouse* is the subject of *ran*, but it is also the object of *the cat bit*. This is of course the result of embedding asymmetric structures inside other asymmetric structures, but the point should not be shunned: linguistic structures have a sui generis architecture, irreducible to anything else (not to mention the derivation trees).

Clearly, the corresponding ToM beliefs do not exhibit a similar structure; in the example above, the X in *I believe that Corballis believes that X. . .* does not enter into similar relations with either *Corballis* or *I*. In terms of SHC structures, the ToM could be construed as effecting a S-HC asymmetry, but the organisation of its internal constituents is unlike that of natural language expressions.

In this context, it is worth mentioning the case of Genie, the feral child who, as a result of the abuse she suffered from her father, started learning English rather late in life. As Curtiss (1977) shows, Genie's general cognition was much more advanced than that of an infant acquiring a language, but this did not convey any advantage to her —in fact, Genie was demonstrably incapable of fully acquiring the English language. The non-isomorphism between linguistic and non-linguistic structures was perhaps clearest in her attempt to understand and produce self-embedded sentences, a task that proved extremely difficult, even if her ToM abilities were within normal parameters. Eventually, she managed to correctly understand self-embedded sentences, and even her production was close to that of normal people, with one particularity (pp. 158–9). Genie's output contained sentences with a V–N–V–N–V structure in which a noun would constitute not only the subject of the following verb, but also the object of the previous verb. These are, effectively, relative sentences without any relative markers, such as *I want Mat is present*, where *Mat* is the object of *to want*, but also the subject of *to be*. Putting all this together, then, it is hard to see a non-accidental connection between the linguistic capacity and general cognition.

Regarding the role of recursion in language and thought, then, it is very likely that the underlying CS is a domain-general component, even if the corresponding structures are very different indeed. In the case of the language faculty, the union of recursive generation, lexical items and the interfaces —a sui generis conglomerate— yields a discrete infinity of sound-meaning pairs that is completely unattested in other domains of the mind. Scholars like Corballis see deep-rooted similarities in different systems, but this only appears to be evident once we simplify the nature of the systems under study to, quite simply, unenlightening shallowness.

Another matter completely is the role of recursion in performance. As argued in chapter 4, there are two possibilities: the perceptual systems may either be sensitive to a given recursive structure or they may effect mechanisms that create chains of deferred operations, with the concomitant effect on working memory. It is perhaps at the performance level that we might find closer connections between language and general cognition, given that we are probing the perceptual systems, an independent component of the mind that is likely to remain uniform across diverse cognitive phenomena. Again, these systems are likely to operate differently depending on the type of structure we focus on, but this is a matter for demonstration.

The overall strategy of chapter 4 can be usefully employed in other domains; that is, an on-line experimental technique to probe the load of working memory can yield some information regarding the character of the operations in play. In relation to this, recall, once again, the artificial grammar learning (AGL) paradigm.

As mentioned earlier, Fitch & Hauser (2004) were interested in probing the expressive power of the grammar that subjects had internalised, and they quite explicitly stated that they did *not* study the different strategies that could have been employed, the "performance variables" (ibid., p. 378). Subsequent studies, however, attempted to probe if subjects were literally and directly employing the corresponding grammars in the processing of the different strings, with 'true recursion' being demonstrated if the subjects were to realise that some *As* are paired with some *Bs* within the A^nB^n strings (Corballis, 2007a, p. 702). I already showed that such an eventuality would mean that subjects are sensitive to the "recursive" structure, not that they are processing such strings recursively. Nevertheless, the actual results the AGL literature reports regarding whether subjects meet Corballis's condition are equivocal.

On the one hand, some studies conclude that subjects are not capable of processing long-distance dependencies, focusing on partitions and chunks instead (Poletiek, 2002; Perruchet & Rey, 2005). Other studies report that subjects are indeed able to process long-distance dependencies (viz., Friederici et al. 2006, Bahlmann, Schubotz & Friederici 2008), but these claims are controversial. Regarding Friederici et al. (2006), it is uncertain that the behavioural results they document in fact indicate this; rather, this conclusion seems to be based on their brain imaging data, which purports to show that the frontal operculum is activated during the processing of both finite-state and context-free strings, while Brodmann's Area 44/45 (i.e., Broca's area) is additionally only activated during the processing of context-free strings, an area they take to be operative in hierarchical processing. de Vries et al. (2008) replicated this and the other experiments mentioned above, and found no evidence for the conclusion that subjects were in fact processing the hierarchical structure of the strings; instead, they could have merely counted the *As* and matched them with the *Bs*, failing to meet, I suppose, Corballis's condition for "true recursion". It is only in Bahlmann et al. (2008) that we find a more conscious attempt to match the corresponding pairs by employing the phonetic features [voice] and [place of articulation], that is, by making sure that A_1 and B_1 share the same features, and so on for the rest. As a consequence, they claimed, subjects were prevented from counting and matching, which seems to have been borne out in the results. The neuroimaging data of Friederici et al. (2006) were replicated, and this suggests, to them at least, that 'the activity in [the latter] regions [is] correlated with hierarchical structure building' (Bahlmann et al., 2008, p. 533). Naturally, hierarchical structure building does not mean recursive structure building, and even less the correct processing of recursive structures.[30,31]

Unlike much of psycholinguistics, however, none of these studies venture to postulate any of the processing operations that surely are at the heart of these abilities. Rather, the default position seems to be that the underlying grammar is operative in some direct way, but no reason to believe this has in fact been offered. Perhaps more tellingly, there is no reason to believe that any of the AGL strings require a recursive process at all. Technically speaking, rewriting rules return sequences of elements, meaning that any associated hierarchical structure is an *added stipulation* that does not arise from the particular rules. This is a shortcoming that affected the employment of rewriting rules within linguistics too, but while it was obvious that linguistic expressions were structured, for reasons other than the actual generative mechanisms employed, no such thing can be said about artificial

[30]Fitch & Hauser (2004) presented the *As* with a male voice and the *Bs* with a female voice, while Perruchet & Rey (2005) employed a high- and a low-pitch, respectively. This quite possibly did not result in *AB* pairs; rather, it is very likely that subjects were sensitive to the changes of voice and to the pitch transitions rather than to the structure.

[31]See my Lobina (2011a) for a fuller description and a critique of the overall AGL project. Cf. de Vries, Christiansen & Petersson (2011), Friederici et al. (2011) and Folia et al. (2011). These three studies, however, share the unfortunate belief, in my opinion, that syntax is nothing more than "structured sequence processing". As I have argued throughout this thesis, the rules of syntax so evident in the sound-meaning pairs the language faculty effects are purely computational in nature, while the processing of language is very much dependent on properties of the perceptual systems, some of which do not appear to be purely computational. On a related note, one may well point to programming languages in this context. Even though computer languages are eventually executed in real-time, the programmer must first learn the "rules of syntax" the user's manual describes before "structured sequence processes" can be at all implemented.

strings of nonsense syllables. Granted, $A_3A_2A_1B_1B_2B_3$ strings are presented in a certain order, with certain cues, so as to force a hierarchical feature-linking operation, but this is a hierarchy among the different applications of the same operation. Present the string in another order, and it will result in a different hierarchy of these applications, but there is absolutely nothing to suggest that any of these strings are hierarchical, let alone self-embedded.

After all, why would anyone think that short, nonsense syllables sharing the same phonetic features are co-dependants across other short, nonsense syllables carrying a different feature? More importantly, why would the subjects interpret them as such? A fortiori, if subjects are really employing the language faculty, this is just a blind alley, as there are not in actual fact any languages that exhibit long-distance dependencies in these terms. In short, this self-embedding property of artificial strings can only be an unwarranted projection onto the data by the experimenter. No doubt that one could introduce many other (semantic or prosodic) cues so as to force a hierarchical interpretation of the strings, perhaps even approximating natural language expressions, but this is to betray the AGL paradigm and its attempt to abstract away from non-syntactic properties.

This is not to deny that the mind does seem to be predisposed to impose structure on the input it receives, regardless of any obvious cues. What I contest is the underlying assumption that our mental capacities are predisposed to assign a self-embedding interpretation to a given string because some experimenters define them a priori to have such a structure. Hence, it is not a surprise that there is much discussion in the AGL literature regarding the presence of bi- and trigrams in the data and the interpretative paths they lead into, and it is reasonable to suggest that AGL scholars ought to study what the mind does upon encountering these and why, rather than trying to avoid them in the search for a different set of results.

Obviously, it does not follow that there could not be any AGL tasks in which a recursive solution would be applicable, but this must follow from the three properties a recursive solution must meet (namely, the reduction of a complex problem into simpler but architecturally-equivalent subproblems whose union provides a solution for the entire task). If this were the case, it is the memory load variable that may help us to distinguish between recursive and non-recursive processes, as I discussed in chapter 4.

A different situation holds in the resolution of a Tower of Hanoi problem.[32] As Fig. 5.5 shows, this puzzle consists in moving the four disks stacked on the leftmost needle to the rightmost one by following two rules only: *a*) only one disk can be moved at a time, and *b*) each disk must be placed on a needle so that there is no smaller disk below it.

Crucially, the task can be resolved either recursively or iteratively, and I describe the recursive solution first.[33] The task is to move a tower of *n* disks from

[32]This puzzle was invented by the mathematician Èdouard Lucas in 1883, but it is also known as either the Tower of Brahma or the Tower of Benares. The multiplication of denominations is partly due to the legend that surrounds its origin.

[33]I am following Roberts (2006) in this exposition.

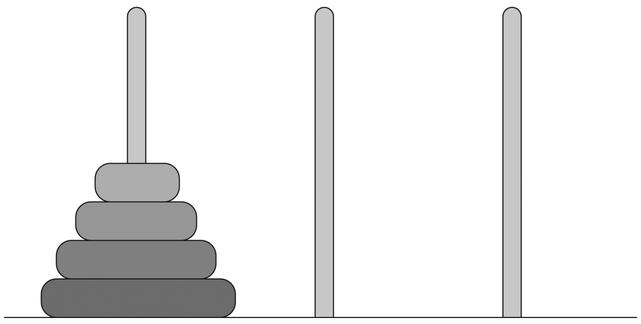

Figure 5.5: Tower of Hanoi

one needle to another. In order to do so, we can assign the following roles to the needles: **start**, the position in which the disks are initially found; **temp**, the needle that can be used to temporally stack one or more disks; and **finish**, the needle where all the disks will be moved to, in this case the rightmost one. If *n* is one, we move that disk from **start** to **finish**. If, however, *n* is greater than one, we can divide the problem into three sub-goals:

1. Move top $n-1$ disks from **start** to **temp**, using the **finish** needle as a temporary repository.
2. Move bottom disk from **start** to **finish**.
3. Move top $n-1$ disks back from **temp** to **finish**, using **start** for temporary storage.

In order to describe the non-recursive solution, it is useful to realise that the moves produced by the recursive algorithm yield many regularities. More specifically, when counting the moves from 1, the ordinal value of the disk to be moved during move *m* is the number of times *m* can be divided by 2. Hence, every odd move involves the smallest disk. This results in the following algorithm; in alternative moves:

1. Move the smallest disk to the needle from which it did not come.
2. Move another disk legally, where there will be only another possibility.

Crucially, the recursive solution is the only one that solves the task in the least possible number of moves, which shows the close connection between a recursive solution and hierarchically-structured tasks. Further, the recursive solution is at all possible because the task meets the three properties mentioned supra: the reduction of a complex task into equivalent, atomic subtasks whose combined resolution provide a solution to the entire, matrix task.

Naturally, the recursive and the iterative solutions differ in the memory strain exerted, and therefore manipulating this variable may shed light on the nature of the implementation that is being executed. Xu & Corkin (2001) employed this strategy in order to study the function of working memory in amnesiacs, but their purpose was not to figure out which strategy subjects naturally employ. Instead, they devised an experiment in which the subjects were to follow specific directions so that they did in fact attempt a recursive solution, the rationale being that this would overflow the working memory of short-term amnesiacs, as indeed was the case.

I am not aware of any experimental work that has attempted to employ a task such as the Tower of Hanoi to probe the possible recursive application of rules. There are many other "recurrent" problems that could be used in similar ways, some of which are catalogued in Graham, Knuth & Patashnik (1989) and Roberts (2006), but the cognitive psychology literature is rather thin on this sort of undertaking.[34]

The literature seems to be much more confident regarding the conceptualisation of self-embedded structures than in the possible corresponding recursive processes in perception. Indeed, the very possibility of "parallel interpretation" is an unambiguous demonstration that our visual cognition system can indeed conceptualise figures into other figures of the same type, but this is not evidence for recursive sub-routines.

A perhaps more relevant case is that of musical perception. Hofstadter (1979, p. 129) suggests that we perceive Bach's Baroque-style modulations recursively, given that the beginning of a C note may be followed by a D note that commences before the C modulation finishes, therefore giving rise to a hierarchy of note modulations; that is, something like this sort of structure: [D. . . [C. . . C]. . . D] (clearly, this is very similar to the hierarchy of feature-linking operations of AGL strings). This is just supposition however, and there is certainly a hint of the usual conflation between specific structural properties of, in this case, musical composition (embeddings of note modulations) and the processing of such structures. Fitch (2010, p. 82) also discusses this very issue, and it is interesting to note that he entertains the possibility of experimentally probing if our cognitive system carries out "pushdown" and "pop-up" operations during the processing of Bach's symphonies. Remarkably, this is rather different from his general framework, which involved, it will be recalled, devising experiments that probed whether subjects construct the right interpretation of supposedly self-embedded structures —his "empirical indic-

[34]Admittedly, these two sources cover these problems from the point of the view of computer science, but some of these tasks are easily adaptable for experimentation. Graham et al. (1989) describe two such problems: how to draw lines in a plane, and the Josephus problem (see pp. 11 et seq. for details).

ator" for recursion.[35] Naturally, the right interpretation of a self-embedded structure does not necessitate recursive processes. Nevertheless, this is yet more evidence that the human mind has and uses self-embedded structures, but there is no clear indication that real-time mechanisms (either perceptual or conceptual) operate recursively.

If the argument in chapter 4 is sound, however, what psycholinguists are in fact studying is the *manner* in which the perceptual systems respond to analysing and producing linguistic structures that are the result, in part, of intrinsic computational properties, a task for which the perceptual systems were most likely not designed to undertake (and mutatis mutandis for cognitive psychology overall).[36] Consequently, we can indeed expect very similar "structured sequence processing" phenomena across different domains, given that the perceptual systems remain uniform. It does not follow, however, that there is any strong relation among the different underlying systems of the mind that generate the corresponding structures (or more accurately, that there are any similarities between different conglomerations of principles and properties, the different cognitive domains). This is perhaps clearest in the results of a recent experiment showing structural priming between the interpretation of mathematical formulae and the understanding of complex sentences (Scheepers et al., 2011), as I now briefly discuss.

Friedrich & Friederici (2009) allude to the structural similarities of hierarchical mathematical formulae and linguistic structure, as exemplified in the following first-order language formula: $(a = c + u) \wedge (v \cdot x < u + y)$. Note that in order to output the correct value for this formula, it is necessary to proceed according to the syntactic rules that generated it —the internal operations must progress according to the hierarchy in which they are embedded. These rules of syntax establish two things, then: how to construct the right interpretation for the formula, and how to proceed to calculate the right output.

Similarly, Scheepers et al. (2011) report an experiment in which the interpretation of ambiguous strings was primed with prima facie similar expressions in mathematics. They employed sentences like *I visited a friend of a colleague who lived in Spain*, for which the language faculty creates two different structures depending on the interpretation. That is, the phrase *who lived in Spain* may either modify the noun phrase *a friend* —this would constitute a case of high attachment— or the noun phrase *a colleague* —low attachment. From an equation such as $80 - 9 + 1 \times 5$, $80 - (9 + 1) \times 5$ would be a case of high attachment, and $80 - 9 + 1 \times 5$ of low attachment. The rationale for this comparison is based on the precedence rules of mathematical operators —viz., multiplication and di-

[35]It is nevertheless very telling that Fitch (2010) does not discuss the possibility of providing a generative account for musical composition. As briefly discussed above, Rohrmeier (2011) has convincingly argued that a musical grammar makes use of recursive devices as much as a generative grammar for language, but Fitch seems fixated on the right interpretation of self-embedded structures.

[36]I say "in part" because the external manifestation of language is the output of the sensori-motor system, and consequently some of the features of the externalised product will reflect some of the limitations of this system.

vision precede addition and subtraction— and the employment of brackets. Thus, in the high-attachment equation the addition operation inside the brackets must be carried out before multiplication applies, whereas in the low-attachment case, 1 is multiplied times 5 before the remainder can be calculated.[37] The results of their experiments show that when the ambiguous linguistic string is preceded by a high-attachment mathematical equation, subjects prefer a high-attachment interpretation of the string, and mutatis mutandis for low-attachment equations and sentences.

Scheepers et al. (2011) consider these results to be a significant contribution to the field, as they claim to have unearthed the first case of 'cross-domain structural priming' (p. 1), another seed to the 'growing body of evidence for the domain generality of structure' (p. 7). Naturally, one would want to know in what sense the mathematical and the linguistic structures they focused on can be at all subsumed under a general representational scheme. The answer is at 'a very high level of abstraction' (p. 1), they tell us, but a moment's reflection speedily dissipates such grandiloquent claims; rather, the similarity they point out is only perceivable at a very high level of *simplification*. It has been a recurrent argument of this essay that language exhibits a very particular and intricate type of structure, unrelatable and irreducible to anything else, and it does not need to be repeated once more.[38]

More to the point, the experiments are perfectly explainable in terms of the strategies the perceptual systems implement for such extraneous inputs. Indeed, it is very likely that these experiments probed the preliminary stage of an analysis-by-synthesis processor. Considering that this stage effects such strategies as the employment of templates, it is not surprising, given the (simplified) similarity of the two representations, that the perceptual systems carry over the same strategy from one domain to the other. Furthermore, note that the manner in which the mathematical materials were constructed —using brackets to mark constituents— forced a specific interpretation, resulting in the subsequent strategy being employed, and therefore preceding (or indeed skewing) the application of the generative rules so characteristic of the synthesis stage. Still, if there is any domain-generality to this at all, it has to do with the general properties of the processor underlying "structured sequencing", at least at an early stage of processing. That these two representations cannot be subsumed under a broad type of scheme is immediately clear as soon as

[37]They employed two groups of subjects, business/mathematics students and psychology students. The former did not need to be reminded of the precedence rules, but for the latter Scheepers et al. (2011) had to introduce further brackets in the notation so that they would not make mistakes in solving the formulae. The modified equations were $80 - ((9 + 1) \times 5)$ and $80 - 9 + (1 \times 5)$, respectively.

[38]Another take on this issue eschews defending a domain-general representation in favour of suggesting that when you simplify linguistic structure to its bare minimum, you do have something akin to the mathematical structures herein discussed. If so, mathematics would be parasitic on the language faculty (this is Chomsky's take, effectively). I refrain from discussing this specific issue, but I do favour a "componential" approach instead, pretty much as defended in the Introduction for the study of the LoT. It should also perhaps be pointed out that Friedrich & Friederici (2009) found that different brain regions were activated during the processing of language and mathematics, perhaps indicating a qualitative difference between the two domains.

we focus on the intricate features of the sentences, such as the asymmetry between the subject and the verb-object compound, the (modifying) relations among noun phrases, ecc. In short, those features that would have to be reconstructed by the synthesis stage of the syntactic parser. What these results indicate, then, is a convergence of distinct domains at the preliminary stage of the perceptual systems; that is, closer to particular properties of the uniform perceptual systems.

In consonance with what I argued in chapter 4, the existing experimental work that is prima facie related to recursive processing is in fact informative of how the mind conceptualises the structural properties of the input. That is, we do not have unambiguous evidence for the existence of recursive sub-routines at the level of real-time implementations, and it is uncertain how we would be able to discern them, if they exist at all. Scholars such as Corballis and Fitch, then, are clearly focused on the correct processing of self-embedded structures, a different matter. There is no doubt that these scholars and their colleagues will in time furnish us with data on recursion in general cognition, but we can be certain that its range of application will be rather narrow in scope.

*****

There is much evidence, then, for hierarchical non-linguistic cognition, which may well be a unique feature of the cognition of humans, if Marcus (2001, p. 151) is right. The existence of hierarchical representations and structures is much more certain, and better studied, than hierarchical generation and processing, but as I have tried to show here, the literature does contain some tentative ideas that certainly need to be further investigated.

Some of the domains in which hierarchy appears to play a central role include visual cognition, tool making (and object manipulation), mathematical operations, musical processing, ToM abilities (and more generally, "social cognition"), and perhaps others. There is even some evidence for self-embedded structures in some of these domains, and it is likely that both constructs instantiate universal features (such as thoughts within thoughts).

It is important to note, however, that all these cognitive domains mandate operations over structures; that is, strong generativity. There is a tendency in the literature to focus on production systems when it comes to modelling mental abilities, and this is perhaps unfortunate. Rogers & Pullum (2011), for example, offer a panoply of "psychological correlates" between different grammars of the Chomsky Hierarchy and cognitive abilities, but this is surely a point about the expressive power of particular abilities, it cannot be an outline the *actual* mechanisms that effect structured sequencing.

There is perhaps a stronger tendency to model hierarchical domains, whatever they are, in terms of an underlying grammar, but the analogy with the linguistic capacity is rather weak. At best, the proposal of a grammar for, as it may be, cultural cognition, is nothing more than a reflection of some scholars' belief that a given phenomenon can be subsumed under an organisational scheme. It does not elucidate, however, the existence of a mental reality.

All in all, the language faculty remains a special system of the mind in both outline and detail. Having said that it is possible that none of its internal properties are specific to it; nevertheless, the aggregate of whatever elements turn out to be its primitive constituents results in a sui generis and rather complex cognitive domain; a specific, distinct *mental reality*.

Concluding remarks

"recursion definition, a euphemism for postulation"

A number of objectives or conceptual threads have converged in the present work. First and foremost, this essay has attempted to meet the expectation that doctoral work should defend a *thesis*. Here, two main opinions have in fact been advanced:

(I) The specification of what a computation is determines the cognitive domain (the mental reality) under study.

(II) Recursively-specified algorithms of mental faculties can, in principle, be reduced to iterative implementations by processing modules and are, in fact, reduced to such implementations.

As a second objective, I have argued that computational theories of mental faculties and modules ought to proceed in an orderly manner, and a three-stage explanatory approach was delineated for this very purpose. A framework for these characteristics starts by outlining the computational system at the heart of a given cognitive domain, and chapter 1 provided the necessary background for such an enterprise. Building on that, chapter 2 sought to establish the actual nature of the algorithm underlying the language faculty, which includes distinguishing the mechanical procedure from the structures it generates. In addition, the first chapter described some of the different formalisms that can appropriately describe exactly what a computation is, and argued that choosing one or the other goes some way towards determining the cognitive domain one is actually studying. In the present case, we drew a distinction between the type of computations that the grammar and the parser carry out, with chapters 3 and 4 thereby devoted to these two *mental realities*; that is, to the two types of implementations that the computational system underlying the language faculty effects.

Furthermore, it was also an important aspiration of this work to provide a conceptual clean-up of the manner in which recursion is interpreted and employed in cognitive studies, an extremely important goal given the parlous state of contemporary scholarship regarding this matter. More often than not, when scholars talk of recursion they actually mean self-embedding, either in the sense of structures (Christiansen & Chater, 1999; Everett, 2005, 2009; Pinker & Jackendoff, 2005; Heine & Kuteva, 2007; Christiansen & MacDonald, 2009; Sauerland, 2010; Roeper, 2011) or operations (Chomsky & Miller, 1963; Tomalin, 2007; Lobina & García-Albea, 2009; Stabler, 2010). In the case of the former, some authors presume that the centrality of recursion in language is to be explained by the role self-embedded sentences play in determining the actual expressive power of language (Sauerland & Trotzke, 2011). Such a perspective, however, is historically unsupported (that is, recursion was not identified with self-embedded sentences in the relevant literature of the 1950s and 60s) and the result of rather careless argumentation. In the case of the latter, some scholars define a recursive operation as one that applies over its own output (Di Sciullo & Isac, 2008; Hornstein, 2009), but this is instead a definition of a recurrent operation tout court. As I have shown in this thesis, recursion can certainly be appropriately applied to both structures and operations, but it is not uncommon for many scholars to conflate these two constructs, resulting in the widely-held belief that recursive structures can only be generated/processed recursively (e.g., Corballis 2007a; Friederici et al. 2006). This can only follow if a recursive operation is defined as one that embeds elements into other architecturally-equivalent elements, as many scholars do (M. D. Hauser, 2009; Kinsella, 2009; Fitch, 2010; Corballis, 2011). Even when recursive generation is correctly characterised in terms of inductive generalisations, the transformation of this concept into self-embedded structures is nevertheless carried out without noticing the obvious non-existent relation between recursive generation and self-embedding (MacWhinney, 2009; Arsenijević & Hinzen, 2010). Finally, the full generative power of language —i.e., the combination of recursive generation, lexical features and the interfaces it interacts with— is sometimes confused with recursion (i.e., self-embedding) per se, and as a result the literature contains a number of misguided attempts to reduce specific properties of language to domain-general cognition (Abe & Watanabe, 2011; Scheepers et al., 2011).

Instead, it has here been shown that the term recursion can apply to four different constructs, all of them underlain by the self-reference feature, and all of them carrying much sui generis technical baggage. Thus, we have (a) recursive definitions, which are likely to be useful in many disciplines (Bar-Hillel, 1953). Secondly, and more importantly, recursion is also a general and central property of algorithms and generative systems (b), such as in production systems, *merge* or in the iterative conception of set. Before moving on to the other two constructs, let me offer some comments regarding the actual issues at hand here.

Both mathematics and theoretical linguistics make ample use of these two interpretations, and the kind of work I have outlined here has attempted to provide a correct specification for the algorithm underlying the computational systems of the mind. More specifically, I have proposed that the algorithm at the heart of the language faculty is a recursor, while processing modules in general are iterators. This is supported by the empirical fact that i) the linguistic system generates an infinite number of binary, hierarchical structures that ii) are produced and processed in tandem with memory and complexity factors. As such, recursion is a central and general property of computational systems, irrespective of the actual nature of the structures they generate (cf. M. D. Hauser et al. 2002, p. 1571).

The analysis at this level also includes what is usually called the *theory of the computation*, an approach that focuses on the formal properties of the mapping functions of the domains that interest us. I have here focused on the language faculty partly because theoretical linguistics has conducted much more extensive work in these very terms than any other field of the cognitive sciences. Nevertheless, this sort of approach describes mental faculties in terms of *functions-in-intension* that may well be central to many other domains such as planning, reasoning, serial order learning, symbol sequencing and open-ended combinatorial manipulation, representation of number and math operands, musical processing, navigation, et alia (ibid., p. 1573).

Naturally, it is important to settle the terminology, as lack of rigour hampers serious research and blinds us to the correct repercussions. As I have tried to show in chapter 3, the uniqueness of natural language lies in the correspondence between a recursor and the interfaces it interacts with, yielding a potentially infinite set of recursive structures. Moreover, I have tried to show that this is the sense Chomsky has always had in mind when discussing recursion (in chapter 2).

It may take an enormous effort to convince scholars that recursive generation has intrinsically nothing special to do with embedding or self-embedding, but it is an undertaking that must nonetheless be carried out. At this point, I may venture to offer some historiographical reasons for this fixation on recursion and self-embedding. As described here, in the 1950's linguists correctly employed recursion in reference to specific rewriting rules. However, ever since their elimination from linguistic theory, most scholars have used recursion, rather puzzlingly, to refer to those structures that recursive rewriting rules were employed to generate. As such, many scholars apparently believe that these structures, having once upon a time been recursively-defined through the application of recursive rewrite rules, can still be so defined even though the original rewrite systems have been eliminated from the theory. If so, this state of affairs may be the unfortunate legacy of employing production systems in the first place.

At another level of explanation, recursion applies to (c) actual processing operations, as manifested in chains of deferred sub-operations. This sort of study focuses on an algorithm's actual *implementation*; that is, it is the study of the so-called *models of computation*, plausibly the purview of cognitive psychology. As I have tried to stress here, whilst it is a necessary condition for a recursive process that it apply to its own output, this does not in itself constitute a sufficient condition. It is in fact trivially true that it does so, but operating on its own output is a feature of recurrent operations in general, including iterations. Consequently, recursive and iterative processes differ in their *mode of operation*, but not necessarily on the *type of operation* they effect; a fortiori, recursion and iteration certainly do not differ on the sort of structures they (may) generate. Unless we know what sort of operation a mechanism executes, nothing at all follows regarding what sort of structures a recurrent operation creates. The conflation of these two properties and the extrapolation onto the resultant structures is perhaps the main problem be-setting communication among scholars on the role and application of recursion in cognition.

Indeed, it is a well-established, though often forgotten, result of the formal sciences that all tasks that can be solved recursively can also be solved iteratively (Rice 1965, p. 114; Roberts 2006). This point is significantly unappreciated in the cognitive sciences, due in my opinion to a number of simple misunderstandings that I have tried to tease out within these pages. In any case, this very point justified the experimental investigation we conducted here, as we identified a recurrent operation —viz., the analysis of linguistic input into SHC configurations— that a priori could have applied either recursively or iteratively. The data we obtained suggest that this operation applies iteratively, as there was no evidence of any deferred operations.

The last construct recursion applies to is (d) the structures the mind actually seems to have and use, and here I have pressed home the importance of what sort of general recursive structure languages universally manifest; viz., SHC structures, of which self-embedded sentences are a subtype. There are strong reasons to think that hierarchically-structured representations are unique in humans, but outlining the precise characteristics of the structures different domains exhibit is no easy matter. It is safe to state, however, that the structures the language faculty generates seem sui generis, being unattested in other cognitive domains (as a close analysis clearly shows; chapters 3 and 5). In relation to processing, and even though there is a natural a priori fit between recursive structures and recursive operations (as forcefully argued by Wirth 1986), whether real-time mechanisms operate over recursive structures in a recursive manner needs to be empirically demonstrated, particularly given the orbiting issues that are involved (memory load, architectural complexity, ecc.).

I should also mention a few issues regarding the formalisation and foundation of cognitive science. It was remarked in section 1.1 that the reduction of recursively-specified algorithms to iterative models of computation is considered by some scholars (viz., Moschovakis 1998, 2001, Moschovakis & Paschalis 2008 and Dean 2007) as the basis for a "founding" of mathematical logic. What these scholars understand by a "founding" is an attempt to define the basic notions of the different levels of explanation of a theory, including their relationship(s). As Dean (2007) points out, cognitive science is yet to receive such a founding, but its status as a computational-representational undertaking plausibly merits it.

I have certainly not tried to provide a foundation for cognitive science here, but this thesis *can* be viewed as offering a programmatic study for a future founding of generative grammar, the theory of the mental faculty of language. The work thus outlined can be described as a family of proposals by which we may provide this foundation, and recursion has been placed at the centre of this endeavour, much like as in the founding of the theory of algorithms (references op. cit.). After all, I have provided some evidence that one specific component of the syntactic parser —or more accurately, the nexus between the preliminary analysis and the first-stage of the parser— proceeds iteratively, even if the underlying *function in intension* is a recursor. If this is so, then we are rather close to computer science in an interesting way: computational systems require the reduction of recursive specifications of algorithms to iterative models of computation. Or otherwise put: reducing recursors to iterators (Dean, 2007, p. iii).

Naturally, a study of these characteristics has involved a fair amount of idealisation, and a number of constructs have been assumed and defended throughout: the fact that there is such a faculty for language; that it is underlain by a finite, mechanical procedure; that specific properties can be ascribed to this system, delimiting a clear space for *linguistic* knowledge in the mind; that this knowledge, together with its underlying grammar, are involved in linguistic behaviour; that this grammar is derivational in nature; ecc. In short, this essay has attempted to provide a foundational analysis of a set of conceptual issues for cognitive science: the role of recursion in the formulation of a computational theory of mental faculties and modules.

In any case, the present work has defended the following conclusions:

(a) Self-reference constitutes the original interpretation of recursion within the formal sciences, a property that can be applied to various theoretical constructs, such as definitions, structures, processes and the very notion of an algorithm.

(b) There is a difference between characterising a computation in terms of systems of recursive equations or as an abstract, mechanical device like a Turing Machine. In particular, the former are more amenable for a formalisation of the abstract implementations of an algorithm, while a Turing Machine is an ideal construct for the study of step-by-step computational processes. These considerations are very relevant for the construction of the explanatory theories that interest the cognitive scientist, as recursion is a central property of computational systems when these are construed as functions in intension —the purview of those scholars that study mental *faculties*—, while the Turing Machine model can be employed to analyse the real-time mental processes that so interest the cognitive psychologist —that is, the analysis of processing *modules*.

(c) Initially, recursion was introduced into linguistic theory as a property of the underlying function in intension (recursive generation), but much attention has been devoted to recursive structures, either in the general sense of Specifier-Head-Complement(s) configurations or in the narrower and more common sense of category recursion (when a phrase is embedded into a phrase of the same kind). It is very important to emphasise that recursive mechanisms and recursive properties should not be conflated; that is, it does not follow that if a structure is recursive, it has been generated (or can only be processed) recursively.

(d) At the level of competence, recursive generation conspires with the interfaces, lexical items and general computational properties to yield a sui generis set of structures, and no other cognitive domain appears to exhibit anything remotely analogous. Despite the recursive character of the underlying generative system, the actual syntactic derivations are iterative in nature. Similarly, the processing of SHC structures —a matter of performance— also applies iteratively, and these two factors perhaps hint at a very general property of cognitive systems; namely, the iterative implementations of recursively-specified algorithms, regardless of whether these implementations are abstract or proceed in real-time.

(e) Studies seeking to establish whether recursion is a universal feature of every natural language have so far exclusively focused on the geometrical features of derived tree structures, but this is a mistake, as the derivation tree structure, including the generative system that underlies it, is the key notion for linguistics. Similarly, attempts to relate (or even reduce) linguistic structure to domain-general representations only succeed if the former are greatly simplified, but a careful analysis clearly shows that linguistic structure is very particular indeed. Instead, it is reasonable to conclude that the human mind has and uses various types of hierarchical structures, which perhaps bear very little resemblance to each other.

Postface

The present work has considered a diverse number of topics and phenomena, all of which have been treated within the narrow confines of recursion and its different guises. Naturally, if we allow ourselves to look beyond the role of recursion, these very topics and phenomena yield new avenues for future research. I shall mention a few of these below, but my intention here is to describe those future research problems that relate to what was discussed in this thesis.

The first chapter began with a catalogue of the different ways an algorithm can be formalised, some of which have been employed within cognitive science to model mental capacities. As I argued there, choosing a specific formalism specifies, at the very least, the level of explanation one is focusing on, and I consequently outlined two different ways to study cognitive phenomena. On the one hand, a field such as linguistics attempts to (re)construct the theory of the computation underlying a specific mental faculty, while a discipline like psychology aims to discover the real-time implementation —the operations involved in the production and perception of language— of the underlying mapping function. Furthermore, these two levels of analysis are closely related to specific mathematical constructs: a recursor in the case of the former, and a Turing Machine in the case of the latter; that is, either an abstract application of an algorithm or its real-time implementation.

As far as I have been able to determine, the cognitive science literature has not made use of all the possible formalisms described in section 1.1. Nevertheless, even if a thorough literature review were to unearth studies utilising some of the formalisms I have been unable thus far to locate in the cognitive science galaxy, it is not clear that such models would expand my two-part division in any meaningful way (cf. Pinker 2005a,b). Still, exegetical exercises should never be shunned, as they are generally very useful for the field (and usually quite enjoyable for the scholar).

Perhaps more substantially, section 1.3 advanced an epistemological thesis regarding the place of the language faculty within mental architecture, and this is easily expandable to other systems of the mind. The argument defended therein suggested that it was entirely accidental that the language faculty came to be embedded within a collection of such diverse systems in the manner it appears to have been, a conglomerate I denominated the (overall) linguistic capacity (the grammar, the parser, the perceptual systems, working memory, ecc.). Clearly, the "casualness" of the language faculty requires a much more detailed analysis if we are to discern the concomitant repercussions of defending such a position. My intention here was merely to offer plausible reasons for such a view of things, given its close connection to the "problem" of language comprehension I outlined in chapter 4. Still, it is likely that an expansion of this point may turn out to be a worthwhile endeavour, and perhaps even a genuine development in the field.

As for other systems of the mind, I am particularly interested in the LoT, a construct that awaits a comprehensive theory of the underlying function responsible for intensionally generating conceptual structures (the propositions) —a research programme I hope to undertake in the near future. In chapter 5, I made reference (at least, implicitly) to three combinatorial mechanisms that must surely lie at the heart of any LoT. Plausibly, the most central of them all, by analogy with the language faculty, is whatever mechanism constructs complex concepts from atomic units; a (perhaps) domain-general *merge*. Intuitively, there must also be a mechanism that combines complex structures into the intricate "plans" and actions that enter behaviour (the TOTE units of chapter 5). Finally, it is necessary to postulate a component that relates language and thought somehow. In my view, the most promising way to study the language-thought relationship, at least at this precise moment, would be to look at how the elements that combine LoT propositions into inferential processes relate to the connectives that link up natural language sentences. According to this take on things, the focus would lie on the overall effects that acquiring and processing natural language connectives such as *and*, *or*, *if... then* or *not* have on general cognition.

It will be recalled that Reinhart (2006) identified an "inferential system" within the C/I interface, a component, she argued, that could only operate over propositions. She also proposed that it was the language faculty that mediated between the repository of concepts and the inferential system, a suggestion I doubted on the grounds that linguists had not in actual fact offered any reasons to believe this was indeed the case, let alone provide an account for the phenomenon (in fact, linguists have not even been very successful in working out how sentences are connected with each other via connectives *within* the language faculty). The question that seems to be of much relevance here is whether acquiring the connectives of natural language results in new ways of combining information, connections that would perhaps be unavailable in the LoT.

Naturally, the LoT must contain at least *some* connectives if we are to explain some of the non-linguistic abilities humans manifest —the cognition of pre-verbal infants being the most obvious case— but it is an open question how many of these it actually has or how they should be classified. Imagine that one were to propose that the LoT only has conjunction at its disposal (i.e., *and*); acquiring a natural language would, then, involve an augmentation in expressive power, as conjunction cannot derive all the truth functions (truth tables) that the four aforementioned connectives are capable of generating (truth functions that are clearly part of our psychology). On the other hand, it is common ground in studies of logic that truth tables can be used to prove logical equivalences between different expressions, and it is possible to employ two connectives (say, *and* and *not*) to express all possible truth values. In fact, there are two logical operators —Sheffer's stroke and NOR— that can, individually, express the truth values of any logical operator. Obviously, deriving all truth functions with one connective only (or even two) would result in rather cumbersome derivations (the expressions would be too long and heavily layered) compared to a system that employs all four, but these two systems would not differ in expressive power.

Imagine now that the LoT has something akin to a Sheffer's stroke. Acquiring a natural language would not result in greater expressive power; it would, however, provide a more suitable representational system, given cognitive limitations in memory, architecture, and perhaps others. That is, acquiring a language would constitute a transparent cognitive enrichment; a better mental organisation. It is at this point that psychology can actually start its investigations; i.e., how many connectives does the LoT actually have and how do they interface with those of natural language? Various sources of data can be brought to inform such an enterprise, and it is to be expected that language acquisition and processing studies would play a central role. Nevertheless, it is plausible to suggest that it is only after we have gained a better understanding of issues such as these that we can even approach the related matter of figuring out what connectives are actually employed during reasoning, perhaps going some way into explaining some of the results reported in the rationality literature.

The three combinatorial operations I have listed are closely related to recursion, given their role in computational processes, architectural complexity, structured representations and the like. We can indeed expect some advances in the study of the architecture underlying general cognition by delving deeper into the nature of these three elements, but I do not think that work on non-linguistic self-embedded structures is likely to yield interesting results. It is certainly to be expected that many scholars will continue to propose that this or that domain exhibits self-embedding, and perhaps some of these academics will even offer experimental evidence showing how humans interpret this specific type of hierarchical representation in non-linguistic domains. However, it is very unlikely that non-linguistic self-embedded structures will ever be shown to be isomorphic to the corresponding linguistic representations, and as a result it seems to me that scholars will be forced to conclude, unenlighteningly, that the mind makes use of certain hierarchical structures that are not very closely related.

The role of recursion in the language faculty, on the other hand (i.e., *merge* and the derivations it effects), appears to me to be well-established and settled, and I cannot see any advances in that respect, apart from the perennial confusion and non sequiturs the literature will continue to provide. Indeed, we do have a good understanding of both the expressive power and full generativity of the language faculty —two properties that have emerged from a specific conglomeration of systems— even though it is still an open question whether any of the internal components of the faculty are specific to language or are, instead, domain general. If some of these components were indeed specific to language, it would obviously be of much interest to work out how analogous the operations of the language faculty and those of the LoT (including the inferential system) turn out to be.

As for real-time processes, it is uncertain whether language comprehension and general cognition make use of recursive sub-routines. Obviously, the possibility cannot be discounted, but at present the cognitive science literature offers scant evidence for it. Chapter 4 probed the possibility that the syntactic parser may involve self-calls by employing a task that engaged working memory, but the results suggested that the process was instead iterative (at least at the early stage of the parsing process we investigated). Nevertheless, the results of our experiments pointed to a clear position effect and a possible "verb search" phenomenon, and both data call for more research.

The position effect seems to have been related to the level of uncertainty subjects experience during the processing of a sentence. Thus, this phenomenon could profitably be investigated by carrying out an experiment with event-related potentials, as there is a well-known component (the P3/P300) that is usually activated in relation to how unexpected an event is. That is, the amplitude of this component is negatively correlated with the probability that a given categorical item will be present: the higher the probability, the lower the amplitude of the P300. Considering that there is less uncertainty towards the end of a sentence, it is to be expected that the amplitude of a P300 associated with the perception of a click placed at the end will be lower than the amplitude of a P300 elicited by a click located at the beginning.

As for the "verb search" datum, it would be of interest to expand the range and complexity of the materials we employed here in order to establish whether this verb effect really is all that substantial. Crucially, Spanish is rather flexible in word order (much like other romance languages) and non-canonical sentences (that is, orders other than SVO) are ideal for the purposes at hand. Consider the following sentences:

(5.1) Juan recibió la carta.
Juan received the letter.

(5.2) Juan la carta recibió.
Juan the letter received.

(5.3) Recibió Juan la carta.
received Juan the letter.

(5.4) Recibió la carta Juan.
Received the letter Juan.

(5.5) La carta recibió Juan.
The letter received Juan.

(5.6) La carta Juan recibió.
The letter Juan received.

These sentences encompass the six possible word orders, and they are all grammatical in Spanish (with varying degrees of naturalness, of course). Note, however, that apart from examples (5.3) and (5.6), these sentences respect the macro SHC structure (that is, the verb and its object(s) go together, while the subject is in a different hierarchical position). It is precisely these four macro SHC structures that ought to be employed in order to figure out what effect the verb is in fact having, as both the position of the verb and the placement of the tones can successfully be manipulated for this very purpose (employing a structure that disrupts the SHC scheme would unnecessarily complicate matters).

These two factors —the position effect and the verb-search phenomenon— are not directly related to recursion, though. This is rather obvious in the case of the position effect, and whilst it is true that the verb effect has been interpreted as evidence that the parser is sensitive to the macro SHC structure, no evidence was found that the processor builds internal SHCs recursively. If so, a change in word order will be orthogonal to whether there are any recursive sub-routines in parsing; rather, word order change is likely to disrupt the application of the SHC template, a different matter altogether. Still, these are issues that directly derive from the work presented herein, and it seems to me that it is imperative that we gain an understanding of the basic properties at play in the first-pass of the parser before venturing to study how they apply in much more complicated sentences, as the processing of the latter involves factors that do not appear to be operative in the analysis of simple declaratives (or at least not to the same extent; some of these factors plausibly include prosody, semantic information, ecc.).

Appendices

A: Materials Experiment 1

Practice items

1. El comisario del distrito ha entrevistado al imputado.
2. La limpieza del edificio se ha cedido a otra empresa.
3. La mayoría de diputados ha votado desde sus hogares.
4. El mensaje de nuestro aliado ha llegado demasiado tarde.
5. El alumno de filosofía se comportó adecuadamente.
6. El asesor novel del juzgado ha mediado de forma ejemplar.
7. Los empleados se rebelaron en contra de los nuevos convenios.
8. El manuscrito se publicará como libro de texto infantil.
9. El cartero ha estimado su llegada a las dos de la tarde.
10. La alergia se propagó por el continente sudamericano.
11. El conductor ha admitido la infracción cometida anoche.
12. Los sillones se subastaron en el evento de esta mañana.

Experimental items

1. El candidato del partido se preparó el próximo discurso.
2. El candidato ha preparado un discurso sobre la sanidad.
3. El caballero de la mesa seis ha pedido un sándwich vegetal.
4. El caballero ha pedido un sándwich grande con patatas fritas.
5. El anciano del ambulatorio se cayó sobre el joven doctor.
6. El anciano se ha caído sobre el doctor del centro de salud.

7. La historia de la democracia ha dado ya muchísimas vueltas.
8. La historia sigue dando muchísimas vueltas sobre este tema.
9. La rebaja de la entrada se aplicará en la misma puerta.
10. La rebaja se aplicará en la puerta central del auditorio.
11. La decisión del actual gobierno se aprobará sin oposición.
12. La decisión se aprobará sin una oposición destacable.
13. El estadio de baloncesto se ha llenado de aficionados.
14. El estadio se ha llenado de aficionados de la cantera.
15. El ministro de educación se ha presentado con su dimisión.
16. El ministro se ha presentado con su dimisión bajo el brazo.
17. El proyecto del banco central se abandonó por varias razones.
18. El proyecto se abandonó por razones bastante misteriosas.
19. Los testigos de la defensa han ocultado demasiadas pruebas.
20. Los testigos han ocultado demasiadas pruebas de importancia.
21. El portero del primer equipo está tratando su renovación.
22. El portero está tratando la renovación del nuevo contrato.
23. La ponencia sobre el paro ha molestado a sus señorías.
24. La ponencia ha molestado a sus señorías más liberales.
25. El experto en inmigración se ha burlado de los refugiados.
26. El experto se ha burlado de los refugiados recién llegados.
27. Los cohetes de los festejos se lanzaron desde el municipio.
28. Los cohetes fueron lanzados desde la terraza del municipio.
29. El abogado de la defensa no se fijó en todos sus gestos.
30. El abogado no se fijó en todos los gestos del acusado.
31. El acuerdo de los sindicatos se firmó con un claro consenso.
32. El acuerdo se ha firmado con un consenso algo limitado.
33. El estudio del mercado se concentraba en la agricultura.

34. El estudio se concentraba en el sector de la agricultura.
35. La apertura del hospital se demoró por diversos motivos.
36. La apertura se demoró por motivos aún desconocidos.
37. La autora de los poemas se dirigió a todos sus lectores.
38. La autora se dirigirá a los lectores de sus biografías.
39. El salario de los funcionarios ha crecido ya más de la media.
40. El salario ha crecido ya más de la media interprofesional.
41. El concierto de la radio local se desarrolló sin ningún orden.
42. El concierto se desarrolló sin ningún orden predeterminado.
43. Las señoras del quinto piso han discutido durante la tarde.
44. Las señoras han discutido todas las tardes de esta semana.
45. El médico de la consulta ha explicado su punto de vista.
46. El médico ha explicado el punto de vista de su superior.
47. El tratado entre los partidos se estableció en los mítines.
48. El tratado no se estableció en los mítines electorales.
49. El título de la Recopa se ha ganado por error del linier.
50. El título se ha ganado gracias al linier de la federación.
51. El conserje de la estación ha resuelto todos nuestros problemas.
52. El conserje ha resuelto ya nuestros problemas con el equipaje.
53. La reunión con el gobernador se realizó hoy en su despacho.
54. La reunión se realizó en un despacho de este edificio.
55. El acusado por estafa ha recurrido la dura sentencia.
56. El acusado ha recurrido la sentencia del juez ordinario.
57. La víctima del incendio se ha expresado de forma confusa.
58. La víctima se expresó de forma confusa por el fuerte golpe.
59. El capitán del navío se ha retirado a su camerote.
60. El capitán se retiró al camarote más grande de la nave.

61. El vecino de la cuarta planta ha firmado todos los papeles.
62. El vecino ha firmado ya los papeles de mayor importancia.
63. El palacio de la Moncloa se construyó hace cuarenta años.
64. El palacio se construyó hace bastante más de cuarenta años.
65. La asociación de padres no estaba luchando por la reforma.
66. La asociación está luchando por la reforma del estatuto.
67. El ataque a la policía ha sorprendido a la población.
68. El ataque ha sorprendido a la mayoría de la población.
69. Las medidas del gerente han generado muchas quejas extrañas.
70. Las medidas han generado muchas quejas por parte de los clientes.
71. Las postales del escritor se valoraron en millones de euros.
72. Las postales se valoraron por una suma de miles de euros.
73. El productor de la película se llevó demasiados elogios.
74. El productor se ha llevado muchos elogios de sus compañeros.
75. El territorio de la capital se llenaba de manifestantes.
76. El territorio se llenó de manifestantes a favor del pacto.
77. El contrato de la hipoteca se guardó en la estantería.
78. El contrato se guardaba en la estantería de mi despacho.
79. El impuesto de circulación se abonaba por correo postal.
80. El impuesto se abonará por correo postal certificado.
81. El alcalde de la provincia se esperaba ya su destitución.
82. El alcalde se esperaba una destitución improcedente.
83. El terremoto de California se percibió desde San Francisco.
84. El terremoto se percibió desde las afueras de San Francisco.
85. Los jugadores de bádminton se presentaron con pocas raquetas.
86. Los jugadores se presentaron con raquetas del año pasado.
87. El concejal de la comarca ha exigido la indemnización.

88. El concejal ha exigido la indemnización por el despido.
89. El árbitro de la semifinal se presentó sin sus auxiliares.
90. El árbitro se presentó sin sus auxiliares de toda la vida.
91. El agente de seguridad ha informado de la incidencia.
92. El agente ha informado de la gravedad de la incidencia.
93. El primer juez del supremo ha desestimado todas las mociones.
94. El supremo ha desestimado las mociones de los diputados.
95. El aspecto del actor belga ha mejorado tras la operación.
96. Su aspecto ha mejorado tras la operación del mes pasado.
97. El mensajero de la agencia se olvidó de nuestro paquete.
98. El mensajero se olvidó el paquete de nuestros empleados.
99. Los asistentes al congreso se marcharon algo decepcionados.
100. Los asistentes se marcharon decepcionados con la conferencia.
101. Las lámparas del recibidor no se rompieron por culpa del viento.
102. Las lámparas no se rompieron por el viento de la pasada noche.
103. Las chaquetas para la fiesta se encargaron al mejor modisto.
104. Las chaquetas se encargaron al modisto mejor recomendado.
105. El carnicero del barrio se ha despedido de todos sus clientes.
106. El carnicero no se despidió de sus clientes de toda la vida.
107. El profesor de literatura ha publicado todos sus cuentos.
108. El profesor ha publicado solo sus cuentos de mayor interés.
109. Los bomberos más antiguos han apagado ya todos los incendios.
110. Los bomberos han apagado ya los incendios más problemáticos.
111. El general de la armada ha declarado ante el tribunal.
112. El general ha declarado ante el tribunal de apelación.
113. Los estudiantes de la facultad se fueron a la manifestación.
114. Los estudiantes se han ido de camino a la manifestación.

115. El conflicto en Oriente Medio se extiende al resto de Asia.

116. El conflicto se extenderá al resto del Asia más meridional.

117. Los turistas extranjeros se alojaron en un lujoso hotel.

118. Los turistas se alojaron en un hotel de la costa dorada.

119. Los libreros de la Rambla se han quejado de las escasas ventas.

120. Los libreros se han quejado de las pocas ventas de navidades.

B: Materials Experiment 2 and 3

New practice items

1. El autocar de las seis ha llegado tarde y hemos tenido que coger un taxi.
2. Luis, que está haciendo un doctorado, se graduará a finales de septiembre.
3. Los aficionados del Barça están seguros de que su equipo ganará la liga.
4. Nevará durante todo el día según las últimas previsiones.
5. Las razones del fiscal serán expuestas mañana al mediodía.
6. Por estos lugares suceden cosas muy extrañas a veces.

Fillers

1. El café ha llegado frío y los clientes se han quejado al camarero.
2. El camión se salió de la carretera pero no atropelló a nadie.
3. El domingo se fue la luz mientras estábamos cenando.
4. Las autoridades suspendieron el desfile y las tropas volvieron al cuartel.
5. La revista se publicó hoy pero nuestro artículo no ha sido incluido.
6. Los trabajadores se han declarado en huelga porque no han recibido sus nóminas.
7. La secretaria recogió todas sus cosas y se trasladó al nuevo edifico.
8. El periodista entrevistó al artista ayer pero no mostró mucho interés.
9. O vamos nosotros a su casa, o cenamos en un restaurante todos juntos.
10. Ni tenemos las llaves de casa ni podemos llamar a ningún familiar.
11. El caballo se salió del circuito mientras el jinete perdía el equilibrio.

12. En cuanto la lluvia cae con fuerza, los peatones desaparecen de las calles.
13. La casa no solo se estaba quemando, sino que también se estaba derrumbando.
14. Los congresistas tienen la certeza de que el presidente dimitirá.
15. La Guardia Civil está cerca de descubrir la identidad del culpable.
16. El jefe de recursos humanos preguntó cómo se llamaba el nuevo empleado.
17. La Iglesia dio dinero a quién más lo necesitaba.
18. Nuestros amigos llegaron al cine cuando la película ya había comenzado.
19. La vecina se mudó de casa cuando falleció su marido.
20. La ciudad que visitaron los turistas se llama Barcelona.
21. El Rey llegó justo cuando los invitados se estaban marchando.
22. El experto en economía da la impresión de que lo sabe todo.
23. El invitado del gobierno parece contento de haber venido a nuestro país.
24. María, que es muy atractiva, se presentó al baile sin su pareja.
25. Por estas vías pasan trenes constantemente.
26. Me gustan mucho los helados artesanales italianos.
27. En esta región existen problemas de difícil solución.
28. Resulta que el presunto violador se ha escapado.
29. Lo que ha ocurrido en estas tierras no está nada claro.
30. No hay noticias fiables sobre el estado de la cantante.
31. Por lo visto, Juan llegará sobre las siete de la tarde.
32. Ahora mismo deben de ser las diez de la noche.
33. Hay que tener paciencia con los alumnos del primer curso.
34. Aquí se juega al tenis todos los días del verano.
35. En la selva amazónica desaparecen periodistas todos los años.
36. Faltan todavía muchos ejercicios por completar.
37. Entra mucho frío por las ventanas del primer piso.

38. Llega gente de todas partes para las fiestas del pueblo.
39. Por esta zona crecen muchos árboles silvestres.
40. En ese bar se grita mucho durante los partidos de fútbol.
41. Está al caer la convocatoria del año dos mil once.
42. Los impuestos subieron por quinto año consecutivo.
43. Parece que vamos a quedarnos sin vacaciones este año.
44. Está prohibido fumar en el interior de la universidad.
45. Los documentos han sido transferidos a nuestro despacho.
46. Las palabras del gobernador fueron aplaudidas por todos los presentes.
47. Comenzaron las festividades antes de lo previsto.
48. En esta tienda se venden muy buenos churros.
49. El mensaje de nuestro aliado ha llegado demasiado tarde.
50. La limpieza del edificio se ha cedido a otra empresa.
51. El asesor del nuevo juzgado ha mediado de forma ejemplar.
52. El accidente aéreo de hoy se ha cobrado muchas vidas.
53. La escuela de diseño ha reformado varias instalaciones.
54. La noticia de su muerte ha conmocionado a la sociedad.
55. El cartero ha estimado su llegada a las dos de la tarde.
56. Los empleados se rebelaron en contra de los nuevos convenios.
57. El escrito se ha publicado como libro de texto infantil.
58. La nueva ley ha limitado los derechos de los trabajadores.
59. La fábrica ha paralizado la producción de automóviles.
60. El coleccionista ha comprado el cuadro más caro de la tienda.

Comprehension task

Practice session

1. ¿Se comportó bien el alumno? (Yes)
2. ¿Llegará el autocar por la mañana? (No)

Experimental sentences

Experimental versions 1-2-3

1. ¿Pidió el caballero un sándwich de jamón? (N)
2. ¿Se cayó el doctor sobre el anciano? (N)
3. ¿Era el ministro de interior quien iba a dimitir? (N)
4. ¿Se está negociando la renovación del portero? (Y)
5. ¿Se retrasó la inauguración del hospital? (Y)
6. ¿Recurrió la decisión el acusado? (Y)
7. ¿Se construyó el palacio hace veinte años? (N)
8. ¿Ha habido quejas por parte de los clientes? (Y)
9. ¿Fue el director el que se llevó muchos elogios? (N)
10. ¿Rechazó el supremo las mociones presentadas? (Y)
11. ¿Pudieron los bomberos con los peores incendios? (Y)
12. ¿Se alojaron los turistas en un hotel de la costa del sol? (N)

Experimental versions 4-5-6

1. ¿Pidió el caballero un sándwich pequeño? (N)
2. ¿Se cayó el doctor sobre el anciano? (N)
3. ¿Era el presidente quien iba a dimitir? (N)
4. ¿Se está negociando la renovación del portero? (Y)
5. ¿Se está retrasando la inauguración? (Y)
6. ¿Recurrió la decisión del juez el acusado? (Y)
7. ¿Se construyó el palacio hace sesenta años? (N)
8. ¿Ha habido quejas a raíz de las medidas del gerente? (Y)
9. ¿Fue el director el que se llevó muchos elogios? (N)
10. ¿Rechazó el juez del supremo las mociones presentadas? (Y)
11. ¿Pudieron los bomberos más veteranos con el incendio? (Y)
12. ¿Se alojaron los turistas en un motel? (N)

Fillers

1. ¿Recibieron los clientes el café caliente? (N)
2. ¿Fue el artista entrevistado hoy? (N)
3. ¿Estaba ardiendo la casa? (Y)
4. ¿Donó la Iglesia dinero a los más necesitados? (Y)
5. ¿Fueron María y su pareja a un baile? (N)
6. ¿Había empezado la película cuando llegaron nuestros amigos? (Y)
7. ¿Han arrestado al presunto violador? (N)
8. ¿Se puede ver deporte por la tele en ese bar? (Y)
9. ¿Se puede fumar en las aulas de la universidad? (N)
10. ¿Se venden churros en la tienda? (Y)
11. ¿Pasan los trenes muy de vez en cuando? (N)
12. ¿Se desconoce el estado de la cantante? (Y)

References

Abe, K. & Watanabe, D. (2011). Songbirds possess the spontaneous ability to discriminate syntactic rules. *Nature*, *14*(8), 1067-74.

Abelson, H., Sussman, G. J. & Sussman, J. (1996). *Structure and interpretation of computer programs*. Cambridge, MA.: The MIT Press.

Abrams, K. & Bever, T. G. (1969). Syntactic structure modifies attention during speech perception and recognition. *The Quarterly Journal of Experimental Psychology*, *21*(3), 280-290.

Ackema, P. & Neeleman, A. (2011). *Subset controllers in agreement relations.* Manuscript.

Adger, D. & Svenonius, P. (2011). Features in minimalist syntax. In C. Boeckx (Ed.), *The Oxford handbook of linguistic minimalism* (p. 27-51). Oxford University Press.

Aho, A. V., Hopcroft, J. E. & Ullman, J. D. (1974). *The design and analysis of computer algorithms*. London, England: Addison-Wesley Publishing Company.

Almela, R., Cantos, P., Sánchez, A., Sarmiento, R. & Almela, R. (2005). *Frecuencias del Español. Diccionario y estudios léxicos y morfológicos*. Madrid, Spain: Editorial Universitas.

Arsenijević, B. & Hinzen, W. (2010). Recursion as a human universal and as a primitive. *Biolinguistics*, *4*(2-3), 165-173.

Bahlmann, J., Schubotz, R. I. & Friederici, A. D. (2008). Hierarchical artificial grammar processing engages Broca's area. *NeuroImage*, *42*, 525-534.

Barceló-Coblijn, L. (forthcoming). Evolutionary scenarios for the emergence of recursion. *Theoria et Historia Scientiarum*, 1-26.

Bar-Hillel, Y. (1953). On recursive definitions in empirical science. In *Proceedings of the 11th International Congress of Philosophy in Brussels* (Vol. 5, p. 160-5). Amsterdam, The Netherlands: North-Holland Publishing Co.

Benacerraf, P. (1965). What numbers could not be. *The Philosophical Review*, *74*(1), 47-73.

Berwick, R. C. & Chomsky, N. (2011). The biolinguistic program: the current state and its development. In A. M. Di Sciullo & C. Boeckx (Eds.), *The biolinguistic enterprise* (p. 19-41). Oxford University Press.

Bever, T. G. (1970). The cognitive basis for linguistic structures. In J. R. Hayes (Ed.), *Cognition and the development of language* (p. 279-362). Wiley-Blackwell.

Bever, T. G. (1973). Serial position and response biases do not account for the effect of syntactic structure on the location of brief noises during sentences. *Journal of Psycholinguistic Research*, *2*, 287-288.

Bever, T. G. & Hurtig, R. R. (1975). Detection of a nonlinguisic stimulus is poorest at the end of a clause. *Journal of Psycholinguistic Research*, *4*(1), 1-7.

Bever, T. G., Lackner, J. R. & Kirk, R. (1969). The underlying structures of sentences are the primary units of immediate speech processing. *Perception and Psychophysics*, *5*(4), 225-234.

Bever, T. G. & Poeppel, D. (2010). Analysis by synthesis: a (re-)emerging program of research for language and vision. *Biolinguistics*, *4*(2-3), 174-200.

Bever, T. G., Sanz, M. & Townsend, D. J. (1998). The Emperor's Psycholinguistics. *Journal of Psycholinguistic Research*, *27*(2), 261-284.

Bickerton, D. (1981). *Roots of language*. USA: Karoma Publishers, Inc.

Bickerton, D. (2009). Recursion: core of complexity or artifact of analysis? In T. Givón & M. Shibatani (Eds.), *Syntactic complexity: diachrony, acquisition, neuro-cognition, evolution* (p. 531-44). John Benjamins.

Blass, A. & Gurevich, Y. (2003). Algorithms: a quest for absolute definitions. *Bulletin of the European Association of Theoretical Computer Science*, *81*, 195-225.

Bloom, P. (1994). Generativity within language and other cognitive domains. *Cognition*, *51*, 177-189.

Boeckx, C. (2006). *Linguistic minimalism*. Oxford, England: Oxford University Press.

Boeckx, C. (2009a). *Language in cognition*. Oxford, England: Wiley-Blackwell.

Boeckx, C. (2009b). The nature of merge: consequences for language, mind and biology. In M. Piatelli-Palmarini, J. Uriagereka & P. Salaburu (Eds.), *Of minds and language: a dialogue with Noam Chomsky in the Basque Country* (p. 44-57). Oxford University Press.

Boeckx, C. (2009c). On the locus of asymmetry in UG. *Catalan Journal of Linguistics*, *8*, 41-53.

Boeckx, C. & Uriagereka, J. (2007). Minimalism. In G. Ramchand & C. Reiss (Eds.), *The Oxford Handbook of Linguistic Interfaces* (p. 541-574). Cambridge, England.: Oxford University Press.

Bond, Z. S. (1972). Phonological units in sentence perception. *Phonetica*, *25*(3), 129-39.

Boolos, G. (1971). The iterative conception of set. *The Journal of Philosophy*, *68*(8), 215-231.

Brattico, P. (2010). Recursion hypothesis considered as a research program for cognitive science. *Minds and Machines*, *20*, 213-41.

Brattico, P. & Liikkanen, L. (2009). Rethinking the Cartesian theory of linguistic productivity. *Philosophical Psychology*, *22*(3), 251-79.

Bresnan, J. (2001). *Lexical-Functional syntax*. Oxford, England: Blackwell Publishers.

Brody, M. (1994). Phrase structure and dependence. *UCL Working Papers in Linguistics*, *6*, 1-33.

Brody, M. (2002). On the status of representations and derivations. In S. D. Epstein & T. D. Seely (Eds.), *Derivation and explanation in the minimalist program* (p. 19-41). Blackwell Publishing.

Bromberger, S. & Halle, M. (1991). Why phonology is different. In A. Kasher (Ed.), *The Chomkyan turn* (p. 56-77). Basil Blackwell Ltd.

Brook, A. (2006). *The prehistory of cognitive science*. London, England: Palgrave Macmillan.

Brookshear, G. (2000). *Computer science*. Reading, MA.: Addison-Wesley.

Cardinaletti, A. & Starke, M. (1999). The typology of structural deficiency. In H. van Riemsdijk (Ed.), *Clitics in the languages of Europe* (p. 145-233). Mouton de Gruyter.

Carroll, J. M. & Tanenhaus, M. K. (1978). Functional clauses and sentences segmentation. *Journal of Speech and Hearing Research*, *21*, 793-808.

Carruthers, P. (2002). The cognitive functions of language. *Behavioral and Brain Sciences*, *25*, 657-726.

Chalmers, D. (1990). Why Fodor and Pylyshyn were wrong: the simple refutation. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society* (p. 340-346).

Chapin, P. G., Smith, T. S. & Abrahamson, A. A. (1972). Two factors in perceptual segmentation of speech. *Journal of Verbal Learning and Verbal Behavior*, *11*, 164-173.

Chesi, C. (2005). Phases and complexity in phrase structure building. In *Computational linguistics in the Netherlands 2004- Selected papers from the Fifteenth CLIN meeting* (p. 59-73).

Chomsky, N. (1955). Logical syntax and semantics: their linguistic relevance. *Language*, *31*(1), 36-45.

Chomsky, N. (1956). Three models for the description of language. In *IRE Transactions of Information Theory IT-2* (p. 113-124).

Chomsky, N. (1957). *Syntactic structures*. The Hague, The Netherlands: Mouton Publishers.

Chomsky, N. (1963). Formal properties of grammars. In R. D. Luce, R. R. Bush & E. Galanter (Eds.), *Handbook of mathematical psychology* (p. 323-418). John Wiley and sons, Inc.

Chomsky, N. (1965). *Aspects of the theory of syntax*. Cambridge, MA.: The MIT Press.

Chomsky, N. (1967). Recent contributions to the theory of innate ideas. *Synthese*, *17*, 2-11.

Chomsky, N. (1971). On interpreting the world. *Cambridge Review*, *92*(2200), 77-93.

Chomsky, N. (1975a). *The logical structure of linguistic theory*. New York, New York: Plenum Press.

Chomsky, N. (1975b). *Reflections on language*. London, England: Maurice Temple Smith.

Chomsky, N. (1980). *Rules and representations*. New York, New York: Columbia University Press.

Chomsky, N. (1981). *Lectures on government and binding*. The Hague: The Netherlands: De Gruyter Mouton.

Chomsky, N. (1986). *Knowledge of language*. New York, New York: Praeger Press.

Chomsky, N. (1995a). Bare phrase structure. In G. Webelhuth (Ed.), *Government and binding theory and the Minimalist Program* (p. 381-439). Blackwell Publishers.

Chomsky, N. (1995b). *The minimalist program*. Cambridge, MA.: The MIT Press.

Chomsky, N. (2000a). Linguistics and brain science. In A. Marantz, Y. Miyashita & W. O'Neil (Eds.), *Image, language, brain* (p. 13-28). The MIT Press.

Chomsky, N. (2000b). Minimalist inquiries: the framework. In R. Martin, D. Michaels & J. Uriagereka (Eds.), *Step by step* (p. 89-156). The MIT Press.

Chomsky, N. (2001a). Derivation by phase. In M. Kenstowicz (Ed.), *Ken Hale: a life in language* (p. 1-51). The MIT Press.

Chomsky, N. (2001b). *New horizons in the study of language and mind*. Cambridge, England: Cambridge University Press.

Chomsky, N. (2002). *On nature and language*. Cambridge, England: Cambridge University Press.

Chomsky, N. (2004). Beyond explanatory adequacy. In A. Belleti (Ed.), *Structures and beyond* (p. 104-131). Oxford University Press.

Chomsky, N. (2006). *Language and mind*. Cambridge, England: Cambridge University Press.

Chomsky, N. (2007a). Approaching UG from below. In H.-M. Gartner & U. Sauerland (Eds.), *Interface+recursion=language?* (p. 1-29). The Netherlands: Mouton de Gruyter.

Chomsky, N. (2007b). Biolinguistic explorations: design, development and evolution. *International Journal of Philosophical Studies*, *15*(1), 1-21.

Chomsky, N. (2008). On phases. In R. Freidin, C. P. Otero & M. L. Zubizarreta (Eds.), *Foundational issues in linguistic theory* (p. 133-166). The MIT Press.

Chomsky, N. & Miller, G. A. (1963). Introduction to the formal analysis of natural languages. In R. D. Luce, R. R. Bush & E. Galanter (Eds.), *Handbook of mathematical psychology, vol. 2* (p. 269-322). John Wiley and Sons, Inc.

Christiansen, M. H. & Chater, N. (1999). Towards a connectionist model of recursion in human linguistic performance. *Cognitive Science*, *23*(2), 157-205.

Christiansen, M. H. & Chater, N. (2003). Constituency and recursion in language. In M. A. Arbib (Ed.), *The Handbook of brain theory and neural networks* (p. 267-71). The MIT Press.

Christiansen, M. H. & MacDonald, M. C. (2009). A usage-based approach to recursion in sentence processing. *Language learning*, *59*(1), 126-61.

Church, A. (1932). A set of postulates for the foundation of logic. *Annals of Mathematics, Series 2*, *33*, 346-66.

Church, A. (1936). An unsolvable problem of elementary number theory. In M. Davis (Ed.), *The undecidable* (p. 88-107). Dover Publications, Inc.

Churchland, P. (1990). Cognitive activity in artificial neural networks. In R. Cummins & D. Cummins (Eds.), *Minds, brains and computers: an anthology* (p. 198-216). Oxford: Blackwell Publishers.

Citko, B. (2005). On the nature of merge: External merge, internal merge, and parallel merge. *Linguistic Inquiry*, *36*(4), 475-496.

Cohen, L. & Mehler, J. (1996). Click monitoring revisited: an on-line study of sentence comprehension. *Memory and Cognition*, *24*(1), 94-102.

Collins, C. & Stabler, E. (2011). *A formalization of minimalist syntax.* Manuscript.

Collins, J. (2004). Faculty disputes. *Mind and Language*, *19*(5), 503-533.

Collins, J. (2007). Linguistic competence without knowledge of language. *Philosophy Compass*, *2*(6), 880-895.

Collins, J. (2008a). *Chomsky: A guide for the perplexed.* London, UK: Continuum International Publishing Group Ltd.

Collins, J. (2008b). Knowledge of language redux. *Croatian Journal of Philosophy*, *22*, 3-43.

Corballis, M. (2003). Recursion as the key to the human mind. In K. Sterelny & J. Fitness (Eds.), *From mating to mentality: Evaluating evolutionary psychology* (p. 155-171). Psychology Press.

Corballis, M. (2007a). Recursion, language and starlings. *Cognitive Science*, *31*(4), 697-704.

Corballis, M. (2007b). The uniqueness of human recursive thinking. *American Scientist*, *95*, 240-248.

Corballis, M. (2011). *The recursive mind.* Princeton, New Jersey: Princeton University Press.

Crocker, M. W. (1996). *Mechanisms for sentence processing* (Tech. Rep.). The University of Edinburgh. (Research paper EUCCS/RP-70)

Curtiss, S. (1977). *Genie: a linguistic study of a modern-day wild child.* New York, New York: Academic Press Inc.

Cutland, N. (1980). *Computability: an introduction to recursion function theory.* Cambridge, England: Cambridge University Press.

Cutler, A. & Norris, D. (1979). Monitoring sentence comprehension. In W. E. Cooper & E. C. T. Walker (Eds.), *Sentence processing: psycholinguistic studies presented to Merrill Garrett* (p. 113-34). Lawrence Erlbaum.

de Vries, M. H., Christiansen, M. H. & Petersson, K. M. (2011). Learning recursion: multiple nested and crossed dependencies. *Biolinguistics*, *5*(1-2), 10-35.

de Vries, M. H., Monaghan, P., Knecht, S. & Zwitserlood, P. (2008). Syntactic structure and artificial grammar learning: the learnability of embedded hierarchical structures. *Cognition*, *107*, 763-774.

Dean, W. (2007). *What algorithms could not be*. Unpublished doctoral dissertation, Rutgers University.

Di Sciullo, A. M. & Isac, D. (2008). The asymmetry of merge. *Biolinguistics*, *2*(4), 260-290.

Dupuy, J.-P. (2009). *On the origins of cognitive science*. Cambridge, Massachusetts: The MIT Press.

Edelman, S. (2008). *Computing the mind*. Oxford, England: Oxford University Press.

Eliot, J., Lovell, K., Dayton, C. M. & McGrady, B. F. (1979). A further investigation of children's understanding of recursive thinking. *Journal of Experimental Child Psychology*, *28*, 149-157.

Epstein, R. & Carnielli, W. (2008). *Computability: Computable functions, logic, and the foundations of mathematics*. Socorro, New Mexico: Advanced Reasoning Forum.

Epstein, S. D. & Hornstein, R. (2000). *Working minimalism*. Cambridge, MA.: The MIT Press.

Epstein, S. D. & Seely, T. D. (2002). *Derivation and explanation in the minimalist program*. Oxford, England: Blackwell Publishing.

Evans, N. & Levinson, S. C. (2009). The myth of language universals: language diversity and its importance for cognitive science. *Behavioral and Brain Sciences*, *32*, 429-492.

Everett, D. L. (2005). Cultural constraints on grammar and cognition in Pirahã. *Current Anthropology*, *46*(4), 621-646.

Everett, D. L. (2009). Pirahã culture and grammar: a reply to some criticisms. *Language*, *85*(2), 405-42.

Everett, D. L. (2010). *You drink. you drive. you go to jail. where's recursion?* lingBuzz/001141.

Feferman, S. (2009). Gödel, Nagel, minds and machines. *The Journal of Philosophy*, *106*(4), 201-19.

Fernández, E. M. & Smith Cairns, H. (2011). *Fundamentals of psycholinguistics*. Oxford, England: Wiley-Blackwell.

Ferreira, F. & Patson, N. D. (2007). The 'good enough' approach to language comprehension. *Language and linguistics compass*, *1*(1-2), 71-83.

Fitch, W. T. (2010). Three meanings of recursion: key distinctions for biolinguistics. In R. Larson, V. Déprez & H. Yamakido (Eds.), *The evolution of human language* (p. 73-90). Cambridge University Press.

Fitch, W. T. & Hauser, M. D. (2004). Computational constraints on syntactic processing in nonhuman primates. *Science*, *303*, 377-380.

Fitch, W. T., Hauser, M. D. & Chomsky, N. (2005). The evolution of the language faculty: clarifications and implications. *Cognition*, *97*, 179-210.

Flores d'Arcais, G. B. (1978). The perception of complex sentences. In W. J. M. Levelt & G. B. Flores d'Arcais (Eds.), *Studies in the perception of language* (p. 155-185). John Wiley and Sons.

Fodor, J. A. (1975). *The language of thought*. Cambridge, MA.: Harvard University Press.

Fodor, J. A. (1979). On the impossibility of acquiring 'more powerful' structures. In M. Piatelli-Palmarini (Ed.), *Language and learning: The debate between Jean Piaget and Noam Chomsky* (p. 142-161). Routledge.

Fodor, J. A. (1983). *The modularity of mind*. Cambridge, MA.: Bradford Books/The MIT Press.

Fodor, J. A. (1998). *Concepts: where cognitive science went wrong*. Oxford: Oxford University Press.

Fodor, J. A. (2001). Doing without what's within. *Mind*, *110*, 99-148.

Fodor, J. A. (2008). *LOT 2: The language of thought revisited*. Oxford: Oxford University Press.

Fodor, J. A. & Bever, T. G. (1965). The psychological reality of linguistic segments. *Journal of Verbal Learning and Verbal Behavior*, *4*, 414-420.

Fodor, J. A., Bever, T. G. & Garrett, M. F. (1974). *The psychology of language*. London, England: McGraw-Hill.

Fodor, J. A. & McLaughlin, B. (1990). Connectionism and the problem of systematicity: why Smolensky's solution doesn't work. *Cognition*, *35*, 183-204.

Fodor, J. A. & Pylyshyn, Z. (1988). Connectionism and cognitive architecture: a critical analysis. *Cognition*, *28*, 3-71.

Folia, V., Forkstam, C., Ingvar, M., Hagoort, P. & Petersson, K. M. (2011). Implicit artificial syntax processing: genes, preference, and bounded recursion. *Biolinguistics*, *5*(1-2), 105-132.

Forster, K. I. & Forster, J. C. (2003). DMDX: A windows display program with millisecond accuracy. *Behavior Research Methods, Instruments, and Computers*, *35*, 116-124.

Forster, T. (2008). The iterative conception of set. *The Review of Symbolic Logic*, *1*(1), 97-110.

Frampton, J. & Gutmann, S. (1999). Cyclic computation, a computationally efficient minimalist syntax. *Syntax*, *2*(1), 1-27.

Frank, R. (2004). Restricting grammatical complexity. *Cognitive Science*, *28*, 669-697.

Frazier, L. (1988). Grammar and language processing. In F. J. Newmeyer (Ed.), *Linguistics: The Cambridge Survey vol. II* (p. 15-34). Cambridge University Press.

Frazier, L. & Clifton Jr., C. (1996). *Construal*. Cambridge, MA.: The MIT Press.

Frazier, L. & Fodor, J. D. (1978). The sausage machine: A new two-stage parsing model. *Cognition*, *6*, 291-325.

Friederici, A. D., Bahlmann, J., Friedrich, R. & Makuuchi, M. (2011). The neural basis of recursion and complex syntactic hierarchy. *Biolinguistics*, *5*(1-2), 87-104.

Friederici, A. D., Bahlmann, J., Heim, S., Schubotz, R. I. & Anwander, A. (2006). The brain differentiates human and non-human grammars: functional localization and structural connectivity. *Proceedings of the National Academy of Sciences of the USA*, *103*(7), 2458-63.

Friedrich, R. & Friederici, A. D. (2009). Mathematical logic in the human brain: syntax. *PLoS ONE*, *4*(5), e5599.

Fukui, N. (2011). Merge and bare phrase structure. In C. Boeckx (Ed.), *The Oxford handbook of linguistic minimalism* (p. 73-95). Oxford University Press.

Gallistel, C. R. (2006). The nature of learning and the functional architecture of the brain. In Q. Jing (Ed.), *Psychological science around the world, vol. 1* (p. 63-71). Sussex: Psychology Press.

Gallistel, C. R. & King, A. P. (2009). *Memory and the computational brain*. Malden, MA: Wiley-Blackwell.

Garrett, M. F., Bever, T. & Fodor, J. A. (1966). Active use of grammar in speech perception. *Perception and Psychophysics*, *1*, 30-2.

Gentner, T., Fenn, K. M., Margoliash, D. & Nusbaum, H. C. (2006). Recursive syntactic pattern learning by songbirds. *Nature*, *440*, 1204-1207.

Gersting, J. L. (1982). *Mathematical structures for computer science*. New York, New York: W. H. Freeman & Company.

Gibson, E. (1998). Linguistic complexity: locality of syntactic dependencies. *Cognition*, *68*, 1-76.

Gibson, E. & Pearlmutter, N. J. (1998). Constraints on sentence comprehension. *Trends in Cognitive Science*, *2*(7), 262-268.

Gleitman, L. R. & Papafragou, A. (2005). Language and thought. In R. G. Morrison & K. J. Hoyoal (Eds.), *The Cambridge handbook of thinking and reasoning* (p. 633-62). Cambridge University Press.

Gödel, K. (1934). On undecidable propositions of formal mathematical systems. In M. Davis (Ed.), *The undecidable* (p. 39-74). Dover Publications, Inc.

Goldberg, A. (2005). *Constructions at work: The nature of generalization in language*. Oxford, England: Oxford University Press.

Gómez, D. M., Bion, R. A. H. & Mehler, J. (2011). The word segmentation process as revealed by click detection. *Language and Cognitive Processes*, *26*(2), 212-23.

Gorrell, P. (1995). *Syntax and parsing*. Cambridge, England.: Cambridge University Press.

Graham, R., Knuth, D. & Patashnik, O. (1989). *Concrete mathematics*. Reading, MA.: Addison-Wesley Publishing Company.

Green, D. W. (1977). The immediate processing of sentences. *Quarterly Journal of Experimental Psychology*, *29*(1), 135-46.

Grodzinsky, Y. & Friederici, A. D. (2006). Neuroimaging of syntax and syntactic processing. *Current Opinion in Neurobiology*, *16*, 240-246.

Guasti, M. T. (2002). *Language acquisition: the growth of grammar*. Cambridge, MA.: The MIT Press.

Hadley, R. (1994a). Systematicity in connectionist language learning. *Mind and Language*, *9*(3), 247-272.

Hadley, R. (1994b). Systematicity revisited. *Mind and Language*, *9*(4), 431-444.

Hadley, R. (2004). On the proper treatment of semantic systematicity. *Minds and Machines*, *14*, 145-172.

Hale, J. T. (2011). What a rational parser would do. *Cognitive Science*, *35*(3), 399-443.

Halle, M. & Maratz, A. (1993). Distributed morphology and the pieces of inflection. In K. Hale & S. J. Keyser (Eds.), *The view from building 20* (p. 111-76). The MIT Press.

Halle, M. & Stevens, K. N. (1959). Analysis by synthesis. In W. Wathen-Dunn & A. M. Woods (Eds.), *Proceedings of Seminar on Speech, Compression and Processing.*

Halle, M. & Stevens, K. N. (1962). Speech recognition: A model and a program for research. *IRE Transactions of the PGIT IT*, *8*, 155-159.

Harley, T. (2001). *The psychology of language*. Sussex: Psychology Press.

Haugeland, J. (1981). Semantic engines: an introduction to mind design. In R. Cummins & D. Cummins (Eds.), *Minds, brains and computers: an anthology* (p. 34-50). Oxford, England: Blackwell Publishers.

Hauser, L. (1995). Doing without mentalese. *Behavior and Philosophy*, *23*, 42-47.

Hauser, M. D. (2009). Origin of the mind. *Scientific American*, *301*(3), 44-51.

Hauser, M. D., Chomsky, N. & Fitch, W. T. (2002). The faculty of language: what is it, who has it, and how did it evolve? *Science*, *298*, 1569-1579.

Hawkins, J. A. (2004). *Efficiency and complexity in grammars*. Oxford, England: Oxford University Press.

Hawkins, J. A. (2007). Processing typology and why psychologists need to know about it. *New Ideas in Psychology*, *25*, 87-1'7.

Heine, B. & Kuteva, T. (2007). *The genesis of grammar*. Oxford, England: Oxford University Press.

Hermer-Vázquez, L., Moffet, A. & Munkholm, P. (2001). Language, space, and the development of cognitive flexibility in humans: the case of two spatial memory tasks. *Cognition*, *79*, 263-299.

Hermer-Vázquez, L., Spelke, E. & Katsnelson, A. S. (1999). Sources of flexibility in human cognition: dual-task studies of space and language. *Cognitive Psychology*, *39*, 3-36.

Hilbert, D. (1902). Mathematical problems. *Bulletin of the American Mathematical Society*, *8*, 437-79.

Hinzen, W. (2008). Prospects for an explanatory theory of semantics. *Biolinguistics*, *2*(4), 348-363.

Hinzen, W. (2009). The successor function + LEX = human language? In K. K. Grohmann (Ed.), *InterPhases* (p. 25-47). Oxford University Press.

Hofstadter, D. (1979). *Gödel, Escher, Bach: an eternal golden braid*. The United States: Basic Books Inc.

Holmes, V. M. & Forster, K. I. (1970). Detection of extraneous signals during sentence recognition. *Perception and Psychophysics*, *7*(5), 297-301.

Hornstein, N. (2009). *A theory of syntax*. Cambridge, England: Cambridge University Press.

Hornstein, N. & Pietroski, P. (2009). Basic operations: minimal syntax-semantics. *Catalan Journal of Linguistics*, *8*, 113-139. Manuscript.

Hudson, R. (1996). The difficulty of (so-called) self-embedded structures. *UCL Working Papers in Linguistics*, *8*, 1-33.

Hunt, E. (1999). What is a theory of thought? In R. J. Sternberg (Ed.), *The nature of cognition* (p. 3-50). The MIT Press.

Hurford, J. (2004). Human uniqueness, learned symbols and recursive thought. *European Review*, *12*(4), 551-565.

Jackendoff, R. (1997). *The architecture of the language faculty*. Cambridge, MA.: The MIT Press.

Jackendoff, R. (2002). *Foundations of language*. Oxford: Oxford University Press.

Jackendoff, R. (2006). A parallel architecture perspective on language processing. *Brain Research*, *1146*, 2-22.

Jackendoff, R. & Pinker, S. (2005). The nature of the language faculty and its implications for evolution of language. *Cognition*, *97*, 211-225.

Jaeger, T. F. & Tily, H. (2011). On language 'utility': processing complexity and communicative efficiency. *WIREs Cognitive Science*, *2*, 323-35.

Jayaseelan, K. (2008). Bare phrase structure and specifier-less syntax. *Biolinguistics*, *2*(1), 87-106.

Johnson, D. & Lappin, S. (1997). A critique of the minimalist program. *Linguistics and Philosophy*, *20*(3), 273-333.

Joshi, A. K., Shanker, K. V. & Weir, D. (1990). The convergence of mildly context-sensitive formalisms. *Department of Computer and Information Science Technical Report (University of Pennsylvania)*, *N/A*, 1-65.

Just, M. A., Carpenter, P. A. & Woolley, J. D. (1982). Paradigms and processes in reading comprehension. *Journal of Experimental Psychology*, *111*(2), 228-38.

Karlsson, F. (2010). Recursion and iteration. In H. van der Hulst (Ed.), *Recursion and Human Language* (p. 43-68). De Gruyter Mouton.

Kayne, R. S. (1981). Unambiguous paths. In R. May & J. Koster (Eds.), *Levels of syntactic representation* (p. 143-83). Foris Publications.

Kayne, R. S. (1994). *The antisymmetry of syntax*. Cambridge, Massachusetts: The MIT Press.

Kinsella, A. R. (2009). *Language evolution and syntactic theory*. Cambridge, England: Cambridge University Press.

Kitcher, P. (2007). Freud's interdisciplinary fiasco. In A. Brook (Ed.), *The prehistory of cognitive science* (p. 230-49). Palgrave Macmillan.

Kleene, S. C. (1938). On notation for ordinal numbers. *The Journal of Symbolic Logic*, *3*(4), 150-55.

Kleene, S. C. (1943). Recursive predicates and quantifiers. In M. Davis (Ed.), *The undecidable* (p. 254-87). Dover Publications, Inc.

Kleene, S. C. (1952). *Introduction to metamathematics*. Amsterdam: North-Holland Publishing Co.

Knuth, D. (1997). *The art of computer programming (3 vols.)*. Upper Saddle River, NJ: Addison-Wesley.

Kremers, J. (2009). Recursive linearization. *Linguistic Review*, *26*(1), 135-166.

Langendoen, T. (1975). The relation of competence to performance. *Annals of New York Academy of Sciences*, *263*, 197-200.

Langendoen, T. (2003). Merge. In A. Carnie, M. Willie & H. Harley (Eds.), *Formal approaches to function in grammar: In honor of Eloise Jelinek* (p. 307-318). John Benjamins.

Langendoen, T. (2010). Just how big are natural languages? In H. van der Hulst (Ed.), *Recursion and Human Language* (p. 139-47). De Gruyter Mouton.

Lashley, K. S. (1951). The problem of serial order in behavior. In L. A. Jeffress (Ed.), *Cerebral mechanisms in behavior* (p. 112-31). Wiley-Blackwell.

Levinson, S. C. & Evans, N. (2010). Time for a sea-change in linguistics: Response to comments on 'the myth of language universals'. *Lingua*, *120*, 2733-58.

Liu, Y. A. & Stoller, S. D. (1999). From recursion and iteration: what are the optimizations? *SIGPLAN Not.*, *34*(11), 73-82.

Lobina, D. J. (2011a). Recursion and the competence/performance distinction in AGL tasks. *Language and Cognitive Processes*, *26*(10), 1563-86.

Lobina, D. J. (2011b). "A running back"; and forth: A review of *Recursion and Human Language*. *Biolinguistics*, *5*(1-2), 151-69.

Lobina, D. J. & García-Albea, J. E. (2009). Recursion and cognitive science: data structures and mechanisms. In N. A. Taatgen & H. van Rijn (Eds.), *Proceedings of the 31th Annual Conference of the Cognitive Science Society* (p. 1347-1352).

Ludlow, P. (2009). *Recursion, legibility, use.* University of Massachusetts Amherst Conference on Recursion (May 2009).

Ludlow, P. (2011). *The philosophy of generative linguistics*. Oxford, England: Oxford University Press.

Luuk, E. & Luuk, H. (2011). The redundancy of recursion and infinity for natural language. *Cognitive Processing*, *12*(1), 1-11.

MacDonald, M., Pearlmutter, N. J. & Seidenberg, M. S. (1994). Lexical nature of syntactic ambiguity resolution. *Psychological Review*, *101*(4), 676-703.

MacWhinney, B. (2009). The emergence of linguistic complexity. In T. Givón & M. Shibatani (Eds.), *Syntactic complexity* (p. 406-432). John Benjamins Publishing Company.

Mancini, S., Molinaro, N., Rizzi, L. & Carreiras, M. (2011). When persons disagree: an ERP study of unagreement in Spanish. *Psychophysiology*, *48*(10), 1361-71.

Marcus, G. F. (2001). *The algebraic mind*. Cambridge, MA.: The MIT Press.

Margolis, E. & Laurence, S. (1999). *Concepts*. Cambridge, Massachusetts: The MIT Press.

Marr, D. (1982). *Vision: A computational investigation into the human representation and processing of visual information*. San Francisco: W. H. Freeman & Company.

Marslen-Wilson, W. D. (1985). Speech shadowing and speech comprehension. *Speech Communication*, *4*, 55-73.

Matthews, R. (1979). Are the grammatical sentences of a language a recursive set? *Synthese*, *40*(2), 209-224.

Matthews, R. (1992). Psychological reality of grammars. In A. Kasher (Ed.), *The Chomkyan turn* (p. 182-99). Blackwell Publishers.

Matthews, R. (1997). Can connectionists explain systematicity? *Mind and Language*, *12*(2), 154-177.

Matthews, R. (2006). Knowledge of language and linguistic competence. *Philosophical Issues*, *16*, 200-220.

McCarthy, J. (1963). A basis for a mathematical theory of computation. In P. Braffort & D. Hirshberg (Eds.), *Computer programming and formal systems* (p. 33-70). North-Holland Publishing Co.

McNeill, D. (1975). The place of grammar in a theory of performance. *Annals of the New York Academy of Sciences*, *263*, 171-7.

Medeiros, D. (2008). Optimal growth in phrase structure. *Biolinguistics*, *2*(2-3), 152-195.

Miller, G. A. (1956). The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review*, *63*(2), 81-97.

Miller, G. A. (1967). *The psychology of communication: seven essays*. The United States: Basic Books.

Miller, G. A. (1975). Some comments on competence and performance. *Annals of the New York Academy of Sciences*, *263*, 201-04.

Miller, G. A. & Chomsky, N. (1963). Finitary models of language users. In R. D. Luce, R. R. Bush & E. Galanter (Eds.), *Handbook of mathematical psychology, vol. 2* (p. 419-492). John Wiley and sons, Inc.

Miller, G. A., Galanter, E. & Pribram, K. H. (1960). *Plans and the structure of behaviour*. New York, New York: Holt, Rinehart and Winston, Inc.

Miller, G. A. & Isard, S. (1963). Some perceptual consequences of linguistic rules. *Journal of Verbal Learning and Verbal Behavior*, *2*, 217-28.

Miller, G. A. & Isard, S. (1964). Free recall of self-embedded English sentences. *Information and Control*, *7*, 292-303.

Miller, P. H. (1999). *Strong generativity capacity*. Stanford, California: CSLI publications.

Miller, P. H., Kessel, F. S. & Flavell, J. H. (1970). Thinking about people thinking about people thinking about...: a study of social cognitive development. *Child Development*, *41*(3), 613-623.

Minsky, M. (1967). Why programming is a good medium for expressing poorly understood and sloppily formulated ideas. In M. Krampen & P. Seitz (Eds.), *Design and planning II - computers in design and communication* (p. 117-122). New York: Hastings House Publishers.

Moravcsik, J. M. (1975). *Aitia* as generative factor in Aristotle's philosophy. *Dialogue*, *14*(4), 622-38.

Moro, A. (2000). *Dynamic antisymmetry*. Cambridge, Massachusetts: The MIT Press.

Moro, A. (2008). *The boundaries of Babel*. Cambridge, MA.: The MIT Press.

Moschovakis, Y. N. (1998). On founding the theory of algorithms. In H. G. Dales & G. Oliveri (Eds.), *Truth in mathematics* (p. 71-104). Clarendon Press.

Moschovakis, Y. N. (2001). What is an algorithm? In B. Engquist & W. Schmid (Eds.), *Mathematics unlimited: 2001 and beyond* (p. 919-936). Springer.

Moschovakis, Y. N. & Paschalis, V. (2008). Elementary algorithms and their implementations. In S. B. Cooper, B. Lowe & A. Sorbi (Eds.), *New computational paradigms* (p. 81-118). Springer.

Mukherji, N. (2010). *The primacy of grammar*. Cambridge, Massachusetts: The MIT Press.

Neeleman, A. & van de Koot, J. (2006). On syntactic and phonological representations. *Lingua*, *116*(10), 1524-1552.

Neeleman, A. & van de Koot, J. (2010). Theoretical validity and psychological reality of the grammatical code. In M. Everaert, T. Lentz, H. De Mulder, O. Nilsen & A. Zondervan (Eds.), *The linguistics enterprise* (p. 183-212). John Benjamins.

Nelson, T. O. (1999). Cognition versus Metacognition. In R. J. Sternberg (Ed.), *The nature of cognition* (p. 625-44). The MIT Press.

Nevins, A., Pesetsky, D. & Rodrigues, C. (2007). Pirahã exceptionality: a reassessment. *Language*, *85*(2), 355-404.

Newell, A. (1980). Physical symbol systems. *Cognitive Science*, *4*, 135-183.

Oppenheimer, L. (1986). Development of recursive thinking. *International Journal of Behavioral Development*, *9*(3), 401-411.

Parker, A. R. (2006). Evolving the narrow language faculty: was recursion the pivotal step? In A. Cangelosi, A. D. M. Smith & K. Smith (Eds.), *The evolution of language: Proceedings of the Sixth international conference on the evolution of language.*

Peacocke, C. (2005). Joint attention: Its nature, reflexivity, and relation to common knowledge. In N. Eilan, C. Hoerl, T. McCormack & J. Roessler (Eds.), *Joint attention: Communication and other minds* (p. 298-324). Oxford University Press.

Perfors, A., Tenenbaum, J., Gibson, E. & Regier, T. (2010). How recursive is language? A Bayesian exploration. In H. van der Hulst (Ed.), *Recursion and Human Language* (p. 159-178). De Gruyter Mouton.

Perruchet, P. & Rey, A. (2005). Does the mastery of center-embedded linguistic structures distinguish humans from nonhuman primates? *Psychonomic Bulletin and Review*, *12*(2), 307-313.

Phillips, C. (2003). Linear order and constituency. *Linguistic Inquiry*, *34*(1), 37-90.

Phillips, C. & Lewis, S. (2009). *Derivational order in syntax: Evidence and architectural consequences.* Manuscript.

Phillips, C. & Wagers, M. (2009). Relating structure and time in linguistics and psycholinguistics. In G. Gaskell (Ed.), *The Oxford handbook of psycholinguistics* (p. 739-756). Oxford University Press.

Phillips, S. (1994a). Connectionism and systematicity. In *Proceedings of the Fifth Australian Conference on Neural Networks.*

Phillips, S. (1994b). Strong systematicity within connectionism: the tensor-recurrent network. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society.*

Piattelli-Palmarini, M. (1980). *Language and learning: the debate between Jean Piaget and Noam Chomsky*. London, England: Routledge and Kegan Paul.

Piattelli-Palmarini, M. & Uriagereka, J. (2008). Still a bridge too far? Biolinguistic questions for grounding language on brains. *Physics of Life Reviews*, *5*, 207-224.

Piattelli-Palmarini, M., Uriagereka, J. & Salaburu, P. (2009). *Of minds and language: a dialogue with Noam Chomsky in the Basque Country*. Oxford, England: Oxford University Press.

Pickering, M. J. & van Gompel, R. P. G. (2006). Syntactic parsing. In M. Traxler & M. A. Gernsbacher (Eds.), *The handbook of psycholinguistics* (p. 455-503). Academic Press Inc.

Pinker, S. (1994). *The language instinct.* London, England: Penguin Books.

Pinker, S. (2005a). A reply to Jerry Fodor on How the Mind Works. *Mind and Language*, *20*(1), 33-38.

Pinker, S. (2005b). So how *does* the mind work? *Mind and Language*, *20*(1), 1-24.

Pinker, S. & Jackendoff, R. (2005). The faculty of language: what's special about it? *Cognition*, *95*, 201-236.

Poeppel, D., Idsardi, W. J. & van Wassenhove, V. (2008). Speech perception at the interface of neurobiology and linguistics. *Philosophical Transactions of the Royal Society B*, *363*, 1071-1086.

Poletiek, F. H. (2002). Implicit learning of a recursive rule in an artificial grammar. *Acta Psychologica*, *111*, 323-335.

Post, E. (1921). Introduction to a general theory of elementary propositions. *American Journal of Mathematics*, *43*(3), 163-85.

Post, E. (1936). Finite combinatory processes. Formulation I. In M. Davis (Ed.), *The undecidable* (p. 288-291). Dover Publications, Inc.

Post, E. (1943). Formal reductions of the general combinatorial decision problem. *American Journal of Mathematics*, *65*(2), 197-215.

Post, E. (1944). Recursively enumerable sets of positive integers and their decision problems. In M. Davis (Ed.), *The undecidable* (p. 304-337). Dover Publications, Inc.

Post, E. (1947). Recursive unsolvability of a problem of Thue. *Journal of Symbolic Logic*, *12*, 1-11.

Postma, G. & Rooryck, J. (2007). *Phase-recursion, restricted linguistic systems and full language.* lingBuzz/000470.

Pratt-Hartmann, I. (2010). Computational complexity in natural language. In A. Clark, C. Fox & S. Lappin (Eds.), *The Handbook of Computational Linguistics and Natural Language Processing* (p. 43-73). Addison-Wesley.

Pullum, G. & Scholz, B. (2010). Recursion and the infinitude claim. In H. van der Hulst (Ed.), *Recursion and Human Language* (p. 113-138). The Netherlands: De Gruyter Mouton.

Pylyshyn, Z. (1973). The role of competence theories in cognitive psychology. *Journal of Psycholinguistic Research*, *2*(1), 21-50.

Pylyshyn, Z. (1984). *Computation and cognition.* Cambridge, MA.: The MIT Press.

Pylyshyn, Z. (1989). Computing in cognitive science. In M. I. Posner (Ed.), *Foundations of cognitive science* (p. 49-92). Cambridge, MA.: The MIT Press.

Radford, A. (2004). *Minimalist syntax: exploring the structure of English*. Cambridge: Cambridge University Press.

Reber, A. S. & Anderson, J. R. (1970). The perception of clicks in linguistic and nonlinguistic messages. *Perception and Psychophysics*, *8*(2), 81-89.

Reich, P. A. (1969). The finiteness of natural language. *Language*, *45*(4), 831-43.

Reinhart, T. (2006). *Interface strategies*. Cambridge, MA.: The MIT Press.

Rice, G. (1965). Recursion and iteration. *Communications of the ACM*, *8*(2), 114-115.

Richards, N. (2010). *Uttering trees*. Cambridge, Massachusetts: The MIT Press.

Roberts, E. (2006). *Thinking recursively with Java*. Hoboken, NJ: John Wiley and Sons, Inc.

Roeper, T. (2007). *The prism of language*. Cambridge, Massachusetts: The MIT Press.

Roeper, T. (2009). *Microscopic minimalism.* Boston University Plenary Address.

Roeper, T. (2011). The acquisition of recursion: how formalism articulates the child's path. *Biolinguistics*, *5*(1-2), 57-86.

Rogers, J. & Pullum, G. K. (2011). Aural pattern recognition experiments and the subregular hierarchy. *Journal of Logic, Language and Information*, *20*(3), 329-42.

Rohrmeier, M. (2011). Towards a generative syntax of tonal harmony. *Journal of Mathematics and Music*, *5*(1), 35-53.

Rosser, J. B. (1939). An informal exposition of proofs of Gödel's Theorem and Church's Theorem. In M. Davis (Ed.), *The undecidable* (p. 223-29). Dover Publications, Inc.

Rumelhart, D. (1989). The architecture of the mind: a connectionist approach. In J. Haugeland (Ed.), *Mind design II* (p. 205-232). Cambridge, MA.: The MIT Press.

Saffran, J. R., Aslin, R. N. & Newport, E. L. (1996). Statistical learning by 8-month-old infants. *Science*, *274*(5294), 1926-1928.

Sag, I. A., Wasow, T. & Bender, E. M. (2003). *Syntactic theory: a formal introduction*. Stanford, California: CSLI publications.

Sahin, N., Pinker, S., Cash, S. S., Schomer, D. & Halgren, E. (2009). Sequential processing of lexical, grammatical, and phonological information within Broca's area. *Science*, *326*, 445-449.

Samuels, R. (2002). The spatial reorientation data do *not* support the thesis that language is the medium of cross-modular thought. *Behavioral and Brain Sciences*, *25*, 697-8.

Samuels, R. (2010). Classical computationalism and the many problems of cognitive relevance. *Studies in History and Philosophy of Science Part A*, *41*(3), 280-93.

San Juan, J. Huarte de. (1575). *Examen de ingenios*. Madrid, Spain: Ediciones Cátedra.

Sauerland, U. (2010). *Experimental evidence for complex syntax in pirahã.* lingBuzz/001095.

Sauerland, U. & Trotzke, A. (2011). Biolinguistic perspectives on recursion: Introduction to the special issue. *Biolinguistics*, *5*(1-2), 1-9.

Schafer, A. J. (1997). *Prosodic parsing: the role of prosody in sentence comprehension.* PhD thesis, University of Massachusetts, Amherst.

Scheepers, C., Sturt, P., Martin, C. J., Myachykov, A., Teevan, K. & Viskupova, I. (2011). Structural priming across cognitive domains: From simple arithmetic to relative-clause attachment. *Psychological Science.*, *DOI: 10.1177/0956797611416997*.

Sebastián-Gallés, N., Martí, M. A., Carreiras, M. & Cuetos, F. (2000). *LEXESP. léxico informatizado del español*. Barcelona: Edicions Universitat de Barcelona.

Segal, G. M. A. (2001). On a difference between language and thought. *Linguistics and Philosophy*, *124*, 125-129.

Sieg, W. (1997). Step by recursive step: Church's analysis of effective calculability. *The Bulletin of Symbolic Logic*, *3*(2), 154-80.

Sieg, W. (2006). Gödel on computability. *Philosophia Mathematica*, *3*(14), 189-207.

Simon, H. (1962). The architecture of complexity. *Proceedings of the American Philosophical Society*, *106*(4), 467-82.

Smith, B. C. (2006). What I know when I know a language. In E. Lepore & B. C. Smith (Eds.), *The Oxford Handbook of Philosophy of Language* (p. 941-82). Oxford University Press.

Smith, B. C. (2008). What remains of our knowledge of language? Reply to Collins. *Croatian Journal of Philosophy*, *8*(22), 57-76.

Smith, J. C. (1991). *Historical foundations of cognitive science*. The Netherlands: Kluwer Academic Publishers.

Smith, N. V. (1999). *Chomsky. Ideas and ideals.* Cambridge, England: Cambridge University Press.

Smolensky, P. (1991). Connectionism and the language of thought. In D. Cummins R. & Cummins (Ed.), *Minds, brains and computers: an anthology* (p. 286-306). Oxford: Blackwell Publishers.

Soare, R. (1996). Computability and recursion. *The Bulletin of Symbolic Logic*, *2*(3), 284-321.

Soare, R. (2007a). Computability and incomputability. In *Proceedings of the Third Conference on Computability in Europe* (Vol. 4497).

Soare, R. (2007b). Incomputability, Turing functionals, and open computing. In *Computation and Logic in the Real World Conference.*

Soare, R. (2009). Turing oracles machines, online computing, and three displacements in computability theory. *Annals of Pure and Applied Logic*, *160*, 368-99.

Soschen, A. (2006). Natural law: the dynamics of syntactic representations in MP. *Linguistics in Postdam*, *25*, 43-76.

Soschen, A. (2008). On the nature of syntax. *Biolinguistics*, *2*, 196-224.

Stabler, E. (2010). *Recursion in grammar and performance.* Recursion conference in Amherst.

Stabler, E. (2011). Computational perspectives on minimalism. In C. Boeckx (Ed.), *The Oxford Handbook of Linguistic Minimalism* (p. 616-641). Oxford University Press.

Starke, M. (2004). On the inexistence of specifiers and the nature of heads. In A. Belleti (Ed.), *Structures and beyond* (p. 252-269). Oxford University Press.

Steedman, M. (1994). *Natural language processing* (Tech. Rep.). Department of Computer and Informacion Science Technical Report, University of Pennsylvania.

Steedman, M. (2000). *The syntactic process*. Cambridge, Massachusetts: The MIT Press.

Stich, S. (1971). What every speaker knows. *Philosophical Review*, *80*, 476-96.

Stroik, T. S. (2009). *Locality in minimalist syntax.* Cambridge, Massachusetts: The MIT Press.

Tiede, H.-J. & Stout, L. N. (2010). Recursion, infinity and modelling. In H. van der Hulst (Ed.), *Recursion and Human Language* (p. 147-58). De Gruyter Mouton.

Tomalin, M. (2006). *Linguistics and the formal sciences*. Cambridge, England: Cambridge University Press.

Tomalin, M. (2007). Reconsidering recursion in syntactic theory. *Lingua*, *117*, 1784-1800.

Tomalin, M. (2011). Syntactic structures and recursive devices: a legacy of imprecision. *Journal of Logic, Language and Information*, *20*(3), 297-315.

Townsend, D. & Bever, T. G. (2001). *Sentence comprehension*. Cambridge, MA.: The MIT Press.

Turing, A. M. (1936). On computable numbers, with an application to the Entscheidungsproblem. In M. Davis (Ed.), *The undecidable* (p. 115-153). Dover Publications, Inc.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, *59*, 433-60.

Turing, A. M. (1954). Solvable and unsolvable problems. In B. J. Copeland (Ed.), *The essential Turing* (p. 576-95). Oxford University Press.

Twyman, A. D. & Newcombe, N. S. (2010). Five reasons to doubt the existence of a geometric module. *Cognitive Science*, *34*(7), 1315-1357.

Tyler, L. K. & Marslen-Wilson, W. D. (1977). The on-line effects of semantic context on syntactic processing. *Journal of Verbal Learning and Verbal Behavior*, *16*, 683-692.

Uriagereka, J. (2008). *Syntactic anchors*. Cambridge, England: Cambridge University Press.

van der Hulst, H. (2010a). A note on recursion and phonology. In H. van der Hulst (Ed.), *Recursion and Human Language* (p. 301-42). De Gruyter Mouton.

van der Hulst, H. (2010b). *Recursion and human language*. Berlin, Germany: De Gruyter Mouton.

van Gelder, T. (1990). Compositionality: a connectionist variation on a classical theme. *Cognitive Science*, *14*, 355-384.

van Gelder, T. & Niklasson, N. (1994a). Can connectionist models exhibit non-classical structure sensitivity? In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society.* Hillsdale, NJ: Erlbaum.

van Gelder, T. & Niklasson, N. (1994b). Classicalism and cognitive architecture. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society.* Hillsdale, NJ: Erlbaum.

van Gelder, T. & Niklasson, N. (1994c). On being systematically connectionist. *Mind and Language*, *9*, 288-302.

van Gompel, R. P. G. & Pickering, M. J. (2009). Syntactic parsing. In G. Gaskell (Ed.), *The Oxford Handbook of Psycholinguistics* (p. 289-307). Oxford University Press.

van Heijningen, C. A. A., de Visser, J., Zuidema, W. & ten Cate, C. (2009). Simple rules can explain discrimination of putative recursive syntactic structures by a songbird species. *Proceedings of the National Academy of Sciences*, *106*(48), 20538-20543.

Wagers, M. W. & Phillips, C. (2009). Multiple dependencies and the role of grammar in real-time comprehension. *Journal of Linguistics*, *45*, 395-433.

Wagner, M. (2010). Prosody and recursion in coordinate structures and beyond. *Natural Language and Linguistic Theory*, *28*, 183-237.

Weinberg, A. (2001). A minimalist theory of human sentence processing. In S. Epstein & N. Hornstein (Eds.), *Working minimalism* (p. 283-314). The MIT Press.

Wirth, N. (1986). *Algorithms and data structures*. USA: Prentice Hall Publishers.

Wolf, F. & Gibson, E. (2003). Parsing: overview. In L. Nadel (Ed.), *Encyclopedia of cognitive science* (p. 465-76). MacMillan.

Xu, Y. & Corkin, S. (2001). H.M. revisits the Tower of Hanoi puzzle. *Neuropsychology*, *15*(1), 69-79.

Yang, C. (2004). Universal grammar, statistics or both? *Trends in Cognitive Science*, *8*(10), 451-456.

Zwart, J.-W. (2009). Prospects for a top-down derivation. *Catalan Journal of Linguistics*, *8*, 161-187.

Zwart, J.-W. (2011a). Recursion in language: a layered-derivation approach. *Biolinguistics*, *5*(1-2), 43-56. Paper presented at the TIN-dag, Utrecht.

Zwart, J.-W. (2011b). Structure and order: asymmetric merge. In C. Boeckx (Ed.), *The Oxford Handbook of linguistic minimalism* (p. 96-118). Oxford University Press.