Recursion and the Competence-Performance distinction in AGL tasks

David J. Lobina

18th September 2010

Abstract

The term "recursion" is employed in at least four distinct theoretical senses within cognitive science. Some of these senses in turn relate to the different levels of analysis described by David Marr some twenty years ago; namely, the underlying competence capacity (the *computational* level), the performance operations employed in real-time processing (the *algorithmic* level), and the neural implementation (the *hardware* level). It is demonstrated here that the current focus of the artificial grammar learning literature on recursion blurs the different levels, resulting in three main corollaries: (1) the literature cannot tell us anything about the first level, as the explanation for the role of recursion therein lies elsewhere; (2) it has not studied the possible recursive processing of artificial strings properly, as studies have so far focused on the correct processing of the internal hierarchy of recursive structures, but none have attempted to probe the underlying processing operations; and (3) as a consequence, much cannot in fact be known about the neural basis of recursion. It is furthermore argued that, in general, these studies can actually tell us very little about the language faculty, and some programmatic remarks are described and defended towards a more coherent study.

Keywords: Recursion; Artificial Grammar Learning; Competence; Performance; Hierarchical Structures.

1 Four meanings of recursion

The notion of "recursion" is currently undergoing something of a promiscuity explosion of different connotations in the Cognitive Science literature. Nevertheless, there are, I think, at least four distinct theoretical constructs to which this core notion appropriately applies. First of all, it may be employed in the specification of technical terms when these are "defined by induction", which is in fact the original sense of recursion in mathematics (Soare, 1996). A recursive definition, as is also known, consists in 'defining a function by specifying each of its values in terms of previously defined values' (Cutland, 1980, p. 32), as shown here for the class of the factorial functions: Def. n!; if $n = 1$, then $n! = 1$ (base case); otherwise: if $n > 1$, then $n! = n \times (n-1)!$ (recursive step). Bar-Hillel (1953) argues that the social sciences may benefit from employing such a definitional technique, and Chomsky (1995) does in fact offer a recursive definition of a *syntactic object* (p.243). Many more examples are likely to be found in the literature, but a recursive definition should not be confused with an "inductive definition", or "mathematical induction", a closely related construct that also makes use of recursion, but that additionally contains an "inductive hypothesis" (see Kleene 1952, p.260 et seq. for clarifications).

Secondly, recursion may appear as a general and central property of algorithms and generative systems. Thus, in the *analysis of algorithms* discipline, systems of recursive equations have been employed to formalise the notion of an algorithm qua formal object (see, especially, McCarthy 1963) and some scholars have proposed that these recursive equations subsume a specific mapping function, termed a "recursor" (Moschovakis, 1998, 2001; Moschovakis & Paschalis, 2008). A recursor is said to describe the structure of an algorithm and, in this sense, algorithms *are* recursors. Production systems of rewriting rules also contain recursion as a central property, but not simply in those specific cases in which the same symbol appears on both the left- and right-hand side of a rewrite rule. Consider, for example, the underlying transformation that converts some structure $\phi_1, \ldots \phi_n$ into some structure ϕ_{n+1}; the $\rightarrow$ relation can then 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}' (Chomsky & Miller, 1963, p.284). This is basically the successor function, one of the primitive class of recursive functions. Whereas the former cases involve an *internal* application of recursion *within* production systems, the latter is a *global* property of *collections* of rewriting rules *qua* production systems.

The successor function also underlies what is known as the "iterative conception of set", a process in which sets are 'recursively generated at each stage', a statement that is to be understood as 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, 223). The current characterisation of Merge, the building operation at the heart of the language faculty, as a set-formation operator seems to be akin to this interpretation of recursion (see Chomsky 2008 and Soschen 2008).[1]

A third sense pertains to the study of computational processes, which is the interest of much of applied computer science. This sense refers not to the algorithm qua formal object, but to its actual *implementation*; that is, it is the study of the so-called *models of computation*. A recursive process, then, is one in which an operation calls itself, creating chains of deferred operations, and is usefully contrasted

[1]It is not the place to carry out an exegesis of Chomsky's enormous output, where this recursive property is not always treated in a consistent manner. I do, however, claim that recursion qua general property of algorithms and generative systems is a common thread running through his writings, providing a smooth transition from the employment of rewriting rules to Merge. This precise connotation ought not to be confused with the other interpretations I am describing, which also appear in his writings.

with an iterative process, wherein an operation reapplies in succession (Abelson, Sussman & Sussman, 1996). The distinction has to do with the way these two processes apply, and *not* with what *sort* of operation they execute or with what sort of structures they produce or operate upon. 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 (Roberts, 2006). Put bluntly, that is, that 'all recursive relations can be reduced to recurrence or iterative relations' (Rice, 1965, p.114). This point is significantly unappreciated in the cognitive sciences, due in my opinion to a number of simple misunderstandings that space will only allow me to allude to below. Nevertheless, I am describing here a model in which sub-routines take place, and there are many examples of such processes in cognitive science (Miller, Galanter & Pribram 1960 and Simon 1962 are some of the most prominent examples). This sense, it will have been noticed, is mainly pertinent to cognitive psychology, due to its (perhaps narrow) focus on real-time processing.

The fourth, and final, sense refers to *structures* rather than operations. Recursive data structures are defined by the U.S. National Institute of Standards and Technology as any object or class 'that is partially composed of smaller or simpler instances of the same data structure'[2]; that is, any structure which includes an abstraction of itself (an X within an X). The prototypical cases here are the "trees within trees" so familiar to generative grammar. It is important to establish what the X in the 'X within an X' is, so as to identify a recursive structure that is in fact of some relevance.

It is doubtless the case that all these constructs are sufficiently independent to merit individual investigation, but it is common to find scholars who either conflate some of these senses or focus on one when they actually mean another. Nevertheless, it may have been noticed that the constructs being considered here seem to be relevant to different levels of explanation, as David Marr delineated them more than twenty years ago (Marr, 1982). The general, recursive property of generative systems, for example, clearly pertains to the study of cognitive structures and capacities in terms of the abstract mapping between inputs and outputs and the formal properties deriving therefrom; that is, to the computational level of explanation, the study of *faculties* as understood by Chomsky (1980). On the other hand, the algorithmic level of explanation focuses on the operations of processing *modules*, in the sense of Fodor (1983), and it might be of interest to find out if these operations are executed recursively or iteratively, with all the concomitant phenomena that they entail (i.e., memory load, architectural complexity, efficiency, etc). Finally, the hardware level is supposed to show us how the previous properties of the mind are realised in the human brain, but it clearly needs to wait for tangible results from the other two levels of explanation. Naturally, both recursive definitions and structures may be of some relevance at either level, but it clearly remains to be determined, both theoretically and experimentally, what precise role they actually

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

play.[3]

It is not surprising that the role of recursion in all these guises has become a contested topic in cognitive science, and the artificial grammar learning (AGL) literature, in particular, contains a burgeoning number of studies relating to this. The article at hand aims to use the distinctions laid out above to demonstrate important shortcomings in the way AGL scholars have approached the subject. The shortcomings range from inadequate definitions and unsupported assumptions and extrapolations, to a complete disregard for Marr's general approach as a coherent framework for productively studying cognition.[4]

2 The state of the art, roughly speaking

The current focus on recursion in the AGL literature seems to follow from the assumption that studying how subjects deal with regular patterns of letter strings allows you to probe exclusively syntactic properties of linguistic knowledge and use. Recursion is, after all, an intrinsically "syntactic" notion and AGL tasks, founded on sequences of "nonsense" syllables that abstract away from any confounding semantic factors are, we are assured, intrinsically syntactic tasks. This being the case, AGL experiments would seem to provide us with information regarding the mechanisms underlying language.

Consider, for example, the different string patterns employed in AGL studies. Such strings are argued to correspond to different formal grammars that generate them.[5] In turn, these grammars, usually characterised in terms of rewriting production systems, can be located in a hierarchy of ever-increasing expressive power (the so-called Chomsky Hierarchy; Chomsky 1956). As a result, experimental data have been used to discern what sort of grammar underlies the subjects' abilities to process/learn the different patterns. This has usually been investigated, in humans at least, with a grammatical judgement paradigm. Subjects are initially exposed, in the training phase of the experiment, to regular patterns of strings. They are then 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

[3]Soare (1996) offers an analysis of the way recursion has been understood within mathematics, identifying four different connotations (I, like him, have focused on the original interpretations here). Tomalin (2007) discusses Soare's paper, adds another sense, and applies the same sort of analysis to linguistics. He does not however treat the recursion/iteration in sufficient detail; or the iterative conception of set, recursors and recursive data structures (let alone processing). Both Kinsella (2009) and Fitch (2010) discuss the role of recursion in mathematics and computer science and its application to linguistics, but with significant misunderstandings and problems. A more cogent, albeit brief, description can be found in Lobina & García-Albea (2009).

[4]What this essay will not treat is the larger issues regarding AGL, such as the distinction between implicit and explicit learning, the strategies subjects are employing (rules, comparison, chunking et cetera) and much else (for a review, see Pothos 2007).

[5]In this sense, a given set of strings is a formal language generated by a formal grammar.

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.

This, at least, is certainly what Fitch & Hauser (2004) —the publication that spurred the current interest of AGL studies on recursion— were after when they compared the learning of two "grammars", one a finite-state grammar and the other a phrase-structure grammar, by both humans and cotton-top tamarin monkeys. The finite-state grammar (FSG), exemplified by rules such as $A \rightarrow a$ and $A \rightarrow aB$, can generate $(AB)^n$ strings, while the phrase-structure grammar (PSG; usually known as a context-free grammar), which contains recursive rules such as $S \rightarrow a(S)b$, can in turn generate A^nB^n strings. The latter strings are sometimes referred to in the literature as "centre-embedded" structures, given that further *AB* pairs can be inserted in the middle.

Significantly, nowhere in their paper do they suggest that subjects process these strings by directly employing the corresponding "grammar", that is, by employing the rewriting rules as parsing operations. In fact, they clearly state that they did *not* study the different strategies that could have been employed, the "performance variables" (idem., p.378), as what they were interested in was the expressive power of the underlying *competence*. That is, they wanted to know what sort of "grammar" humans and monkeys had internalised, an obvious precondition for being able to learn/process the strings in the first place, and chose not to discuss the relationship between the grammar and the parser. Further, they nowhere actually mention the term "recursion". Nevertheless, their conclusion that only the human subjects were able to process A^nB^n strings, combined with the fact that the strings investigated were explicitly generated by a grammar that contains a recursive step, has naturally led the subsequent literature to focus on whether recursion is in fact the distinguishing property the two species.[6]

These AGL studies, in their attempt to probe if subjects were literally and directly employing the corresponding grammars, have proceeded more or less as follows. Sets of production rules are used to generate sequences of nonsense syllables, with special emphasis falling on the PSG class, given that these can generate centre-embedded strings, which is taken to be the (main) locus of recursion. Appropriate tests are then used to check if subjects are able to process the strings correctly, as grammatical judgements are supposedly not enough, given that the different strings can be generated by various algorithms, and not necessarily with any of the formal grammars so far described. Specifically, Corballis (2007) has proposed that in order to demonstrate 'true recursion' (p.702), subjects would have to realise that some *As* are paired with some *Bs* within the A^nB^n strings. That is,

[6]The following literature has focused on issues that will be marginal here, such as whether other species are able to correctly process the supposedly self-embedded strings (Gentner et al., 2006; Marcus, 2006), and if so, if they are doing it by employing the recursive rules of a given grammar or other simpler strategies (van Heijningen et al., 2009). Despite strong denunciations to the contrary (viz., Coleman et al. 2004), Fitch & Hauser (2004) did not claim that humans were employing recursive rules as a strategy to learn the PSG pattern. I will also not discuss the different methodological techniques these studies have employed, as my qualms lie elsewhere.

‘recursive syntactic parsing’ would be demonstrated if subjects bound *AB* pairs from the outside inwards (idem.), and “cues” can be employed to indicate these internal relationships.

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 FSG and PSG strings, while Brodmann’s Area 44/45 (i.e., Broca’s area) is additionally only activated during the processing of PSG, 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).[7]

I will discuss this in detail in the next section, but suffice to say here that the idea seems to be that if you manage to make sure that subjects process the hierarchical structure correctly, that is, that they are embedding strings into other strings (a particular of the general strategy of “putting something inside itself”), this suggests recursion.

There are some terminological problems in this way of approaching the matter, and I will certainly delve into this point more deeply below. More importantly, however, I will claim that the current AGL literature finds itself in a similar position to what psycholinguists faced during the height of generative grammar in the 60s and 70s, and similar solutions are needed.

[7]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. There is no reason to think that these cues resulted in *AB* pairs; rather, it is likely that subjects were sensitive to the changes of voice and to the pitch transitions rather than to the structure.

3 The significance of David Marr's levels of explanation

3.1 Grammars and parsers

The early stages of the interactions between generative grammarians and psychologists were characterised by the attempt to probe "the psychological reality" of linguistic theories, by which it was meant whether the properties and rules that linguists postulated were used in real-time processing. Linguists never actually made such claims, and it took some time before (some) psycholinguists settled on a reputed connection between the two disciplines, establishing a relationship that was meant to salvage the psychological reality of linguistics.

It is probably safe to say that there is no consensus on the matter, but it seems to me that the position defended in Fodor, Bever & Garrett (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 distal stimuli —the actual physical phenomenon received by the organism— but how these are represented (the proximal stimuli); hence, their attempt to subsume the perceived stimuli under schema. They certainly had a theory of the distal stimuli —this was provided by the physical sciences— but the principles and properties governing their schema-descriptions were of an altogether different nature, and they had no viable theoretical account of the proximal stimuli (idem., p.xvi). Fodor et al.'s proposal addressed precisely this fault, arguing 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, research on the intrinsic properties of the structural descriptions forms an independent domain of study, and in the case of generative linguistics, these properties have been described in terms of the computations and representations that underlie the mental capacity for language. Marr called this sort of work the computational level of analysis, which focuses on properties of the grammar and not the parser.

There is some confusion about this point in the AGL literature. Thus, some scholars take the result that subjects did not compute long-distance dependencies to cast doubt on claims that a context-free grammar is employed, which is supposed to carry over to the study of the language faculty itself (Perruchet & Rey 2005, p.311; Gentner et al. 2006, p.1206). The latter is however unwarranted, and seems to rest on a misunderstanding of Chomsky's position. They take Chomsky to have argued that a system more powerful than a FSG is needed to explain language *use*, but nothing of the sort has ever been claimed. Indeed, Chomsky (1963) pointed out that even though a finite automaton could not capture the basic facts of linguistic *competence*, the system underlying linguistic *performance* could well be such a thing, with all the obvious corollaries that follow from this fact (p.390). That is, the error these scholars are making is to extrapolate a property of the parser from a property of the grammar, but the two are distinct levels of explanation.

There is little reason to believe that AGL studies have a theory of the structural descriptions that are being built up during the processing of artificial strings. Contrary to this, one could claim that mathematical linguistics does in fact provide such a theory, given that its subject matter is the study of formal languages, grammars and automata. However, it could not possibly be claimed that these studies directly inform us about the mental representations that the mind constructs when analysing these strings; that is, it could not be claimed that these studies provide an accurate description of a mental reality. Another possibility, which I take to be implicitly assumed by AGL scholars, is that these representations are in fact parasitic on natural language representations; I will explicitly deny this later on, but will, for the time being, suspend judgement on this specific point.

Much like psycholinguistics, though, a number of postulated processing operations ought to be put forward and analysed. Interestingly, the default position seems to be that the underlying grammar is operative in some direct way, even if no reason to believe this has in fact been offered. Given this starting point, it is no big mystery that AGL scholars believe they are actually probing recursion in their studies, not only in the sense that recursive structures are those that are generated by recursive rules, but in the much stronger sense that the recursive rules are, in fact, parsing operations. I will however claim that no AGL study has in fact taken the possibility of recursive processing seriously, mainly because these scholars do not understand what this really amounts to. It is to this that I draw attention now.

3.2 Recursive processes, or the processing of recursive structures

Any description from the computer science literature on the distinction between recursive and iterative processes points to the differences in memory load (e.g. Abelson et al. 1996, p.37 et seq.). That is, 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. Let us take the factorial functions we defined recursively above. Importantly, these can be processed in two distinct ways. The recursive processing (shown on the left handside of the Table below) naturally follows from the recursive definition, while the iterative solution (shown on the right handside) necessitates a subtle observation. This is simply that 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. (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.)

As the shape of these implementations show, the material kept in memory at any stage differs greatly. In the second line of the recursive processing, the actual

Table 1: Recursive and iterative implementations

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

operation in course is *factorial 2*, while what is being kept in memory is *4 × (3 × ...)*. This is in great contrast to any stage of the iterative process, as the only things in working memory are the operation in course and the variables it operates upon. Naturally, an iterative process is in general more efficient; still, there exist clear data structures meriting a recursive solution.

Three properties must be met for a recursive solution to be the most natural: *a*) the original problem must be decomposable into simpler instances of the same problem; *b*) the subproblems must be so simple that they can be solved without further subdivision; and *c*) it must be possible to combine the results of solving these subproblems into a solution to the original problem (Roberts, 2006, p.8). Of course, recursive structures are naturally (and intuitively) the ideal candidates, but this should not distract from the point just made, namely that there is nothing intrinsically recursive about the factorial class. That is, the *suitability* of the recursive solution has to do with the nature of the *solution* itself, and not with the structures themselves. The connection between a structure and a recursive processing is, therefore, an *empirical* matter to be worked out on an *individual* basis; it cannot be simply assumed.

There is a great confusion about this in the cognitive sciences. Thus, much of the literature clearly conflates structures and mechanisms, inevitably concluding that recursive structures can only be generated or produced by recursive mechanisms, and mutatis mutandis for iterative structures and mechanisms. I have already noted above that a general property of implementations is that any sort of task which can be solved recursively can also be solved iteratively, a well-known fact about the study of computable functions. Indeed, at the most general level, any function or task that can computed by the partial recursive functions of Church/Kleene (Kleene, 1952), i.e. a recursor, is computable by a Turing Machine, and the latter *is* an iterator (Moschovakis, 1998). Translating this general result into actual processes is no small matter, but the literature provides many cases. Liu & Stoller (1999), for example, 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, of which Fibonacci sequences are an example (contrary to what

Fitch 2010, p.78 seems to think.).

This is related to Corballis's contention that 'true recursive parsing' would be demonstrated if the parser links elements from the inside out. Given that the pairs of the aforementioned experiment were linked by phonetic features, it is these features that must be kept in memory in order to link the different pairs, but this does not mean that the processing is recursive in any sense. As I mentioned above, the memory load exerted by 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).[8]

A recursive process does indeed result when a given procedure calls itself, but this self-call is *simpliciter*; in the factorial example above, 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, the string $A_1A_2A_3$ does not necessitate that the B elements appear in any particular order for the correct linking of features to take place; this occurs here merely as an artefact of the way the experimenter creates and presents the materials.[9] The resultant memory load therefore follows from this linking of features; 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 so forth. 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 system entirely.

The problem is that Corballis (2007) 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 the structures

[8]de Vries et al. (2008) hint at this point in their discussion of the results reported in Friederici et al. (2006) when they wonder that they 'may have... examined participants' performance on distinguishing sequences based on properties other than their linguistic structure' (p.772).

[9]I am ignoring the fact that crossed dependencies may be generated by context-sensitive grammars, which also employ recursive rules, because the literature I am reviewing has mainly focused on centre-embedding. However, the point I am making carries over, I believe, to these grammars too.

and mechanisms distinction, a tactic that is fairly endemic in the literature (e.g., as in Friederici et al. 2006, p.2458). Non-recursive mechanisms are in fact capable of processing recursive structures such as the self-embedding supposedly exemplified in PSGs; 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.

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 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 an 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.[10] 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 seems to me 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

[10]This general point applies, I believe, to the visual arrays that Jackendoff & Pinker (2005, p.218) discuss.

a recursive solution would be applicable, but this must follow from the three properties a recursive solution must meet (see supra). 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 now briefly discuss.

Recursive processes are characterised by the two operations that implement the chains of deferred sub-operations: "push-down" (when the chain begins and the parser moves down a level) and "pop-up" (when the internal sub-operations have been completed and the parser moves up a level). Necessarily, the instantiation of these operations involves a higher memory load at the specific points where they apply, a load which distinguishes this type of processing from its non-recursive kin. Consequently, an online secondary task such as some sort of monitoring (phoneme, extraneous sound or else) can be employed to compare the different predictions a recursive and non-recursive process yield. This, however, involves an important step that the AGL literature has completely shunned; namely, the postulation of putatively operative processing mechanisms.

Quite clearly, these postulated operations may bear no relation to the *competence* of this specific domain; that is the underlying *grammar* —a point that has been repeated ad nauseam in the linguistics literature. Given that AGL studies are mainly concerned with performance variables (pace Fitch & Hauser 2004), and that we have no idea what processing operations they are studying, we remain in the dark as to how to interpret their results.

3.3 Recursion and hierarchy

Most of this confusion is perhaps the result of misunderstanding the relationship between recursion and hierarchy, of which self-embedding is a subtype. These issues have been treated extensively in the formal sciences, but their implications do not seem to have been fully grasped by AGL scholars.

Take the case of computer science, a discipline which also employs "trees" in order to represent nonlinear data structures. Computer scientist Donald Knuth certainly echoes a widely-held view when he writes that 'any hierarchical classification scheme leads to a tree structure' (Knuth, 1997, p.312), but more importantly, we need to understand his contention that 'recursion is an innate characteristic of tree structures' (idem., p. 308). By "innate", here, is probably meant intrinsic, and we can clarify what this means by providing a graphic representation of the recursive implementation of the factorials, as shown by Fig. (1).

Note that the hierarchical structure directly stems from the fact that the implementation is underlain by a two-equation system: a variable plus a self-call, and it is the latter that expands into the base case, effectively terminating the recursion and the overall computation. It is this specific characteristic that explains why this type of hierarchy, a binary tree, automatically results from a recursive implementation. This hierarchy, however, 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 factorial of 4. This would amount to a definition of a structure in terms of how

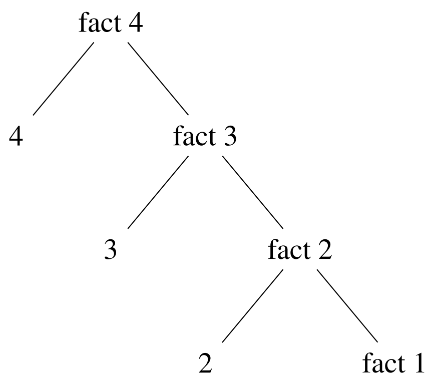

Figure 1: Recursive implementation of the factorials

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.

There is certainly a difference between representing a hierarchy of operations and representing a complex object; the factorials example is meant to illustrate that a recursive implementation automatically results in a binary hierarchy, but that one cannot necessarily infer the former from the latter, which, I think, is the crucial mistake AGL studies are making.

It is also worth pointing out that an implementation is a real-time computational process, and we are therefore on a different level of analysis than when discussing, say, Merge. Granted, linguistic expressions also exhibit a binary tree structure, and it is certainly the case that Merge effects this geometry, but crucially it does not do so in the way of a recursive implementation. Recursive generation (successor function) and recursive implementations are different things, even *if* they may result in similar forms.

This point about recursion and hierarchy is being stressed because it has clear repercussions for actual research, as a look at the discussion in Fitch (2010), I think, demonstrably shows. 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. More importantly for our purposes, Fitch puts forward two problematic claims; firstly, that a recursive rule has the property of self-embedding (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.

Technically speaking, rewriting rules only return strings, not structures, which is presumably one of the reasons why rewritings rules were eliminated from linguistic theory (Cf. Collins 2008, 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 a production system are recursive devices for the same reason, that is, qua generative systems that are underlain by the successor function, but it is true that the replacement of one for the other further involves the postulation of an operation that embeds elements into one another. Merge does this in a bottom-up fashion rather than generating strings in the left-to-right manner of rewriting rules (only the latter would be similar to a recursive implementation, at least in this weak sense), but 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 (i.e., embedding) with how it proceeds (which is how a recursive operation is identified).

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 (p.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. 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 of the resultant structure. More importantly, we are owed an explanation as to why this could tell us anything about recursive generation (competence) or processing (performance), as opposed to self-embedded structures only.

This position is perhaps too widespread for comfort, and has recently even made it into the popular press (see 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.

3.4 Grammars and parsers in the brain

The discussion so far has attempted to show how the role of recursion changes between levels of analysis, and how it easily slips into confusion and unsupported assumptions and results. Given the general slant of this paper, it is only natural that I will argue that Marr's last level, the hardware implementation, must await a more coherent study of the previous two levels. In fact, I will claim, albeit very briefly, that the neuroimaging evidence pertaining to AGL and recursion is confusing and confused; as things stand, moreover, it could not be otherwise.

I will only focus on the two neuroimaging studies mentioned supra (i.e., Friederici et al. 2006; Bahlmann et al. 2008), which claimed that area BA 44/45 is the seat of hierarchical, recursive building. This was supposed to follow from the fact that this area was only activated during the processing of allegedly hierarchical PSG strings. The strings, it will be remembered, in actual fact required a feature-linking parsing operation, and it was this that the aforementioned studies probed, and not the underlying grammar.

Nevertheless, these two studies seem to be an attempt to map the online processing mechanisms onto brain areas, in contrast to what I suppose Fitch & Hauser (2004) would have intended had they undertaken a neuroimaging experiment.[11] In fact, Grodzinsky & Friederici (2006) draw this very distinction when discussing techniques to locate in the brain two separate theoretical constructs: grammatical principles and the actual processing mechanisms. The former approach is based on studying dissociations and impairments in Broca's aphasia and by mapping the loci of activation in fMRI, while the latter aims to map onto brain space and time 'psycholinguistically defined processing components' (p.243). It would then follow that the former approach aims to locate Merge (the seat of recursion in the language faculty), and it is not surprising to find an exclamation mark as to its locus of activation in fMRI (see Table 2, p. 273).

Surely, these neuroimaging studies cannot tell us anything about the core computational mechanism, despite statements to the contrary. At best, they will tell us something about processing, but it is not easy to interpret the results that they do present, given the scant nature of the postulated mechanisms that Friederici et al. (2006) and Bahlmann et al. (2008) were studying. We do not find a great deal of description of any operation in these papers, and not much more is to be found in Grodzinsky & Friederici (2006) either, apart from the three very broad phases they identify: computing local phrase structures, building dependency relations and syntactic integration. It goes without saying that the psycholinguistics literature has proposed myriad principles, operations and theories to account for these phenomena, but these studies do not bear on any of them (at least not directly).

[11]This interpretation is supported by how the two brain imaging studies define their subject matter. Friederici et al. (2006, p.2458) tell us that '[i]t has been argued that the human language faculty is based on the capacity to process recursive structures', while Bahlmann et al. (2008, p.525) focused on the 'ability to deal with hierarchical structures and recursion is a key feature of human language processing'. It goes without saying that I believe both statements to be fundamentally wrong.

What these studies inform us is on the feature-linking operation, and there *is* a hierarchy among its different applications (i.e., the different phonetic features that appear), but this is certainly an artefact of the way the data are presented, which might explain why BA 44/45 is activated.[12]

I have painted a rather gloomy picture of AGL studies, but it is not clear that any other conclusion can in fact be drawn. I believe this is mainly due to the widespread conviction that studying AGL and natural language tap the same capacity and/or processor, a point I address directly in the last section. Therein, I will also add some programmatic remarks for a more coherent study of AGL.

4 A program for AGL studies

One of the underlying points of this long discussion is surprisingly basic; namely, rewriting rules return strings, and not structured representations. Indeed, this is rather clear in the way the different sequences are defined for each grammar. An FSG grammar returns $(AB)^n$ strings while a PSG generates A^nB^n sequences, but there is no reason to believe that there is any internal hierarchy within them. All that they merit is the trite remark that they exhibit regular patterns of *repetitions*, and the evidence so far amassed suggests, if anything, that humans are genuinely sensitive to such repetitions. If there is any hierarchy, this is a property of collections of rewriting rules in the corresponding grammars (or of the parsing operations), but this is clearly a different matter. Unsurprisingly, scholars have introduced “cues” in order to force a self-embedding interpretation of these sequences, but this, as I have discussed above, does not improve matters very much.

The discontinuity between AGL strings and natural language structures is mentioned in passim by Hochmann et al. (2008) when they state that AGL strings can only successively mirror Subject-Verb configurations (that is, linking *As* with *Bs*), but Verb-Object(s) patterns are entirely disregarded, which is not a small matter (p.1033). It is not clear at all that Subject-Verb configurations are in fact being mirrored, given what has been discussed before. Indeed, AGL strings cannot duplicate the hierarchy that exists between subjects and verbs; moreover, subjects and verbs agree, at least in English and Romance languages, in terms of abstract morphosyntactic features, which remain operative even if subjects are “dropped”. Nevertheless, the point about Verb-Object(s) is an important one, though, as centre-embedded sentences such as *The rat the cat the dog bit chased escaped* exhibit a certain internal structure; namely, the object of the verb *bit* is *the cat*, and so on, but such intricacies are beyond the capabilities of artificial strings. Naturally, this is just another way of stating that there is neither self-embedding nor hierarchy in AGL strings.

One must then be careful about extrapolating properties of natural language

[12]Both de Vries et al. (2008, p.772) and Hochmann, Azadpour & Mehler (2008, p.1033) seem to doubt this possibility, but I think they have the computation of hierarchical structures in mind, a different matter.

from demonstrations of the operations that humans manifest over artificially created strings. For example, much has been made of the results of Saffran, Aslin & Newport (1996) with regard to the abilities of 8-month-olds in recovering statistical regularities in artificial strings, an ability that has been argued could explain the segmentation of speech into words in these very terms. As Yang (2004) has shown, however, performance in this sort of task greatly decreases when children are exposed to child-directed, wordlike units, that is, elements that respect the specific principles of how natural language words are structured, including their prosody, which suggests that these principles must be added to the child's repertoire if they are to acquire a language. This is an instance of the great peril of taking the analogy between AGL strings and natural language expressions too far, and I would like to discuss one final example which is very relevant for the subject-matter of this essay, namely the experiments reported in Makuuchi et al. (2009).

Therein, they attempt to separate the core computational faculty of language (manifested in centre-embedded sentences, according to them) from working memory, and, in doing so, they employ sentences from a natural language (i.e., German) under the assumption that certain constructions are direct analogues of FSG and PSG sequences. I will not discuss in any detail the merits of the overall theoretical analysis or the actual results. Instead, I want to discuss a narrower theoretical question, namely the character of the materials they employed. They propose a two-way factorial design to dissociate *structure* (hierarchical or linear) from the working memory load codified in the factor *distance* (long or short). FSG sequences $(AB)^n$ are consequently employed as examples of linear structures that can vary along the distance axis (the distance is the amount of material between the related elements, and it is either 8 or 4 elements), while PSG sequences (A^nB^n) are examples of hierarchical structures that can also vary regarding the distance between the outermost *AB* pairs (also 8 or 4). The relevant question is whether the German sentences they employ can in fact do the job for which they require them.

All the sentences were preceded by the phrase *Peter wusste, dass...* (Peter knew that...), and they provide an example of each condition, namely:[13]

(1) Maria$_1$, die$_2$ Hans, der$_3$ gut aussah$_3$, liebte$_2$, Johann geküsst hatte$_1$.
(long-distance centre-embedding structure)

(2) Maria$_1$, die$_2$ weinte$_2$, Johann geküsst hatte$_1$ und zwar gerstern abend.
(short-distance centre-embedding structure)

(3) Achim$_i$ den großen Mann gestern am späten Abend gesehen hatte$_i$.
(long-distance linear structure)

[13]Indices indicate dependencies. The distance between *Maria* and *hatte*, in the first pair, and between *Achim* and *hatte*, in the second, is either 8 or 4 words. Makuuchi et al. (2009, p.2 of 6) provide the following translations for the four sentences, in the order presented here: *Maria who loved Hans who was good looking kissed Johann*, *Maria who cried kissed Johann and that was yesterday night*, *Achim saw the tall man yesterday late at night* and *Achim saw the tall man at night and that was late*.

(4) Achim$_j$ den großen Mann gesehen hatte$_j$ und zwar am Abend.
(short-distance linear structure)

Unfortunately, these materials do not display the properties that these experimenters need them to have. They are postulating these pairs to be qualitatively different in a very strong sense; namely, that one pair is hierarchical while the other is not (in other words, that one is a PSG sequence while the other is a FSG string). I will ignore the latter distinction for now, but I would like to point out that all four sentences are in fact *hierarchical* in nature. Granted, they are only focusing on Subject-Verb configurations, and we could accept this is the only hierarchy they are studying, but if so, then their attempt to work out whether non-syntactic verbal working memory can be dissociated from syntactic computation is doomed to fail. In order to show this, let us focus on the first sentence of the last pair, the supposedly "linear" structures. Note that the hierarchy between the subject (*Achim*) and the verb (*hatte*, the auxiliary of "to have") is not the only one that these sentences manifest, as they also contain a number of phrases (so-called complements) that modify the verbal phrase *gesehen hatte* ("to have seen") in various ways. Thus, the direct object *den großen Mann* ("the tall man") answers the question of *what* was seen by Achim, but note that it contains the adjectival phrase *großen* ("tall") within, which modifies the noun *Mann*. Further, the adverbial phrase *gestern* ("yesterday") also modifies the verb, as it tells us *when* this tall man was seen, which is in turn further modified by another adverbial phrase *am späten Abend* ("late at night"). Therefore, the parser must be sensitive to a number of hierarchies: the first is between the subject and the verb with its complements; the second is between the verbal phrase and the two complex phrases that modify it, the noun phrase (the direct object) and the adverbial phrase; and finally, these two complex phrases are composed of smaller phrases that modify their respective heads.

It is therefore hard to see how these data could be employed to segregate syntactic computation and non-syntactic working memory if the *linear* (i.e., non-hierarchical) condition is in fact not part of the experiment. Further, there is no indication that their methodology brings out the linear-hierarchical distinction, or that subjects are in fact analysing these sentences in these terms. This mistake stems, I believe, from a very unhealthy connection between AGL strings and their natural language counterparts.[14]

This discontinuity between AGL strings and natural language expressions just shows a basic property of the linguistic system, namely that it manifests a much more general type of recursive structure than has customarily been identified. At the appropriate level of abstraction, a structure that contains an instance of itself (i.e., an X within an X) appears to be a feature of *any* type of syntactic structure. That is, every syntactic phrase (NPs, VPs, etc.) accords to the same geometry,

[14]I am of course simplifying the analysis of the "linear" sentence I have focused on, but I do not think that a more thorough analysis of this sentence or the others is needed to support the point I am making. Regarding the actual results they provide and their discussion, a reanalysis seems in order, but I cannot undertake that here.

an asymmetric structure of the following kind: [Specifier [Head - Complements]] (Moro, 2008, p.68), as shown in Fig.(2).

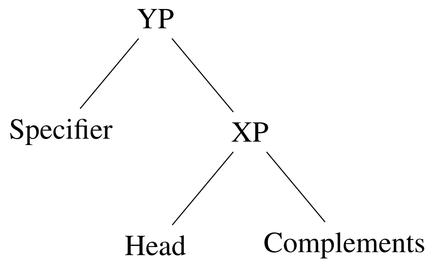

Figure 2: Asymmetric S-H-C structure

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, p. 205 et seq.) argues, all human languages appear to follow this scheme, despite some variation in the linear order. In other words, linear order is not the key property; rather, the central point is the basic hierarchical configuration: S is *always* more prominent than [H-C] and H is *always* more prominent than C.

At this level, 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: endocentric and asymmetric X structures. There is no indication that AGL strings even approximate such a fundamental result of linguistic theory, let alone the many other features the linguistic system shows (movement, locality principles, binding, and many other more).

There are therefore no reasons to believe that AGL can make any use of linguistics for a theory of the structural descriptions that must be encoded and decoded. Further, it cannot be decided a priori that the cognitive abilities manifested in the many AGL experiments so far carried out (see a short compilation in Hochmann et al. 2008) delimit specific capacities and structures of the human mind, let alone the language faculty, until a thorough investigation is carried out —an investigation at the level of what Marr called the *computational level*. Given that subjects seem to be sensitive to the regular patterns of artificial strings, it would not be a surprise to find out that the language faculty is employed somehow, but, if so, this is probably in conjunction with other cognitive capacities; after all, the grammar underlying natural language generates structures, not strings. Chomsky (1980, p.273), in fact, offers a similar account when discussing "languages" that might be learnable through the exercise of other faculties, but it is not clear that the other two faculties he mentions elsewhere in that book (common-sense understanding and science-forming) would be the relevant ones in this case.

It is therefore impossible to evaluate what the huge AGL literature tells us about human cognition overall as long as the field continues to neglect undertaking this enterprise, a state of affairs that is sadly widespread in the cognitive sciences, with its incessant focus on processing, simulation, or brain imaging. In the end, I am calling for an exercise in what might be termed “theoretical psychology”, and it is here consequently proposed that AGL must settle on: *a*) the nature of the actual domain under study, in order to *b*) figure out the underlying properties of this domain so as to *c*) propose models of processing that successfully reconstruct the right structural descriptions.

References

Abelson, H., Sussman, G. J. & Sussman, J. (1996). *Structure and interpretation of computer programs*. Cambridge, MA.: The MIT Press.

Bahlmann, J., Schubotz, R. I. & Friederici, A. D. (2008). Hierarchical artificial grammar processing engages Broca's area. *NeuroImage*, *42*, 525-534.

Bar-Hillel, Y. (1953). On recursive definitions in empirical science. In *Proceedings of the 11th International Congress of Philosophy* (p. 160-5).

Boolos, G. (1971). The iterative conception of set. *The Journal of Philosophy*, *68*(8), 215-231.

Chomsky, N. (1956). Three models for the description of language. In *IRE transactions of information theory IT-2* (p. 113-124).

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. (1980). *Rules and representations*. New York, New York: Columbia University Press.

Chomsky, N. (1995). *The minimalist program*. Cambridge, MA.: The MIT Press.

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.

Coleman, J., Kochanski, G., Rosner, B. & Grabe, E. (2004). *Letter comment on Fitch and Hauser 2004* (Vol. 303). http://kochanski.org/gpk/papers/2004/FitchHauser/FitchHauserScienceLetter.pdf.

Collins, J. (2008). *Chomsky: A guide for the perplexed*. London, UK: Continuum International Publishing Group Ltd.

Corballis, M. (2007). Recursion, language and starlings. *Cognitive Science*, *31*(4), 697-704.

Cutland, N. (1980). *Computability: an introduction to recursion function theory*. Cambridge, England: Cambridge University Press.

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.

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 a nonhuman primates. *Science*, *303*, 377-380.

Fodor, J. A. (1983). *The modularity of mind*. Cambridge, MA.: Bradford Books/The MIT Press.

Fodor, J. A., Bever, T. G. & Garrett, M. F. (1974). *The psychology of language*. London, England: McGraw-Hill.

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.

Gentner, T., Fenn, K. M., Margoliash, D. & Nusbaum, H. C. (2006). Recursive syntactic pattern learning by songbirds. *Nature*, *440*, 1204-1207.

Grodzinsky, Y. & Friederici, A. D. (2006). Neuroimaging of syntax and syntactic processing. *Current opinion in neurobiology*, *16*, 240-246.

Hauser, M. D. (2009). Origin of the mind. *Scientific American*, *301*(3), 44-51.

Hochmann, J.-R., Azadpour, M. & Mehler, J. (2008). Do humans really learn anbn artificial grammars from exemplars? *Cognitive Science*, *32*, 1021-1036.

Jackendoff, R. & Pinker, S. (2005). The nature of the language faculty and its implications for evolution of languages. *Cognition*, *97*, 211-225.

Kinsella, A. R. (2009). *Language evolution and syntactic theory*. Cambridge, England: Cambridge University Press.

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.

Liu, Y. A. & Stoller, S. D. (1999). From recursion and iteration: what are the optimizations? *SIGPLAN Not.*, *34*(11), 73-82.

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).

Makuuchi, M., Bahlmann, J., Anwander, A. & Friederici, A. D. (2009). Segregating the core computational faculty of human language from working memory. *Proceedings of the National Academy of the Sciences of the USA*, *106*, 8362-8367.

Marcus, G. F. (2006). Startling starlings. *Nature*, *440*, 1117-1118.

Marr, D. (1982). *Vision: A computational investigation into the human representation and processing of visual information*. San Francisco: W. H. Freeman & Company.

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.

Miller, G. A., Galanter, E. & Pribram, K. H. (1960). *Plans and the structure of behaviour*. New York, New York: Holt, Rinehart and Winston, Inc.

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 algorightms and their implementations. In S. B. Cooper, B. Lowe & A. Sorbi (Eds.), *New computational paradigms* (p. 81-118). Springer.

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.

Poletiek, F. H. (2002). Implicit learning of a recursive rule in an artificial grammar. *Acta Psychologica*, *111*, 323-335.

Pothos, E. (2007). Theories of artificial grammar learning. *Psychological Bulletin*, *133*(2), 227-244.

Rice, G. (1965). Recursion and iteration. *Communications of the ACM*, *8*(2), 114-115.

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.

Saffran, J. R., Aslin, R. N. & Newport, E. L. (1996). Statistical learning by 8-month-old infants. *Science*, *274*(5294), 1926-1928.

Simon, H. (1962). The architecture of complexity. *Proceedings of the American Philosophical Society*, *106*(4), 467-82.

Soare, R. (1996). Computability and recursion. *The Bulletin of Symbolic Logic*, *2*(3), 284-321.

Soschen, A. (2008). On the nature of syntax. *Biolinguistics*, *2*, 196-224.

Tomalin, M. (2007). Reconsidering recursion in syntactic theory. *Lingua*, *117*, 1784-1800.

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.

Yang, C. (2004). Universal grammar, statistics or both? *Trends in Cognitive Science*, *8*(10), 451-456.