Reconsidering recursion in syntactic theory

Marcus Tomalin *

*Tel.: +44 1223 766515.
E-mail address: mt126@cam.ac.uk.

1 Recent considerations of this topic include Christiansen and Charter (1999); Hauser et al. (2002); Premack (2004); Pinker and Jackendoff (2005), Chomsky et al. (2005), and Tomalin (2006). Also a recent Zentrum für allgemeine Sprachwissenschaft (ZAS) symposium entitled ‘Interfaces + Recursion = Language?’ (24/03/05) focused primarily upon the role of recursion and the interfaces in syntactic theory.

Abstract

This paper considers the role of recursion in syntactic theory, and three particular aspects of this broad topic are addressed. The use of recursive devices in formal grammars in the 1950s is summarised, and the work of Bar-Hillel and Chomsky from this period is analysed in some detail. Having established this historical context, the role of recursive definitions within the Minimalist Program is discussed at length, and the main focus falls upon comparatively recent claims concerning the centrality of recursion in the context of biolinguistics. Specifically, the hypothesis that recursion constitutes a species-specific property of the human language faculty that is particularly associated with natural language is (re)assessed, and it is suggested that, in the context of syntactic theory, the problematic term ‘recursion’ should be abandoned and replaced by less ambiguous terminology such as ‘inductive definition’.

© 2007 Elsevier B.V. All rights reserved.

Keywords: Recursion; Syntax; Biolinguistics; Minimalism

1. Introduction

In recent years, the role of recursion in syntactic theory has once again become a topic of extensive debate.¹ In some respects, this development is rather surprising because formal grammars have explicitly contained recursive components since (at least) the 1950s, and therefore it is certainly not the case that recursion has appeared suddenly and unexpectedly in the context of linguistic analyses. Indeed, with specific reference to Generative Grammar (GG), ever since the initial development of this influential framework in the 1950s, recursive devices of...
various kinds have been incorporated into the different versions of the theory. In particular, it is standardly claimed that such components are required mainly because (as Chomsky put it in 1956) “[i]f a grammar has no recursive steps [...] it will be prohibitively complex [...] If it does have recursive devices, it will produce infinitely many sentences” (Chomsky, 1956:116). Consequently, it is conventionally assumed that recursive components enable a potentially infinite set of potentially infinite syntactic structures to be generated by a given grammar. Initially, therefore, within the context of GG, recursive devices were viewed as useful formal mechanisms which, while finite in themselves, allowed infinite structures to be generated; and, in the earliest work, although recursive components were considered to constitute useful formal procedures that simplified the basic analytical framework, no strong claims were made concerning their biological status. Gradually, though, as the theory of GG developed, and, in particular, when the innateness hypothesis was formally adopted in the 1960s, the role of recursion within the GG framework began to acquire cognitive connotations, with the eventual result that, in the past few years, it has been hypothesised that recursion is a genetically-embedded computational procedure that is a central component of the human language faculty. Indeed, in recent work, it has been suggested that the most fundamental aspects of natural language can be exhaustively described simply in terms of recursion and mappings to the interfaces (e.g., Hauser et al., 2002). Although this hypothesis is peculiarly provocative, it cannot be comprehensively assessed without a secure understanding of various issues concerning the role of recursion in linguistic theory. For instance, in order accurately to evaluate biolinguistic hypotheses involving recursion, it is desirable to know something about (i) the manner in which recursion was initially incorporated into linguistic theories in the 1950s, (ii) the range of possible interpretations associated with the term ‘recursion’ in the related mathematical and logical literature, and (iii) the consequences that follow from (i) and (ii) for the perceived role of recursion within (Minimalist) GG. Accordingly, this paper seeks to elucidate some of these issues.

2. Recursion and linguistics

While it is apparently the case that certain linguists appear to have been aware of the recursive nature of natural language before the 1950s, for convenience the recent history of recursion in the context of linguistic theory can be traced back to 1953. This date can be taken as the starting point because it was in 1953 that the linguist and logician Yehoshua Bar-Hillel (1915–1975) published a short paper entitled ‘On Recursive Definitions in Empirical Science’. As the title suggests, Bar-Hillel argued that recursive definitions need not only be deployed in the context of pure mathematics and formal logic, and he claimed specifically that such definitions could greatly facilitate the analytical methodologies adopted by empirical sciences such as linguistics. As an example of elementary recursion, he considered the axioms for the natural numbers that the logician Giuseppe Peano (1858–1932) had presented in 1889 in his ‘Arithmetices Principia Nova Methodo Exposita’. However, Bar-Hillel does not cite Peano’s work directly, rather he refers to the summary that Stephen Kleene (1909–1994) had provided in his influential 1952 textbook Introduction to Metamathematics. Following Kleene, Bar-Hillel considers the two-part definition

\[ a + 1 = a' \]
\[ a + n' = (a + n)' \]  

(1)

---

2 In the 1960s, Chomsky traced the idea that languages make infinite use of finite means back to von Humboldt. For further details, see vanHumboldt (1989[1836]) and Chomsky (1966) (especially pp. 20–22).
and notes that, in this equation pair, the recursion is induced by the second step. Having considered the basic implications of adopting such functions in a mathematical context, Bar-Hillel suggests that, in the empirical sciences, definitions have often included statements that are actually recursive, even though this property has not been explicitly acknowledged before, and, in order to illustrate his point he considers several examples from linguistics. This manoeuvre is rather unexpected, since, at the time, recursive definitions were associated with the formal sciences, and were not explicitly used by linguists. However, despite this, Bar-Hillel works within a language-based domain, employing English as a metalanguage to analyse French, his chosen object language, and, while outlining his approach, he introduces the following definition (see Bar-Hillel, 1953:163):

**Definition 1: Sentence (Recursive in Disguise)**

\[ x \text{ will be called a sentence (in French) if (and only if) } x \text{ is a sequence of a nominal and a (intransitive) verbal, or a sequence of a nominal, a (transitive) verbal and a nominal or, . . ., or a sequence of a sentence, the word “et”, and a sentence, or, . . .} \]

Here, the terms “nominal” and “verbal” indicate ‘noun phrase’ and ‘verb’, respectively, and this definition accounts for sentences which combine clauses using the conjunction “et”. Having introduced Definition 1, Bar-Hillel demonstrates that although it does not explicitly manifest the two-part structure of a recursive definition, it can be classified as being “recursive in disguise”, and he accomplishes this by introducing an alternative definition (see Bar-Hillel, 1953:163):

**Definition 2: Sentence (Recursive)**

1. \( x \) is a sentence\(_1\) (a simple sentence) \(=_{df} x \) is a sequence of a nominal and a (intransitive) verbal or a sequence of a nominal, a (transitive) verbal, and a nominal, or . . .
2. \( x \) is a sentence\(_{n+1}\) (a compound sentence of the \( n \) + 1th order) \(=_{df} x \) is a sequence of a sentence\(_p\), the word “et”, and a sentence\(_m\), where either \( p \) or \( m \) (or both) are equal to \( n \) and none is greater than \( n \), or . . .

According to Bar-Hillel, Definition 2 constitutes “a pair of simultaneous recursive definitions”, and he claims that it is “simple” to check whether a given compound sentence is “a proper French sentence” or not: the compound structure can be broken down recursively into smaller units until a single simple sentence is obtained (Bar-Hillel, 1953:164). If this occurs, then the initial compound sentence is “proper”, and, if this cannot be achieved, then the sentence is (presumably) improper (i.e., ungrammatical). Appropriately, Bar-Hillel concludes his short

\(^3\) The various ellipses in the following examples are Bar-Hillel’s.
paper with an impassioned plea which was surely designed to intrigue those linguists who were interested in the relationship between the formal sciences and linguistic theory:

In conclusion, let me say that, in view of the role played by recursive definitions in concept formation in empirical science, it is the task of the methodologists to dedicate time and effort to the evaluation of their precise import in different fields of inquiry and the task of the scientists to become acquainted with the recent investigations on recursive definitions to a degree, at least, that would free them from the misconceptions that have so frequently been connected with their occurrence in disguise. (Bar-Hillel, 1953:165)

In Bar-Hillel’s opinion, then, certain fields of empirical inquiry were infested with “misconceptions” primarily because the researchers concerned had failed to realise that certain phenomena could be most easily analysed recursively. Given the topic of Bar-Hillel’s own discussion – namely, natural language – it would be difficult to avoid the conclusion that linguistics was one of the empirical sciences that was standardly bedevilled by such misconceptions.

As the above discussion has indicated, certain linguists working in the 1950s were well aware of the developments in recursive function theory that had occurred during the first decades of the 20th century, and it is sometimes possible to trace the routes along which recursive techniques entered the domain of linguistic enquiry. For instance, in 1991 Zellig Harris (1909–1992) recalled the intellectual origins of his own work in the 1950s:

The expectation of useful mathematical description of the data of language stems from developments in logic and the foundations of mathematics during the first half of the twentieth century. One main source was the growth of syntactic methods to analyse the structure of formulas, as in Skolem normal form and Löwenheim’s theorem, and in the Polish School of logic, (as in the treatment of sentential calculus in J. Łukasiewicz, and the categorial grammar of S. Leśniewski, and later K. Ajdukiewicz […]), and in W. V. O. Quine's Mathematical Logic (Norton, New York) of 1940. Another source is in the post-Cantor paradoxes constructionist views of L. E. J. Brouwer and the Intuitionist mathematicians, and in the specific constructionist techniques of Emil Post and Kurt Gödel, in recursive function theory, and from a somewhat different direction in the Turing machine and automata theory […]. (Harris, 1991:145)

Seemingly, Harris was persuaded that “recursive function theory”, and the work of Post and Gödel in particular, should be considered when a mathematical framework was being constructed for linguistic theory, and there is no doubt that a number of Harris’ students shared this conviction. In the following extract, for instance, Chomsky recalls his early days at MIT:

Perhaps a word might be usefully added on the general intellectual climate in Cambridge at the time when it [MT: i.e., The Logical Structure of Linguistic Theory] was written. Interdisciplinary approaches to language communication and human behaviour were much in vogue […]. Roman Jakobson’s work was well known and influential. Oxford ordinary language analysis and Wittgenstein’s later work were attracting great interest. The problem of reconciling these approaches (if possible) with Quine’s provocative ideas on language and knowledge troubled many students. Mathematical logic, in particular recursive function theory and metamathematics, were becoming more generally accessible, and
developments in these areas seemed to provide tools for a more precise study of natural
language as well. All of this I personally found most stimulating. (Chomsky, 1975[1955]:39)

So, for Chomsky, “recursive function theory” appeared to provide “tools” that could be
deployed by linguists, and passages such as these suggest that the use of recursive devices in
formal grammars was directly inspired by recursive function theory. Given this direct
association, it is necessary briefly to indicate the complex manner in which recursion was
presented in the mathematical logic literature produced during the first decades of the 20th
century. For instance, although Peano had presented recursive axioms in 1889 (as discussed
above), it was Gödel’s use of primitive recursive functions in his celebrated 1931 incompleteness
theorem that drew widespread attention to such techniques, and, in that paper, primitive recursive
functions were described as follows:

A number-theoretic function \( \phi \) is said to be recursive [rekursiv] if there is a finite sequence
[eine endliche Reihe] of number-theoretic functions \( \phi_1, \phi_2, \ldots, \phi_n \) that ends with \( \phi \) and has
the property that every function \( \phi_k \) of the sequence is recursively defined [rekursiv definiert
ist] in terms of two of the preceding functions, or results from any of the preceding
functions by substitution, or, finally, is a constant or the successor function \( x + 1 \).
(Gödel, 1986[1931]:159)

However, in a 1934 paper, Gödel extended this theory by defining general recursive functions.
While attempting to specify which classes of number-theoretic functions could be accurately
classified as ‘recursive’, Gödel had realised that there were certain effectively calculable
functions that were not primitive recursive functions (for instance, functions that were defined by
induction in respect to two variables simultaneously), and, consequently he was obliged to define
the wider class of general recursive functions.\(^4\) Given Gödel’s definitions of primitive and general
recursive functions, a number of researchers developed recursive function theory in the mid
1930s. For instance, in 1936, Alonzo Church (1903–1995) introduced the idea of recursively
enumerable sets, and these mathematical structures were influentially reformulated by Emil Post
(1897–1954) in the early 1940s (especially in Post, 1944). In Post’s presentation, a set of
nonnegative natural numbers, \( A \), is recursively enumerable if there is a general recursive
function, \( \phi \), which enumerates the members of the set. As will be demonstrated later, Post’s
presentation of recursively enumerable sets exerted a considerable influence upon the
development of syntactic theory in the 1950s.

As noted above, recursively enumerable sets were first introduced in 1936, and this proved to
be an *annis mirabilis* in the development of recursive function theory more generally, since in
that year Church and Alan Turing (1912–1954) separately published papers which introduced the
notions of \( \lambda \)-definability and computability, respectively, and these papers were largely
responsible for prompting the development of computability theory as an independent area of
research during the 1940s and 1950s.\(^5\) Since it had soon been demonstrated that general recursive
functions (as defined by Gödel) could be re-expressed both in terms of \( \lambda \)-calculus and computing
(i.e., Turing) machines, these three distinct theories swiftly became intertwined—sometimes
with problematical and perplexing consequences. In his *Introduction to Metamathematics*

\(^4\) Primitive recursive functions are defined in Gödel (1986[1931]), while general recursive functions are discussed in

\(^5\) For specifics, see Church (1936) and Turing (1936). An accessible historical summary can be found in Gandy (1988).
(the textbook that Bar-Hillel used), Kleene chose to focus predominantly upon recursive function theory, but he certainly indicated the co-extensiveness of the notions of general recursiveness, \( \lambda \)-definability, and computability. For instance, although he does not discuss \( \lambda \)-calculus in detail, he introduces “Church’s thesis” (i.e., the statement that every effectively calculable function is general recursive) on page 300, and he considers at some length the evidence that substantiates this thesis (Kleene, 1952:317–323). In a similar manner, “computable functions” are presented in chapter 13, and Kleene explicitly states “Turing’s thesis” (i.e., the statement that general recursive functions are computable, and vice versa) on page 376. More specifically, the equivalence between these three formalisms is emphasised in passages such as the following:

\[
\text{three notions arose independently and almost simultaneously, namely general recursiveness, } \lambda \text{-definability } \ldots \text{ and computability } \ldots \text{ The equivalence (i.e., co-extensiveness) of the } \lambda \text{-definable functions with the general recursive functions was proved by Church (1936) and Kleene 1936a } \ldots \text{ The equivalence of the computable to the } \lambda \text{-definable functions (and hence to the general recursive functions) was proved by Turing 1937. (Kleene, 1952:320)}
\]

Consequently, Kleene acknowledges the connections that relate these three formalisms, and, having established this, it is necessary now to consider the way in which some of the techniques considered above were incorporated into syntactic theory during the 1950s.

### 3. Recursion in Generative Grammar

Although it is well-known that, in its various 1950s guises, GG contained recursive components, the sources upon which Chomsky drew while formulating the initial versions of these components are not often discussed. As mentioned earlier, “recursive devices” (Chomsky, 1956:116) were required in GG in order to generate an infinite number of structures using only finite means, and it is clear that Chomsky’s ideas concerning these topics were directly influenced by the work of Bar-Hillel. For instance, Chomsky remarked in his paper ‘Logical Syntax and Semantics: their linguistic relevance’ that

At one point, Bar-Hillel suggests that recursive definitions may be useful in linguistic theory; whether this turns out to be the case or not, I agree in this instance with the spirit of his remarks. (Chomsky, 1955:45)

Given this endorsement of “the spirit” of Bar-Hillel’s suggestions concerning the use of recursive definitions in linguistic theory, it is no surprise that recursive techniques came to be used explicitly and extensively in Chomsky’s post-1955 work. It would be misleading, however, to claim that Chomsky encountered recursive definitions only in the work of Bar-Hillel. Indeed, to take one example, it is well-established that, during the 1950s, Chomsky was also profoundly influenced by Post’s work on recursively enumerable sets. For instance, in ‘Three Models for the Description of Language’, Chomsky stressed the desirability of creating formal grammars that were predictive, and he stated plainly that a GG-style approach to syntactic analysis requires “recursive devices” (Chomsky, 1956:116) in order to construct the full range of possible grammatical sentences. In order to achieve this, he drew upon Post’s theory of recursively enumerable sets, and this theory provided a useful framework that could be adapted for the purposes of theoretical syntax. As mentioned previously, the dominant intuition behind Post’s
theory was that all the elements belonging to a recursively enumerable set are in the range of a general recursive function, and, in a 1959 paper, Chomsky explained why sets of this type are important to the basic GG framework:

Since any language in which we are likely to be interested is an infinite set, we can investigate the structure of $L$ only through the study of the finite devices (grammars) which are capable of enumerating its sentences. A grammar of $L$ can be regarded as a function whose range is exactly $L$. Such devices have been called ‘sentence-generating grammars’. (Chomsky, 1959:137)

This passage indicates that, in mature GG, the set of grammatical sentences, $L$, was considered to be an infinite set of sentences, the elements of which could be generated by the function, $g$, where $g$ constitutes the grammar of the language $L$. It is worth noting that even the terminology used here reveals the connection between the theories: Post frequently used the verb ‘generate’ in order to describe how a recursively enumerable set is obtained from its associated general recursive function, and Chomsky explicitly acknowledges this terminological source, suggesting that he adapted these techniques directly from Post’s work. In a footnote following his use of the phrase ‘sentence-generating grammars’, Chomsky adds

Following a familiar technical use of the term ‘generate’ cf. Post (1944). The locution has, however, been misleading, since it has erroneously been interpreted as indicating that such sentence-generating grammars consider language from the point of view of the speaker rather than the hearer. Actually such grammars take a completely neutral point of view. [...] We can consider a grammar of $L$ to be a function mapping the integers onto $L$, order of enumeration being immaterial (and easily specifiable, in many ways) to this purely syntactic study [...]. (Chomsky, 1959:137 footnote 1)

This passage indicates that Post’s work concerning recursively enumerable sets was at least “familiar” to Chomsky by the late 1950s, and that the latter knowingly adapted the terminology that the former had introduced. This fact should be borne in mind throughout the following discussion since it is crucial to recognise that Chomsky was acquainted with Post’s own research and that he did not simply obtain his ideas concerning recursively enumerable sets from later expositions of the theory. Having considered some of the sources for Chomsky’s ideas concerning recursive components in formal grammars, it is necessary now to consider a few specific instances of the types of recursive devices that Chomsky used in his early work, and since the recursive aspects of the theory are presented most extensively in The Logical Structure of Linguistic Theory (1975[1955]; from henceforth $LSLT$), two different types of recursion discussed in $LSLT$ will be considered. The first type involves the successive application of the sequentially ordered rules of the grammar, while the second type involves the inclusion of rules within the grammar which are themselves recursive by definition.

In chapter 7 of $LSLT$ Chomsky presents the phrase structure component of his grammar. In particular, he proposes that this component can be specified as a sequence of conversions of the familiar form ‘$X \rightarrow Y$’, and he refers to this type of formulation as “a linear grammar”

---

6 The term ‘image’ is sometimes used rather than the term ‘range’ when functions are discussed. Since Chomsky generally uses the latter term, I have adopted this convention also in order to avoid confusion.
(Chomsky, 1975[1955]:194). However, he is quick to note that the finite nature of a linear grammar requires the sequence of rules to be applied iteratively:

The linear grammar is a sequence of conversions statements $S_1, \ldots, S_n$, where each $S_i$ is of the form $X_i \rightarrow Y_i$. We can produce derivations from this linear grammar by applying the conversions $S_i$ (interpreted as the instruction “rewrite $X_i$ as $Y_i$”) in sequence. Among the $S_i$ we distinguish between obligatory conversions that must be applied in the production of every derivation, and permissible conversions that may or may not be applied. There are only a finite number of ways to run through the linear grammar, applying all obligatory and some permissible conversions; hence only a finite number of derivations can be produced by the linear grammar $S_1, \ldots, S_n$. This was not a difficulty on earlier levels [MT: e.g., the phonemic level, the syntactic categories level], but we know that infinitely many sentences must be generated by some mechanism in the grammar. We can permit this infinite generation on the level $P$ by allowing the possibility of running through the linear grammar $S_1, \ldots, S_n$ an indefinite number of times in the production of derivations. If the derivation formed by running through the sequence of conversions does not terminate with a string in $P$ [MT: the set of strings composed of morphological heads and syntactically functioning affixes], then we run through the sequence again. Thus we can understand the linear grammar to be the sequence of conversions $S_1, \ldots, S_n; S_1, \ldots, S_n; S_1, \ldots, S_n; \ldots$ We then say that a derivation $D$ is recursively produced by the linear grammar $S_1, \ldots, S_n$. We define a proper linear grammar as a linear grammar which is so constructed that it is impossible to run through it over and over again vacuously. (Chomsky, 1975[1955]:194–195)

Intriguingly, as it is expressed here, the problem of the recursive application of conversions appears to be similar to Turing’s well-known ‘halting problem’: the constructional process fails if the conversions are applied repeatedly ad infinitum to no purpose. However, the main point is that Chomsky (at least) considered the repeated application of the conversion statements in the linear grammar to be a recursive procedure, and this type of recursion was introduced specifically to enable the grammar to generate an infinite number of sentences.

If the repeated application of conversions in LSLT constitutes one type of recursive procedure, then another type was associated with certain rules in the phrase structure part of the grammar—namely, rewrite rules such as ‘$X \rightarrow YX$’, where the category to the left of the arrow, ‘$X$’, also appears on the right-hand side of the rule, ‘$YX$’. Chomsky’s introduces this idea at the start of chapter 7 of LSLT, and he clarifies the manner in which these rules are used in GG:

When we turn to the level of phrase structure, we find that certain rules may have a recursive character. Thus Noun Phrase (NP) might be analysed in such a way that one of its components may be a NP as in such sentences as “the man who made the discovery is my brother”, which might be derived by means of a set of such conversions as

$$(1) \quad \text{NP} \rightarrow \text{NP}_1 \sim \text{who} \sim \text{VP}$$

$\text{VP} \rightarrow \text{V} \sim \text{NP}^3$

Conversions of this sort will permit the generation of infinitely many sentences by that part of the grammar that deals with phrase structure [...] (Chomsky, 1975[1955]:171–172)

---

7 While it would be possible for the sequence of conversions $S_1, \ldots, S_n$ to be applied recursively, Chomsky considers them to be iterative in nature—that is, the rules are simply applied one after another in sequential order, so no recursion is involved at this stage, at least in the LSLT formalism.
The formalism that Chomsky offers here does not provide the specific definitions of NP that are required in order to terminate the recursion. However, these are provided later in the chapter. Despite this, the basic method is clear: the phrase structure component contains recursive rewrite rules. In the above example, NPs can be rewritten as larger structures (containing relative clauses) which in turn contain NPs, and so on *ad infinitum* (at least potentially). Although Chomsky introduces such rules in chapter 7, he does not consider them in detail until chapter 8. For example, when discussing the structure of verb phrases, Chomsky proposes the following:

\[
\begin{align*}
\text{Sentence} & \rightarrow NP \\
VP & \rightarrow VP_A \rightarrow VP_1 \\
VP_1 & \rightarrow \langle VP_B \rangle VP_2 \\
VP_A & \rightarrow VP_{A1} \langle VP_{A2} \rangle \\
VP_{A1} & \rightarrow \{C, \text{ed}\}(M) \\
VP_{A2} & \rightarrow \langle \text{have} \rightarrow \text{en} \rangle \langle \text{be} \rightarrow \text{ing} \rangle \\
VP_B & \rightarrow Z_1 \langle Z_2 \langle \ldots \langle Z_n \rangle \rangle \rangle, \quad \text{where each } Z_i \text{ is one of the forms} \\
V_c \rightarrow \text{to}, V_r \rightarrow \text{ing} \\
\end{align*}
\]

(Chomsky, 1975[1955]:249–250)

where \(VP_A\) and \(VP_B\) both enable certain kinds of auxiliary verb constructions to be generated, with the difference that \(VP_B\) is defined so as to permit an unlimited number of \(Z_i\)'s. Using these rules, the sentence ‘John wants to read the book’ can be obtained as follows:

\[
\begin{align*}
\text{Sentence} & \\
NP & \rightarrow VP \\
NP & \rightarrow VP_A \rightarrow VP_1 \\
NP & \rightarrow VP_A \rightarrow VP_B \rightarrow VP_2 \\
NP & \rightarrow C \rightarrow VP_B \rightarrow VP_2 \\
NP & \rightarrow C \rightarrow \text{want} \rightarrow \text{to} \rightarrow VP_2 \\
John & \rightarrow C \rightarrow \text{want} \rightarrow \text{to} \rightarrow \text{read} \rightarrow \text{the} \rightarrow \text{book} \\
\end{align*}
\]

(Chomsky, 1975[1955]:250)

As this example indicates, the recursive rewrite rules were initially introduced into the phrase structure component of GG, and therefore they were deployed when the kernel of basic sentences was generated. However, as Chomsky elaborated his theory of transformations in chapters 9 and 10 of *LSLT*, the recursive rules were eliminated from the phrase structure component entirely, and were placed instead in the transformational component of the grammar. As a further modification, Chomsky also suggested that the recursive applications of conversion statements (discussed earlier) should be removed, and he summarised the new framework towards the end of chapter 10:

In the course of this analysis we have found that much of the recursive part of the grammar of phrase structure […] has been cut away. It seems reasonable to place the formal requirement that no recursions appear in the kernel grammar. Specifically we rule out such statements as [MT: \(VP_B \rightarrow Z_1 \langle Z_2 \langle \ldots \langle Z_n \rangle \rangle \rangle\)], and we drop the constructions […] that permit running through the grammar indefinitely many times. As far as I can determine,
this formal requirement on \( P \) does not exclude anything that we would like to retain in \( P \); nor does it impose any artificial or clumsy limitation on the actual statement of the grammar corresponding to \( P \), now that the transformational analysis presents an alternative way of generating sentences [...] Now that the higher level of transformational analysis as been established, it is no longer necessary to require that generation by the grammar of phrase structure is infinite. As the level \( T \) has been formulated, the process of transformational derivation is recursive, since the product of a \( T \)-marker can itself appear in the \( P \)-basis of a \( T \)-marker. (Chomsky, 1975[1955]:516–518)

Since the phrase structure component no longer contains recursive elements, the recursive part of the grammar must necessarily be located in the transformational component. Nevertheless, despite such modifications, 1950s-style GG required recursive rules of some kind, and, as suggested above, such rules were based on techniques that had been proposed initially in the mathematical and logical literature.

4. Minimalism and recursion

In the previous section, a (very) brief overview was given of the types of recursive devices that were used in the earliest versions of GG, and, despite the many changes that have resulted from the gradual and inevitable development of the theory during the past 50 years, one aspect that has remained remarkably constant is the need for some kind of recursive component. Indeed, in the recent MP literature, the centrality of recursion has been emphasised repeatedly and with increasing explicitness. For instance, in *The Minimalist Program*, Chomsky states that “the operations of \( C_{HL} \) recursively construct syntactic objects” (Chomsky, 1995:226), where the latter are defined as follows (see Chomsky, 1995:243):

**Definition 3: Syntactic Objects**

1. lexical items
2. \( K = \{\gamma\{\alpha, \beta}\} \), where \( \alpha, \beta \) are objects and \( \gamma \) is the label of \( K \)

This definition states that all lexical items (LIs) are syntactic objects, and that further syntactic objects can be created by combining existing syntactic objects in a principled manner. While discussing this definition, Chomsky explicitly notes that it is clause 2 which provides the “recursive step” (Chomsky, 1995:243), and, in order to clarify this point, he presents the following example:

Suppose a derivation has reached state \( \Sigma = \{\alpha, \beta, \delta_i, \ldots, \delta_n\} \). Then application of an operation that forms \( K \) as in [Definition 3, clause 2] converts \( \Sigma \) to \( \Sigma' = \{K, \delta_i, \ldots, \delta_n\} \) including \( K \) but not \( \alpha, \beta \). (Chomsky, 1995:243)

Since this summary is rather laconic, it is useful to work through a full example in order to appreciate the constructional framework that is being proposed. For instance, if it is assumed that \( C_{HL} \) contains only one operation, Merge, then derivation creation can be viewed as a process that
involves the repeated application of this operation, and this process terminates only when the initial numeration is mapped to a single syntactic object. Schematically, if $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ are all LI's in a given numeration and if they each have an index value of 1, then for a derivation that begins in state $\Sigma = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$, one possible series of subsequent steps can be explicitly represented as follows:

### Derivation: Example 1

Given $\Sigma = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$:

1. **Step 1:** $K_1 = \text{Merge}(\alpha_1, \alpha_2)$ and $\Sigma' = \{K_1, \alpha_3, \alpha_4\}$
2. **Step 2:** $K_2 = \text{Merge}(K_1, \alpha_3)$ and $\Sigma'' = \{K_2, \alpha_4\}$
3. **Step 3:** $K_3 = \text{Merge}(K_2, \alpha_4)$ and $\Sigma''' = \{K_3\}$

The derivation in Example 1 terminates when state $\Sigma'''$ is reached because only one syntactic object, $\{K_3\}$, remains. Since “no new objects are added [...] apart from rearrangements of lexical properties lexical properties” (Chomsky, 1995:228) during the course of the derivation, the whole process is entirely determined by the operation Merge and by the features associated with the LI's. In some versions of Minimalism, when Merge is applied in this manner, it causes two syntactic objects, $\alpha$ and $\beta$, to be combined in a principled fashion in such a way that a label derived from either $\alpha$ or $\beta$ can be determined and associated with the resulting hierarchically-structured syntactic object. In more recent work, the need for labels has been questioned (for example, see Collins, 2002; Chomsky, 2000), but even if a label-free system is proposed, the essential constructional process remains the same, and, significantly, Chomsky appears primarily to associate recursive definitions with the use of a finite set of primitive elements (i.e., LI's) and a finite set of operations (i.e., Merge) in order to generate a potentially infinite set of hierarchical structures (i.e., sentences). Crucially, it is essential to note that, since the process given in Definition 3 terminates when the numeration is reduced to a singleton set, this state of affairs constitutes the base-case which halts the recursion.

Given the need for some form of recursion within the MP framework, it is of interest that, in recent work, Chomsky has re-emphasised the connections that exist between the kind of recursive component standardly adopted within the MP (summarised in the previous paragraph) and the sort of recursive definitions historically associated with elementary arithmetic. For instance, in a recent paper, ‘On Phases’, Chomsky states the following:

Suppose that a language has the simplest possible lexicon: just one LI, call it “one’. Application of Merge to the LI yields \{one\}, call it “two”. Application of Merge to \{one\} yields \{\{one\}\}, call it “three”. Etc. In effect, Merge applied in this manner yields the successor function. It is straightforward to define addition in terms of $\text{Merge}(X, Y)$, and in familiar ways, the rest of arithmetic. (Chomsky, 2005:6)

---

In this passage, the “successor function” referred to can be associated with the function that had been used so influentially by Peano in 1889 in order to define the natural numbers inductively, and which was discussed in relation to recursive function theory by Gödel, Kleene, and others during the first decades of the 20th century. Given Definition 3, though, it is necessary to clarify a few aspects of the above example. For instance, since the lexicon contains only a single element (i.e., the LI ‘one’), it follows that the numeration associated with each derivation in this language can only contain a single LI, and, as it is presented here, this appears to suggest the LI ‘one’ cannot even be merged once since the numeration only contains a single element, a state of affairs which terminates the derivation. Presumably, therefore, it is assumed in the above passage that the LI ‘one’ is associated with an index, $i$, such that $1 \leq i \leq \infty$. Consequently, a given numeration can be viewed as the pair $(one, i)$, where ‘one’ is the LI in the numeration and $i$ is the index associated with this LI, and, since the index is reduced by 1 every time the LI in the numeration is merged, the LI can be merged repeatedly in order to produce the sequence of natural numbers. However, it is crucial to note in addition that the objects generated by the repeated application of Merge are not associated with labels (at least as Chomsky presents them here), and therefore the computational processes of arithmetic do not seem to require the same information concerning hierarchical structure that is required by the computational procedures that generate syntactic objects such as those defined in fully developed standard MP formalisms (e.g., Definition 3). If it is indeed the case that an MP-style successor function which generates the set of natural numbers (and, eventually, “the rest of arithmetic”) does not require hierarchical information concerning the structures produced, then this implies that there is a distinction of some kind between the types of recursive processes that are required in order to produce elementary arithmetic, and those that are required in order to produce natural language syntax.

Nevertheless, despite possible differences of this sort, the basic analogy between recursion in the MP and Peano’s successor function suggests that, in the current MP framework, LIs are inductively defined, in a way that is similar to the manner in which the members of the set of natural numbers are defined, and therefore a correspondence (of some kind) can be identified as existing between linguistic and mathematical cognitive function:

The emergence of the arithmetical capacity has been puzzling ever since Alfred Russell Wallace, the co-founder of modern evolutionary theory, observed that the “gigantic development of the mathematical capacity is wholly unexplained by the theory of natural selection, and must be due to some altogether distinct cause” if only because it remained unused. It may, then, have been a side product of some other evolved capacity (not Wallace’s conclusion), and it has often been speculated that it may be abstracted from LF [MT: logical form] by reducing the latter to its bare minimum. Reduction to a single-membered lexicon is a simple way to yield this consequence. (Chomsky, 2005:6)

As a result of this perceived connection between the linguistic and arithmetical capacities, Chomsky has recently suggested that, rather than merely constituting a useful constructional procedure that was initially derived from the formal sciences, recursion may actually constitute a unique language-related aspect of human cognitive function. For instance, in a 2002 paper ‘The Faculty of Language: What Is It, Who Has It, and How Did It Evolve’, which Chomsky co-authored with Marc Hauser and Tecumseh Fitch, the authors discuss what they refer to as the Faculty of Language in the Narrow sense (FLN), and they indicate that this term denotes “the abstract linguistic computational system alone, independent of the other systems with which it interacts and interfaces” (Hauser et al., 2002:1571). Further, they state explicitly that “a core
property of FLN is recursion”, before amplifying this claim a little by summarising the now familiar belief that “FLN takes a finite set of elements and yields a potentially infinite arrangement of discrete expressions” (Hauser et al., 2002:1571). So far, none of this is especially unusual within the GG tradition. However, in addition to these established claims, Hauser, Chomsky, and Fitch go on to hypothesise the following:

[. . .] we suggest that FLN - the computational mechanism of recursion - is recently evolved and unique to our species [. . .] we propose in this hypothesis that FLN comprises only the core computational mechanisms of recursion as they appear in narrow syntax and the mappings to the interfaces. If FLN is indeed this restricted, this hypothesis has the intriguing effect of nullifying the argument from design, and thus rendering the status of FLN as an adaptation open to question. (Hauser et al., 2002:1573)

In essence, the hypothesis is that FLN (a unique species-specific property) consists only of recursion and the interface mappings. Clearly, then, within the context of contemporary GG, recursion now plays a crucial role. More specifically, while recursive definitions were initially viewed simply as useful procedures derived from the formal sciences which (as Bar-Hillel had indicated in 1953) could be incorporated into formal grammars in order to facilitate linguistic analyses, if the above hypothesis proves to be correct, then, in contemporary GG, recursion will have to be viewed as the most fundamental component of the faculty of language.

It is no surprise, perhaps, that this bold hypothesis has been criticised by certain linguists. For instance, in a recent critique, Pinker and Jackendoff have argued that Hauser, Chomsky, and Fitch use the term recursion with undesirable vagueness. Specifically, while claiming that phonological structures are not recursive, Pinker and Jackendoff observe that:

[. . .] the segmental/syllabic aspect of phonological structure, though discretely infinite and hierarchically structured, is not technically recursive. (As mentioned, HCF [i.e., Hauser, Chomsky, and Fitch] use “recursion” in a loose sense of concatenation within hierarchically embedded structures). Recursion consists of embedding a constituent in a constituent of the same type, for example a relative clause inside a relative clause (a book that was written by the novelist you met last night). This does not exist in phonological structure: a syllable, for instance, cannot be embedded in another syllable. (Pinker and Jackendoff, 2005:10)

The main claim here is that the kind of recursive procedures adopted by Hauser, Chomsky, and Fitch cannot be classified as being “technically recursive” simply because they are defined too loosely, and, in response to this, rather than merely associating recursion with discrete infinity, Pinker and Jackendoff explicitly stress that fact that recursion must involve “embedding a constituent in a constituent of the same type”. Disagreements of this kind suggest that different linguists interpret the word recursion in different ways—an alarming state of affairs, if true.

Given the discussion in section 2, though, it should not be surprising that the Hauser–Chomsky–Fitch (HCF) hypothesis (as it currently stands) is potentially ambiguous, and it should be obvious that this ambiguity is caused primarily by the confusion that has enveloped the term ‘recursion’ since the late 1930s. In short, as mentioned earlier, the post-1936 development of recursive function theory was both complex and bemusing, with the result that, although the term ‘recursion’ is still used within the domain of contemporary mathematics, it is
often used by different people to refer to different things; and the central cause of this undesirable state of affairs is the equivalence of (general) recursiveness, $\lambda$-definability, and computability. Indeed, ever since the pioneering work of Gödel, Church, and Turing, these three distinct fields have overlapped with the unfortunate result that research undertaken within the framework of (say) computability theory (i.e., research which explicitly uses Turing-machines, computable functions, and so on) is often formally expressed in terms of recursive function theory. A decade ago, the mathematician Robert Soare referred to this as the “Recursion Convention”, and highlighted the dangers of this lamentable terminological muddle (Soare, 1996:28):

The Recursion Convention has brought “recursive” to have at least four different meanings [. . .] This leads to some ambiguity [. . .] Worse still, the Convention leads to imprecise thinking about the basic concepts of the subject; the term “recursion” is often used when the term “computability” is meant (By the term “recursive function” does the writer mean “inductively defined function” or “computable function”?) Furthermore, ambiguous and little recognized terms and imprecise thinking lead to poor communication both within the subject and to outsiders, which leads to isolation and a lack of progress within the subject, since progress in science depends upon the collaboration of many minds. (Soare, 1996:29)

This is a disquieting passage, and it certainly emphasises the current uncertainty that surrounds the use of the term ‘recursion’ within the formal sciences. Consequently, in the light of these remarks, it is necessary to reconsider the implications of the HCF hypothesis, and it is apparent that the first task is simply to attempt to determine which type of ‘recursion’ it implicates. In order to accomplish this, it is important to note specifically that there is currently a range of possible interpretations of the term ‘recursion’, and only the most obvious are enumerated below:

- (I1) ‘recursion’ = inductive definition (à la Peano, 1958[1889])
- (I2) ‘recursion’ = primitive recursion (à la Gödel, 1986[1931])
- (I3) ‘recursion’ = general recursion (à la Gödel, 1986[1934])
- (I4) ‘recursion’ = $\lambda$-definability (à la Church, 1936)
- (I5) ‘recursion’ = computability (à la Turing, 1936)

Given the discussion in section 2, it should be clear that the implications of the five definitions given here are very different. For instance, the claim that the FLN contains a component that is primitive recursive (i.e., (I2)) is not the same as the claim that the FLN contains a component that is (Turing) computable (i.e., (I5)), since while it is the case that all primitive recursive functions are (Turing) computable, it is not the case that all (Turing) computable functions are primitive recursive. The task, therefore, is to determine precisely which interpretation is implied by the HCF hypothesis. Regrettably, the informal discussion of the subject contained in the Hauser, Chomsky, and Fitch paper makes it impossible to determine which one of the above interpretations (if any) is assumed. Certainly the authors seem explicitly to associate ‘recursion’ with the process of creating an infinite arrangement of discrete expressions from a finite set of elements (Hauser et al., 2002:1571, quoted previously), and this is undoubtedly reasonable, although it is essential to note that this is a far more general procedure than those associated with any of the five possible interpretations of ‘recursion’. For instance, if the sole requirement is to generate an infinite number of structures using finite means, then an iterative, rather than a ‘recursive’, process could accomplish this, and while such a procedure may well be less efficient than a ‘recursive’ procedure,
the basic point is that a requirement for infinite structures to be created using finite means is not in itself sufficient to motivate the use of specifically 'recursive' techniques.

In order to relate these observations more directly to the five equalities given above, it is useful to consider a number of possibilities in more detail. For example, if interpretation (I2) is adopted in order to clarify the implications of the HCF hypothesis, then, the recursive component must be viewed as utilizing primitive recursive procedures in order to generate infinite structures from finite means, and, if this were the case, then the type of ‘recursion’ associated with \( C_{HL} \) would necessarily involve procedures that implemented explicitly constructional techniques. However, since self-reference is not stressed in their paper, Hauser, Chomsky, and Fitch may be using the term ‘recursion’ to indicate something more general such as ‘effectively calculable’ or ‘specified by a finite algorithm’, in which case an interpretation of ‘recursion’ that associates it with \( \lambda \)-computability or computability theory, respectively, rather than with Gödelian recursive function theory, would seem to be the most likely option (e.g., either (I4) and (I5)). However, if this were the case, then it would surely be preferable to refer specifically to the ‘recursive’ component as being either ‘a \( \lambda \)-definable component’ or else ‘a Turing machine’ since this would obviate the otherwise problematic vagueness that surrounds the current terminology. Indeed, as this brief discussion has indicated, given the brief and non-technical presentation offered by Hauser, Chomsky, and Fitch, it is simply impossible to determine with certainty which interpretation of ‘recursion’ the authors are actually adopting.

Thankfully, this indeterminate conclusion does not prevent an independent evaluation of the five interpretations of ‘recursion’, and, if such an evaluation is conducted with full awareness of the mathematical context from which the syntactic theory of ‘recursion’ was initially adapted, then it is perhaps possible to delimit the range of plausible interpretations of this term when it is used in relation to linguistic theory. For instance, it should be obvious that the most general of the five interpretations enumerated above is (I1), since this interpretation makes fewer particular claims about the types of objects that can be generated by the theory proposed, and therefore it is worth attempting first to determine whether it is necessary to adopt an interpretation that is more specific than (I1). In effect, then, if (I1) (i.e., essentially the interpretation that Bar-Hillel proposed in 1953) were to be adopted as the interpretation of the word ‘recursion’ as it is used in relation to the constructional procedures associated with \( C_{HL} \), then the HCF hypothesis would essentially resolve itself to the (comparatively) transparent claim that, in addition to the interface mappings, the FLN contains a component that simply enables syntactic objects to be defined inductively. This interpretation is attractive for various reasons, and, in particular, it accomplishes the following:

- It captures the characteristic properties of ‘recursion’ which seem to be required by \( C_{HL} \), at least as these are encapsulated in definitions such as Definition 3.
- It does not make undesirably specific claims about the manner in which syntactic objects must be generated, a characteristic that is not manifestly true of interpretations ((I2)–(I5)).
- It does not undermine the perceived association between elementary arithmetic and the faculty of language, since, as indicated in the foregoing sections of this paper, these can both be analysed using inductive definitions.

Since other interpretations of the term ‘recursion’ (i.e., ((I2)–(I5))) necessarily make stronger (possibly unwarranted) claims concerning the nature of FLN, it seems reasonable initially to adopt the ‘inductive definition’ interpretation (I1), and to replace this with an interpretation that makes more elaborate claims only if it proves to be inadequate in some way. Presumably the task
of establishing the inadequacy (or otherwise) of such an interpretation is primarily an empirical issue that can only be determined by means of the usual process of assessing particular data in relation to theory-based predictions concerning linguistic structure. Since the (I1) interpretation is formally equivalent to the definition of ‘recursion’ stated in Definition 3 (and other closely related variants), though, there would appear to be no conspicuous empirical reason for this interpretation not to be adopted. However, if (I1) is accepted as an accurate specification of the type of ‘recursive’ procedures that are associated with \( C_{HL} \), then it would be preferable to cease to use the term ‘recursion’ and its cognate forms entirely when discussing the FLN and the constituent structure of \( C_{HL} \), since, as the foregoing discussion demonstrates, it would be preferable to substitute more accurate terminology (such as ‘inductive definition’) in place of the problematic word ‘recursion’. Consequently, and in summary, \( C_{HL} \) can be viewed as a computational component that enables syntactic objects to be generated by means of a procedure that utilizes inductive definitions, and there is no need to refer to ‘recursive’ components of any kind.

5. Conclusion

This paper has attempted to accomplish a number of tasks. Section 2 summarised Bar-Hillel’s suggestion that syntactic theory would benefit if recursive definitions were used, and a brief overview of the recursive function theory, \( \lambda \)-calculus, and computability theory was given. Having sketched in this historical context, the use of recursive components in the earliest GG formalisms was assessed in section 3, while the use of similar techniques in the MP was considered in section 4. Further, the HCF hypothesis was presented, and it was suggested that, as it currently stands, this hypothesis is simply too vague to be either accepted or rejected. This vagueness is primarily caused by ambiguities inherent in the notion of ‘recursion’, as this term is standardly used within the formal sciences. In response to this, it was proposed that, in the context of linguistic theory, the term ‘recursion’ should be understood to mean ‘inductive definition’ since, if this interpretation of the term is adopted, the ‘recursive’ components of \( C \) are able to accomplish all that is required of them (i.e., the generation of infinite structures using finite means) and the correspondence between the human linguistic and arithmetical cognitive functions is not undermined. Any other interpretation of the term ‘recursion’ makes stronger (and possibly undesirable) claims about the nature of FLN, and therefore should be avoided if possible. If this proposal is accepted, then it would be better to abandon the term ‘recursion’ entirely in the context of linguistic theory, and replace it by a more precise phrase such as ‘inductive definition’.

As should be apparent from the above summary, this paper does not promulgate a specific stance concerning the role of ‘recursion’ within contemporary GG. More specifically, this paper does not promulgate a specific stance concerning the validity (or otherwise) of the HCF hypothesis. In essence, this paper primarily constitutes a plea for greater precision and clarification in discussions concerning the role of ‘recursion’. If ‘recursion’ really is the most fundamental component of the faculty of language (in the narrow sense), then surely it is essential to determine the properties of this component as precisely as possible. In particular, if avoidable ambiguity is identified in discussions of this topic, then surely such ambiguity should be eradicated. If this eradication does not occur, and if current debates concerning the role of ‘recursion’ within syntactic theory continue without enhanced accuracy and lucidity, then misunderstandings and confusion will necessarily ensue, and an avoidable development of this kind would be objectionable since it would inevitably have a pernicious effect upon what could otherwise prove to be a profoundly significant exploration.
References