★ REVIEWS ★

"A Running Back" and Forth: A Review of *Recursion and Human Language*

Harry van der Hulst (ed.). 2010. *Recursion and Human Language.* (Studies in Generative Grammar 104) Berlin: De Gruyter Mouton.

by David J. Lobina

An attentive reader of the cognitive science literature would have noticed that the term *recursion* has appeared in myriad publications, and in many guises, in the last 50 or so years. However, it seems to have gained a disproportionate amount of attention ever since Hauser *et al.* (2002) hypothesized (for that is what it was) that this property may be the central and unique feature of the faculty of language.

Indeed, a barrage of publications, conferences, and even critical notes in the popular press about recursion has recently flooded academia. The volume under review here is the result of one such conference — one that was celebrated in 2007 at the Illinois State University — and it offers, or so it says, a compendium of works that tackle this notion from different perspectives.

I will not be following the thematic division underlying this volume as a way to frame the "different perspectives" it advertizes. Rather, this critical note will focus on four distinct senses of the term recursion that can appropriately be applied, or so it will be argued here, to four well-defined theoretical constructs of the cognitive sciences. The formal sciences will, naturally, inform most of this discussion, but the focus of this note will fall on relating the different perspectives of this collection to the four senses of recursion I will outline. Ultimately, this review will press one main point: Contrary to a(n apparently) widespread belief, neatly stated in this book's back cover, it is simply not true that recursive structures in languages "suggest recursive mechanisms in the grammar" (at least not in the sense that is usually intended in the literature — see *infra*).[1]

The one feature that binds together the four theoretical constructs I will be focusing on is the self-reference property that characterizes recursion — a feature that is quite unrelated to the uses to which this notion can be applied. This self-

My gratitude to Noam Chomsky for comments on an earlier version of this paper. This work was partly funded by grants SEJ 2006-11955, 2009 SGR 401, and 2010 FI_B2 00013.

[1] I will refer to individual chapters by writing the last names of their authors in bold. Thus, the first chapter would be referenced as **Sakel & Stapert**. The editor's introduction to the volume will be the only exception, as I will refer to it as, plainly, the **Introduction**. When addressing a particular chapter in more detail, the first-mention includes the author's first names; two chapters did not make it into this review essay for sake of coherence, the one by Hans-Jörg **Tiede &** Lawrence Neff **Stout** and the one by Simon D. **Levy**.

ISSN 1450–3417

reference property is readily demonstrated by the first connotation I will be considering (the primary meaning in mathematics, in fact), which consists in "defining a function by specifying each of its values in terms of previously defined values" (Cutland 1980: 32); that is, a definition by induction (or recursive definition). The factorial functions (*n!*) offer a standard and rather trivial example:

(1) *Def. n!:*

a. if n = 1 n! = 1 *base case*
b. if n > 1 n! = n x (n-1)! *recursive step*

Note that the recursive step involves another invocation of the factorial function. Thus, in order to define the factorial of, say, *4* (i.e. 4 x 3!), the function must define the factorial of *3*, and so on until it reaches the factorial of *1*, the base case, effectively terminating the recursion.

A definition by induction ought not to be confused, however, with a related construct that receives similar denominations, such as an 'inductive definition', 'inductive proof', or 'mathematical induction'.

An inductive definition, a mathematical technique employed to prove if a given property applies to an infinite set, proceeds as follows: First, we show that a given statement is true for *1*; then, we assume it is true for *n*, a fixed number (the inductive hypothesis); lastly, we show that the statement is therefore true for *n+1* (the inductive step). If every step is followed correctly, we conclude that the statement is true for all numbers (Epstein & Carnielli 2008). An inductive definition, then, also employs recursion, but it additionally includes an inductive hypothesis. These two constructs should not be conflated, even if they are closely related. In fact, 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, the recursive definition of a function is justified insofar as it ranges over the domain established by the inductive definition (Kleene 1952: 260 *et seq.*).[2]

Recursive and inductive definitions are discussed a few times in this collection; this is the case, to a certain extent, for **Introduction**, **Langendoen**, **Kinsella**, and, most notably, in Geoffrey **Pullum &** Barbara C. **Scholz**. The latter 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 framed, I believe, alongside three main themes and it is of interest to have a closer look at them; these are: a critique of the actual 'standard' argument supporting infinitude claims, the accompanying assumptions, and a number of loosely related *obiter dicta* that I will not discuss in any detail.

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*, etc. (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

[2] The mathematical literature contains many different types of recursive functions: the primitive class, the general class, and the partial class, among others. They are all recursively defined functions, but range over different types of objects; whatever objects/relations they subsume should not distract from the fact that recursion remains a central property.

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*). It is not clear 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** have in mind. Indeed, the examples they do provide are rather strained; specifically, the connection they make between mathematical induction and a remark by Pinker in the context of a popular science book (see p. 119) seems rather weak.

More interestingly, later on in their paper (p. 124 *et seq.*), they point to the (supposedly widely-held) assumption that languages are collections — in a strong mathematical sense.[3] Given that they take this to be the case, it is no surprise the burden they place upon linguists to prove the infinitude claim. In a similar vein, Terence **Langendoen**'s contribution orbits these very issues, and is ended 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. 145).

Note that the whole issue, then, turns out to revolve around 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. This is, however, unwarranted. It is certainly true that many linguists have employed mathematical techniques and vocabulary to study natural languages, but these were so used because they were useful — certainly not to reduce linguistic phenomena to abstraction. In fact, the latter play a rather limited role in linguistic explanation, for note that linguists typically focus on informants' grammatical judgements in order to unearth the underlying structure of strings. This kind of study focuses 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 toolkit for a call to reduce linguistics to mathematics, but no such thing ought to be accepted by the working linguist (I will retake the infinitude claim below).

On another note, one would expect to see the opposite argument (that is, the finiteness of a given language) to be placed under the same burden, but this

[3] They rank this assumption as one of four factors that may account for the persistent presence of infinitude claims in the literature, but in fact only provide three (pp. 124–129). The second of these — the connection between recursion and linguistic creativity — rests on an obvious misrepresentation, corrected many times before (see Chomsky 2006: xviv). Roughly, creativity does not rest on the ability to construct new sentences (p. 126); rather, this property points to the fact that linguistic behaviour is generally stimulus-free, and that speakers/hearers have the capacity to understand/produce novel sentences that are appropriate to context.

does not appear to be the case for languages that *prima facie* lack the 'grammatically-preserving extensibility' syntactic facts mentioned in (i) (p. 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 (such as self-embedding and coordination), which leads 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 & Scholz**'s (and **Langendoen**'s) burden.

Be that as it may, I now want to argue that none of this has, in actual fact, much to do with the introduction of recursion into linguistics — at least not in the sense that Chomsky has treated this notion.

A second sense of the term recursion has it 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 formalize 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 $\phi_1...\phi_n$ 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 $\phi_1...\phi_n$, then it also generates the structure ϕ_{n+1}" (Chomsky & Miller 1963: 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 (see *infra*).

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 characterization 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).[4]

This is better understood in the context of the discussion Soare (1996) provides on the state of the art within mathematical logic. Therein, he argues that the field has for long assumed that recursion and computation are synonymous terms (and the same would apply for recursive and computable). This, he argues,

[4] Particularly, it is in this context that Soschen's (2008: 199) statement regarding how "singleton sets are indispensable for recursion" should be understood.

has resulted in what he calls the Recursion Convention (RC), a state of affairs he has attempted to reverse in subsequent publications. The RC has three parts: (i) Use the terms of the general recursive formalism to describe the results of the subject, even if the proofs are based on the formalism of Turing computability; (ii) use the term Church Thesis to denote various theses; and (iii) name the subject using the language of recursion (e.g., Recursion Function Theory).

Granted, even if it is commonly conceded that a Turing Machine captures the manner in which every conceivable mechanical device computes a calculable function (and it is, furthermore, generally accepted as the best formalization in the field), Turing's model did not in actual fact provide a formalization of what an algorithm qua formal object is. Indeed, there is a distinction to be had between formalising an algorithm qua a 'model of computation' — that is, an analysis of what actually happens during a computational process — and qua an abstract mathematical object. It is to the former construct that Turing's model appropriately applies, while it is the latter that systems of recursive equations and recursors subsume. Furthermore, it is a well-known result that Turing Machines and the partial recursive functions formalism of Church/Kleene (but not the general recursive class) are extensionally equivalent in the sense that both identify the class of computable functions. From the same inputs, both formalisms return the same outputs, albeit in different ways (these 'intensional differences' will be of some importance later on, though). Finally, it is no surprise that the Turing Machine model has been more prominent in cognitive psychology, with its emphasis on real-time processes, while the more abstract characterization of what a computation is — one based on recursion — has found its natural place in theoretical linguistics mainly. Indeed, as Collins (2008) states in the context of an introductory book to Chomsky's thought: "via the Church/Turing thesis, computation is just recursion defined over a finite set of primitive functions" (p. 49).

There is, therefore, a certain consistency in Chomsky's writings if we understand his treatment of recursion in the terms just described. Perhaps rather tellingly, he *has* pointed to the connection between grammatical theory and recursive function theory in many writings (e.g., in Piattelli-Palmarini 1980: 101), which suggests that he may have been influenced by the RC.[5]

Naturally, the whole point of introducing recursion into linguistics was to account for the fact that speakers/hearers show a continuous novelty in linguistic behaviour — a novelty that does not appear to be capped in any meaningful respect. Further, since speakers/hearers cannot possibly store all the possible sentences they understand or utter, the cognitive state accounting for this linguistic behaviour must be underlain by a finite mechanical procedure — an algorithm. This is one of those properties that one would argue are a matter of 'conceptual necessity'. A rather trivial matter, perhaps, but the whole point has been muddied by orbiting issues. Just as **Pullum & Scholz** do, many studies focus on the so-called self-embedded sentences (sentences inside other sentences,

[5] This is actually confirmed in personal correspondence with Noam Chomsky (May 2009): "[T]here is a technical definition of 'recursion' in terms of Church's thesis (Turing machines, lambda calculus, Post's theory, Kleene's theory, etc.)", the only one used he's ever used, "a formalization of the notion algorithm/mechanical procedure". Further, he states that he's "always tacitly followed the RC".

such as *I know that I know* etc.) as a way to demonstrate the non-finiteness of language, and given that self-embedding is sometimes used as a synonym for recursive structures (see *infra*), too close a connection is usually drawn between the presence of these syntactic facts and the underlying algorithm of the language faculty. However, even if there were a language that did not exhibit self-embedding but allowed for conjunction, you could run the same sort of argument and the non-finiteness conclusion would still be licensed. These two aspects must be kept separate; one focuses on the sort of expressions that languages manifest (or not), while the other is a point about the algorithm that generates *all* natural language structures.

There is surprisingly little in the collection under review here that touches on these very issues. Both Harry van der Hult's **Introduction** and Arie **Verhagen** do discuss global and local applications of recursion, but not quite in the terms outlined here. **Verhagen** describes two roles for recursion, a specific one that gives rise to long-distance dependencies (supposedly the self-embedded sentences, see below) and a more general one that delineates a mechanism for embedding phrases inside other phrases. As for the **Introduction**, and even though the discussion presented there is framed in terms of rewriting rules, the main points pertain to structures only. Thus, the general application refers to the general embedding of phrases into other phrases, while the specific one refers to those phrases that are embedded within a constituent of the same kind. I will come back to this below, but it is worth pointing out now that this is the closest this collection gets to discussing the central role of recursion in the formalization of an algorithm — a clear shortcoming, given that the RC, systems of recursive equations, recursors, etc form the core of Chomsky's thought on the matter.

There is some tangential discussion regarding Merge by Jan **Koster**. He takes issue with the postulation of a recursive Merge as the syntactic engine that generates linguistic expression. His worries seem to be twofold; on the one hand, a recursive Merge cannot do anything unless in combination "with external, invented cultural objects — lexical items" (p. 289); on the other hand, these lexical items come with specific combinatorial properties that already account for the hierarchical structure that Merge will, then, redundantly generate "once more" (p. 292). The latter is, of course, a valid point that recapitulates the debate between representational and derivational views on theories of grammar. The former is, however, more troubling. Even if we were to grant **Koster** that lexical items *are* cultural inventions, we would do well to remind ourselves that there cannot be any cultural inventions that are not entertained in the mind first, which is perhaps a trivial point. Moreover, we can be sure that external, cultural inventions do not come with "fully-fledged combinatorial properties" (*idem.*) other than the mould that the linguistic capacity imposes — and this is clearly an internalist explanation. More importantly, this bears very little relation to my description of the role of recursion in linguistic theory as a general property of the underlying mechanical procedure.

I have been defending a common thread running through Chomsky's writings, but this is not to say that he has always been consistent — or that the focus has not fallen, on many occasions, on the internal applications of recursion within production systems.

The characterization of this internal application has changed dramatically as the theory progressed. An early characterization of generative grammar divided the computational system 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 some phrase markers into other phrase markers, preserving structure). In Chomsky (1957), the recursive property of certain rules is ascribed to the latter system, while Chomsky (1965) assigns it to the base component. By the 1970s and 1980s, most of the rewriting rules were in fact eliminated from syntactic theory, perhaps completely so by the time Chomsky (1986) appeared. The latter is an important point, given that most discussions on recursive mechanisms — and this is no exception in this collection — seems to be centered exclusively on rewriting rules, which is rather unfortunate. It should be trivial at this point to remark that recursion as a general property of generative systems remains at the center of linguistic theory regardless of the replacement of production systems with Merge — both, as I have tried to show, are underlain by the successor function.

Nevertheless, it is of interest to discuss recursive rewriting rules to some extent, given the prominence they receive in the literature — and in this collection. Consider the following sample below.

(2) a. S → NP VP
b. NP → D N
c. VP → V NP
d. NP → N (NP)
e. VP → V S

Rules (2d) and (2e) are recursive, as the category on the left-hand side of the arrow is reintroduced on the right-hand side (directly in (2d), indirectly in (2e)). These can generate what I for now will call nested constructions (sometimes called self-embedding), such as noun phrases inside other noun phrases (such as *John's [brother's [teacher's book]] is on the table*), or sentences inside other sentences (as in *John thinks (that) [Michael killed the policeman])*. In general, it is this sort of structures that most linguists have in mind when they talk about recursion.

It is sometimes supposed that nested structures and recursive rules are very closely connected, to the point that nested constructions cannot be generated by anything other than recursive rules. This is mentioned in this collection a couple of times, sometimes with (dubious) references to the mathematical linguistics literature on formal grammars. This is clearly not quite correct, however. To a first approximation, it is worth noting that recursive rules were introduced in order to simplify the grammar. Take a nested string such as [a[ab]b], where the *a's* stand for subjects and the *b's* for verbs. This can easily be generated by the employment of rules like (2d) and (2e), while a more complicated generation would involve the repeated application of rules like A→aB, B→aC, C→bD, D→b. It is precisely in this context that Chomsky (1956: 115–116) states that "if a grammar has no recursive steps […] it will be prohibitely complex", with danger of reducing it to a list of sentences.

Amy **Perfors *et al.*** (Josh Tenenbaum, Edward Gibson, and Terry Regie)

provide (partial) confirmation for this intuition by employing a qualitative Bayesian analysis to calculate the ideal trade-off between simplicity of grammars (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 formalization 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 the observed linguistic behaviour (this point about corpora resurfaces in many other contributions). Secondly, it is an obvious point to make that this analysis only applies to those theories that postulate production systems as grammars — those linguists close to the generative framework, though, have long dispensed with them.

Quite clearly, none of this applies to theories that focus on Merge as the central syntactic engine. Moreover, it certainly has very little to do with the general point made *supra*; namely, Chomsky's leitmotif is based on recursion qua general property of the computational system underlying language, be this a production system or a set-operator like Merge.

Despite it all, it is worth delving into the uses (and abuses) that systems of rewriting rules have been put to as to unearth some (seemingly) widespread mistakes. This will allow me to introduce the third sense I would like to discuss, one that pertains to the study of computational processes, which is the interest of much of applied computer science. At this point, though, it seems "a reasonable conjecture" to claim that at root "there is only one fixed computational procedure that underlies all languages" (Chomsky 1995: 11); a 'recursive' Merge in this sense.

It is important to note that there is a significant discontinuity between rewriting rules and linguistic expressions. 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: 58)[6]. It is a point that deserves emphasis, as its neglect hampers clarity. Take Fitch (2010), for instance; therein, he 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

[6] I say "technically" in reference to the historical fact that rewriting rules have always been employed as string substitution operations. It is sometimes stated, however, that a system of rewriting rules *strongly generates* a set of structures, while it *weakly generates* a set of strings, but there is no obvious difference in the actual rules to merit the distinction — apart from the definition. Perhaps this should be rephrased as follows: A computational system such as rewriting rules generates weakly but a system like Merge generates strongly.

structure (p. 80), which I suppose carries over to simply embedding rules and embedded structures.

The first claim is simply not correct. A rule is recursive if there is a self-call, but this is independent of *what* operation is in fact executed. There is a distinction to be had between what a rule does and how it actually proceeds, and it is to the latter than recursion applies. It is this reflexive property that makes the definition of the factorial functions recursive, but there is no sense in stating that there is any embedding whatsoever.

As mentioned, rewriting rules return strings, not structures; *a fortiori*, there is no such thing as a self-embedding rewriting rule. Moreover, Fitch misplaces the long-held stipulation he identifies. In previous models, the rules of the base component would return simple declarative sentences, and these would be converted into more complex structures by the transformational component; the latter were not part of the set of rewriting rules.

The replacement of Merge for production systems involved 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, but 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.

The conflation, *apropos* recursion, between what an operation does and how it proceeds is rather common in the literature, and this collection of papers is no different. Some contributions (**Karlsson**, **Verhagen**, **Kinsella**, **Harder**, **Hunyadi**) discuss what they call center-embedding rules, tail-recursive rules, the sort of structures these generate, and their relationship. Much like Fitch, these terms actually refer to the structures themselves, rather than the actual rules. Thus, a center-embedding rule is supposed to generate nested structures in which, say, a sentence is embedded in the middle of a bigger sentence, like in the classic (*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 terms, however, have absolutely nothing to do with the recursive character of the rules themselves, only to the type of embedding the resultant expression manifests. A center-embedding rule, after all, is not one in which the reflexive call occurs in the middle of a derivation, but even if it did, this has no substantial consequences. As for tail-recursion, this is a widely-used term in computer science, and it 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.

Rather surprisingly, some of the aforementioned chapters seem to have a much stronger point in mind. Fred **Karlsson**, following Parker (2006; cited therein), states that 'nested recursion' rules (i.e. center-embedding; **Verhagen**, p. 103 tells us that this is sometimes known as 'true recursion', but no reference is

provided) cannot be reduced to iterations (while tail-recursion can)[7], a claim that is repeated by Peter **Harder** (p. 239) and, with qualifications, by Vitor **Zimmerer &** Rosemary A. **Varley** (p. 397).

They could not possibly mean this as a general point about computability theory, however. After all, 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: 114). In fact, one of the references mentioned in this collection, 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 relations, such as multiple base cases or multiple recursive steps, of which Fibonacci sequences are an example (contrary to what Fitch 2010: 78 seems to think).

Perhaps what these authors have in mind is a much narrower point; namely, the interrelations between recursion and iteration *within* sets of rewriting rules. In this context, James **Rogers &** Marc **Hauser** offer a solid discussion of formal grammars and their potential relevance for the study of behaviour. Still, the formal literature hardly contains a mention of 'center-embedding recursion', a term that only seems to appear in some linguistic papers; as I stated above, it tends to appear in the context of recursive rewriting rules, even if in reality it refers to either an embedding operation of a particular kind, or to a certain type of structure.

As for the recursion/iteration equivalence in general terms, let us take the factorial functions we defined recursively above to clarify this point, which brings me to the third sense I would like to focus on. This 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, which is usefully contrasted with an iterative process, wherein an operation reapplies in succession (Abelson *et al*. 1996: 33–34).[8]

The recursive processing (shown on the left-hand side of Table 1) naturally follows from the recursive definition, while the iterative solution (shown on the right-hand side) 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*. (NB: The first digit of the iterative solution shows the factorial whose number we are calculating, the second

[7] Further, he incorrectly states, by misunderstanding the discussion in Tomalin (2006: 64), that Bar-Hillel might have reintroduced recursion into linguistics. Rather, Bar-Hillel seems to have been interested in a more precise definitional technique for theoretical constructs. Chomsky (1955: 45) manifests his agreement in spirit, while two years later sees "success along these lines unlikely" (Chomsky 1957: 58).

[8] This is usefully contrasted with the 'clear' definitions of recursion and iteration that **Kinsella** offers on page 182. Note that what she actually provides is a clear example of the conflation we discussed *supra*; namely, between embedding (or not) and recursion (otherwise, iteration).

digit is the actual counter and the third is the running product.)

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	

Table 1: Recursive and Iterative Implementations

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 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: (i) the original problem must be decomposable into simpler instances of the same problem; (ii) the sub-problems must be so simple that they can be solved without further subdivision; and (iii) it must be possible to combine the results of solving these sub-problems into a solution to the original problem (Roberts 2006: 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. Indeed, at the most general level, any function or task that can computed by the partial recursive functions of Church/Kleene (Kleene 1952), that is, 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 (see Liu & Stoller 1999, mentioned *supra*).

There is not an awful lot of discussion on real-time recursive processes in the collection under review here. **Perfors *et al***. mentioned, in passing, that even though syntax may well be fundamentally recursive (in the sense of the grammar containing recursive rewriting rules), the parser could "usefully employ non-

recursive rules" for simpler sentences (p. 170) — a well-taken point about the efficiency of non-recursive rules in processing.

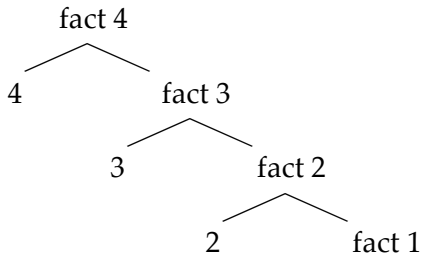
László **Hunyadi** does offer some data regarding possible recursive performance. After correctly stating that recursion and iteration would access (working) memory differently (p. 347), some experimental evidence is provided on the type of prosodic structure associated with some of the structures alluded to earlier (self-embedding and tail-recursion; an 'iterative' structure is also delineated, *viz. John is an excellent, cheerful, good-humoured man*). The experiments were rather low-key; subjects were to read some sentences aloud, so that tonal phrases could then be analysed. They found different pitch levels for the embedded sentences in the nested constructions, a phenomenon that was coupled with a 'tonal continuity' — that is, there is a long-distance dependency between two discontinuous segments. For example, for *[A…[C]…B]*, the tonal properties of *[A…B]* are identical to a continuous *[AB]* phrase. Further, this phenomenon is accompanied by three other effects: (i) there is no lowering of the pitch contours in *C* (a general tendency called 'downdrift'), (ii) the phrase *C* is realized in a different pitch, and (iii) there is an 'upstep' of *B* to the initial pitch of *A*.

Hunyadi sees this process as a clear example of recursion, given that the tonal properties of *A* must be kept in memory during *C*, so that they can be restored at *B*. This, **Hunyadi** believes, is a direct probe of memory — a 'bookmark effect'. Further analyses with tail-recursive and iterative structures show that the bookmark effect does not appear, which would suggest a principled distinction between these and self-embedded structures.

Despite couching the whole discussion in terms of the computational principles of recursion, tail-recursion and iteration, these results appropriately describe structural properties of prosody (or grouping in general; see pp. 361–365), but not its actual production (let alone the underlying grammar). That is, the different memory loads that recursive and iterative *processes* would incur was, in actual fact, not probed at all — there is a distinction, after all, between probing structures and probing mechanisms. We can doubtless be certain that the prosodic structure is, roughly, isomorphic to the syntactic structure, but not much follows about the underlying processing mechanisms.

Furthermore, this contribution introduces some confusion regarding the relationship between recursion and hierarchy (and iteration), and it might be worth our time to clarify it. Take 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: 312); 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 here:

(3)



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 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.

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'.

It is to the structural 'forms' the language faculty generates that we move onto now, the fourth and last sense of recursion. 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'; 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.

Note that this is a definition that focuses on properties of structures only, independently of the operations/mechanisms that generate/process them. There is nothing odd about this, Chomsky & Miller (1963) defined certain constructions in these very terms; they defined the tail-recursive sentences as either right- or left-recursive (depending on the direction of the embedding), and offered the term self-embedding (still used today) for what some call the center-embedding constructions (p. 290).

It is to actual structures and their properties that a significant amount of

papers in this collection focus on. One set of papers focuses on languages that, *prima facie*, lack self-embedding sentences; I will focus on these first.

Jeanette **Sakel &** Eugenie **Stapert**, then, review the data presented by Daniel Everett on Pirahã, an Amazonian language claimed to lack any type of self-embedding. There are two main points here; one is that Pirahã lacks 'mental state' verbs (verbs like *think*, *believe*, etc); *a fortiori*, there is no outright clausal embedding, but simple juxtaposition of individual sentences. The correctness of the latter rests on the status of the verbal suffix *–sai*, a marker that Everett, in earlier work he now considers mistaken, classified either as a nominalizer or as a clausal embedding indicator (see p. 5). Ultimately, **Sakel & Stapert** support Everett's contemporary analysis of this suffix as a single marker of semantic cohesion between parts of discourse. Sauerland (2010), however, 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 would appear, according to Everett's earlier study, Sauerland found that the pitch level in the nominalizer condition was indeed much greater than in the clausal condition, indicating that there *are* two versions of this marker, one of which marks embedding.

Sauerland's methodology is an interesting one, and can complement more traditional ways of determining whether a language exhibits self-embedding. Marianne **Mithun** lists some of the usual formal features that languages with self-embedding manifest (*viz.* complementizers, omission of co-referential arguments, non-finite verb forms; p. 23) and provides an analysis of a variety of languages, such as central Alaskan Yup'ik, Mongolian Khalkha, and North American Mohawak. Apparently, these three languages exhibit some kind of self-embedding, but in different ways, which suggests, to **Mithun** at least, that this feature manifests cross-linguistic gradience (sic.) and variation (p. 39). Perhaps this variation is present within individual languages too. Ljiljana **Progovac** seems to suggest this much for English, given the impossibility of nesting root small clauses (e.g., *Me first*, *Case closed*; p. 193) into other root small clauses. It is to be supposed that even if this is correct, it is so for a small part of the grammar, leaving other claims (*viz.* the presence of self-embedding else-where) virtually uncontested.

Karlsson, on the other hand, offers a typology of recursive and iterative structures (two and six types, respectively; see pp. 43–49 for definitions and examples) based on a quantitative analysis of spoken corpora. By recursive structures he means self-embedding and tail-recursion, and the central claim of his paper is that this sort of corpora analysis provides qualitative data — in the form of 'constraints' — that explain why recursive structures are so rare in spoken language. **Karlsson** then concludes that multiple nesting is an artificial feature of language that "arose with the advent of written language" (p. 64) — it is not a central feature of language.

A similar rationale informs Ritva **Laury &** Tsuyoshi **Ono**. Therein, they provide a corpora analysis of conversations conducted in Finnish and Japanese, reaching similar results (and conclusion): Nested constructions are not very common in spoken Finnish and Japanese (pp. 84–55); therefore, recursive structures cannot be a central property of language (I will come back to this presently).

Another set of papers, on the other hand, focus on self-embedding outside

syntax. Thus, Eva **Juarros-Daussà** argues that there is a restriction (what she calls the two-argument restriction) that prevents argument structure (i.e. the predicate with its lexically encoded arguments) to be truly recursive. Quite clearly, however, she is not arguing against the possibility that an element may well be embedded inside an element of the same type (which automatically makes a structure recursive). Rather, she is suggesting that this embedding cannot go on into infinitude (a slightly different matter) — that is, she is arguing for the finitude of argument structure (p. 253). Similarly, Yury A. **Lander &** Alexander B. **Letuchiy** provide data from a Northwest Caucasian language, Adyghe, that seems to allow self-embedding within its verb forms.

On a much grander scale, Harry **van der Hulst** discusses self-embedding in phonology, a topic that has generated some heated debate (as he discusses). Phonological structure is clearly hierarchical, but whether it also manifests self-embedding is rather controversial. This chapter defends the controversial view (phonology is recursive), and the overall idea seems to be that given that recursive structures are principally semantic phenomena (a manner of organizing information), there must be an isomorphic structure in morphology (p. 303). The remainder of the chapter offers a long discussion of phono-morphotactic structure, phonotactic structure, and prosodic structure, concluding that there is, after all, self-embedding at the syllable/foot, word, phrase and prosodic level. Pretty grand claims, and it will certainly be interesting to see what the literature makes of it (a thorough discussion of these issues is beyond the present review).

It will have been noticed that I have discussed all these papers in the context of structures only. Indeed, the study of self-embedded structures in natural language is an important one, but it ought to be clear that this phenomenon tells us much more about semantics than about syntax. Such structures, it is clear, provide the linguistic system with a way of "organizing and constraining semantic information", and their distribution appears to be construction- and language-specific (Hinzen 2008: 358–359).

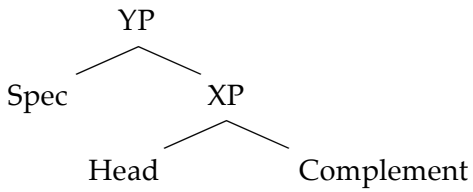
Once the dubious connection between these structures and specific rewriting rules is disregarded, it is not at all clear why some contributors believe that self-embedding cannot be converted into other types of phrases — a claim that is in fact explicitly denied in **Kinsella** (p. 188). Therein, Anna **Kinsella** makes clear that languages like Pirahã, even if they really do not manifest self-embedding, do not come at an 'expressive' loss to their speakers. That is, there is no reason to believe that Pirahã cannot "express [similar] concepts using alternative means". 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 some would call, I suppose, tail-recursive) or *The dog chased the cat and the cat bit the mouse and the mouse ran away* (a type of iterative structure, according to **Karlsson**).

Furthermore, there is a lot of interesting work regarding the concomitant properties that self-embedded structures exhibit, which range from their role in language acquisition and their cross-linguistic distribution, to their connection to the conceptual system (see, for example, Roeper 2009). Be that as it may, Merge remains a simple, recursive generator for reasons that lie elsewhere — the presence (or not) of self-embedded structures in a particular language is an ancil-

lary matter.

As a final point in this lengthy review, I might as well mention that there are, in fact, 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 (i.e. an X within an X) appears to be a feature of any type of syntactic structure. That is, every syntactic phrase, as Moro (2008: 68) shows, accords to the same geometry, an asymmetric structure [Specifier [Head – Complement]]. This is shown below:

(4)



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: 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 configuration: S is always more prominent than [H–C] and H is always more prominent than C.[9]

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. As such, category recursion is a subtype of structural recursion (in the same way that 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.

Certainly, an extensive terminological clean-up is in order, as much of the nomenclature currently in use (such as 'true, nested or center-embedding recursion', 'tail-recursion', 'self-embedding rules', *et alia*) is likely to create confusion rather than anything else.

Epilogue

It appears that the word recursion entered the English language in the 17th century as an adaptation of the past participle of the Latin verb *recurrere* 'to go back'. Thus in his *An English Expositor* (1616) — that compendium of the "hardest words" — John Bullokar defined *recursion* as "a running back". A convoluted term it remains, but for different reasons.

---

[9] The S–H–C schema invokes X-bar configurations. However, current linguistic theory doubts the existence of the specifier position. If so, the overall architecture would be something like this: […Head … (Compl) … […Head … (Compl) …] …]. The point I am making still applies; that is, this sort of general recursive structure is present in all languages, independently of the most usual form of self-embedding.

This critical note has attempted to outline four contemporary senses of this term that appropriately applies to well-established theoretical constructs of the cognitive sciences. An attempt was made to encompass the material of this collection around these four connotations as to elucidate a number of issues. Naturally, many topics went untreated, and the focus of this review has befallen on two main points.

Firstly, I have claimed that Chomsky, like many in the mathematical literature, takes recursion to be a central property of what a mechanical procedure is. Despite the different applications recursion has received within his vast output, a recent paper states that linguistic "competence is expressed by a generative grammar that *recursively enumerates* structural descriptions of sentences" (Chomsky 2006: 165; my emphasis) — a very close statement to the spirit of the Recursion Convention.

Secondly, I have tried to show that there is a clear conflation between, on the one hand, recursion and (self)-embedding and, on the other, recursive structures and recursive mechanisms. All these should in fact be kept separate unless there are principled reasons (and there might well be) to link them. Their connection, however, cannot be simply assumed.

It is rather clear that the present collection completely disregards the first point, while being guilty, for the most part, of the second. Perhaps we can concoct an explanation for why this collection so utterly fails to address Chomsky's actual introduction of recursion in linguistics — the overarching effect of one paper: Hauser *et al.* (2002).

It is undeniable that this paper has generated an incredible amount of discussion, but recursion was certainly not its main topic; indeed, to a certain extent, it received a rather indefinite characterization. This has had the unfortunate result that many recent publications on the role of recursion in cognition (and this is true for many of the contributions of the collection under review) come up with rather outlandish definitions, which are then loosely related to the aforementioned piece, even if on closer inspection, the actual work presented has very little to do with it — or more importantly, with Chomsky's leitmotif.[10]

Unfortunately, the literature is steadily moving towards an increasingly confused study of recursive structures in conflation with mechanisms, obscuring what ought to be a rather straightforward and uncontroversial point: The centrality of recursion within the formalization of the mechanical procedure that underlies the language faculty.

References

Abelson, Harold & Gerald J. Sussman with Julie Sussman. 1996. *Structure and Interpretation of Computer Programs*. Cambridge, MA: MIT Press.

[10] Moreover, there are good reasons to believe that both Hauser and Fitch have a different view (and, in my opinion, an incorrect one) to Chomsky on what recursion actually is; namely, a self-embedding operation (see Hauser 2009 and Fitch 2010).

Boolos, George. 1971. The iterative conception of set. *The Journal of Philosophy* 68, 215–231.

Chomsky, Noam. 1955. Logical syntax and semantics: Their linguistic relevance. *Language* 31, 36–45.

Chomsky, Noam. 1956. Three models for the description of language. In *IRE Transactions of Information Theory IT-2*, 113–124.

Chomsky, Noam. 1957. *Syntactic Structures*. The Hague: Mouton.

Chomsky, Noam. 1965. *Aspects of the Theory of Syntax*. Cambridge, MA: MIT Press.

Chomsky, Noam. 1980. *Rules and Representations*. New York: Columbia University Press.

Chomsky, Noam. 1986. *Knowledge of Language: Its Nature, Origin, and Use*. New York: Praeger.

Chomsky, Noam. 1995. *The Minimalist Program*. Cambridge, MA: MIT Press.

Chomsky, Noam. 2006. *Language and Mind*, 3rd edn. Cambridge: Cambridge University Press.

Chomsky, Noam. 2008. On phases. In Robert Freidin, Carlos P. Otero & Maria Luisa Zubizarreta (eds.), *Foundational Issues in Linguistic Theory: Essays in Honor of Jean-Roger Vergnaud*, 133–166. Cambridge, MA: MIT Press.

Chomsky, Noam & George A. Miller. 1963. Introduction to the formal analysis of natural languages. In Robert D. Luce, Robert R. Bush & Eugene Galanter (eds.), *Handbook of Mathematical Psychology*, vol. 2, 269–322. New York: Wiley.

Collins, John. 2008. *Chomsky: A Guide for the Perplexed*. London: Continuum.

Cutland, Nigel. 1980. *Computability: An Introduction to Recursion Function Theory*. Cambridge: Cambridge University Press.

Epstein, Richard & Walter Carnielli. 2008. *Computability: Computable Functions, Logic, and the Foundations of Mathematics*. Socorro, New Mexico: Advanced Reasoning Forum.

Fitch, W. Tecumseh. 2010. Three meanings of recursion: Key distinctions for biolinguistics. In Richard Larson, Viviana Déprez & Hiroko Yamakido (eds.), *The Evolution of Human Language*, 73–90. Cambridge: Cambridge University Press.

Hauser, Marc D. 2009. Origin of the mind. *Scientific American* 301, 44–51.

Hauser, Marc D., Noam Chomsky & W. Tecumseh Fitch. 2002. The language faculty: What is it, who has it, and how did it evolve? *Science* 298, 1569–1579.

Hinzen, Wolfram. 2008. Prospects for an explanatory theory of semantics. *Biolinguistics* 2, 348–363.

Kleene, Stephen C. 1952. *Introduction to Metamathematics*. Amsterdam: North Holland Publishing Co.

Knuth, Donald. 1997. *The Art of Computer Programming* (3 vols.). Upper Saddle River, NJ: Addison-Wesley.

Liu, Yanhong A. & Scott D. Stoller. 1999. From recursion and iteration: What are the optimizations? *SIGPLAN Notices* 34, 73–82.

McCarthy, John. 1963. A basis for a mathematical theory of computation. In Paul Braffort & David Hirshberg (eds.), *Computer Programming and Formal Systems*, 33–70. Amsterdam: North-Holland Publishing Co.

Moro, Andrea. 2008. *The Boundaries of Babel*. Cambridge, MA: MIT Press.
Moschovakis, Yiannis N. 1998. On founding the theory of algorithms. In Harold G. Dales & Gianluigi Oliveri (eds.), *Truth in Mathematics*, 71–104. Oxford: Clarendon Press.
Moschovakis, Yiannis N. 2001. What is an algorithm? In Björn Engquist & Wilfried Schmid (eds.), *Mathematics Unlimited: 2001 and Beyond*, 919–936. Berlin: Springer.
Moschovakis, Yiannis N. & Vasilis Paschalis. 2008. Elementary algorightms and their implementations. In S. Barry Cooper, Benedikt Löwe & Andrea Sorbi (eds.), *New Computational Paradigms*, 81–118. Berlin: Springer.
Piattelli-Palmarini, Massimo. 1980. *Language and Learning: The Debate between Jean Piaget and Noam Chomsky*. London: Routledge and Kegan Paul.
Rice, Gordon. 1965. Recursion and iteration. *Communications of the ACM* 8, 114–115.
Roberts, Eric. 2006. *Thinking Recursively with Java*. Hoboken, NJ: Wiley.
Roeper, Tom. 2009. *Microscopic Minimalism*. Boston University Plenary Address.
Sauerland, Uli. 2010. Experimental evidence for complex syntax in Pirahã. Ms., Zentrum für allgemeine Sprachwissenschaft, Typologie und Universalienforschung, Berlin. [http://ling.auf.net/lingBuzz/001095]
Soare, Robert. 1996. Computability and recursion. *The Bulletin of Symbolic Logic*, 2, 284–321.
Soschen, Alona. 2008. On the nature of syntax. *Biolinguistics*, 2, 196–224.
Tomalin, Marcus. 2006. *Linguistics and the Formal Sciences*. Cambridge: Cambridge University Press.

David J. Lobina
Universitat Rovira i Virgili
Departament de Psicologia
Centre de Recerca en Avaluació i Mesura de la Conducta
Ctra. de Valls s/n
43007 Tarragona
Spain
davidjames.lobina@urv.cat