Cogn Process (2011) 12:1–11
DOI 10.1007/s10339-010-0368-6

REVIEW

The redundancy of recursion and infinity for natural language

Erkki Luuk · Hendrik Luuk

Received: 3 December 2009 / Accepted: 7 July 2010 / Published online: 23 July 2010

Abstract An influential line of thought claims that natural language and arithmetic processing require recursion, a putative hallmark of human cognitive processing (Chomsky in Evolution of human language: biolinguistic perspectives. Cambridge University Press, Cambridge, pp 45–61, 2010; Fitch et al. in Cognition 97(2):179–210, 2005; Hauser et al. in Science 298(5598):1569–1579, 2002). First, we question the need for recursion in human cognitive processing by arguing that a generally simpler and less resource demanding process—iteration—is sufficient to account for human natural language and arithmetic performance. We argue that the only motivation for recursion, the infinity in natural language and arithmetic competence, is equally approachable by iteration and recursion. Second, we submit that the infinity in natural language and arithmetic competence reduces to imagining infinite embedding or concatenation, which is completely independent from the ability to implement infinite processing, and thus, independent from both recursion and iteration. Furthermore, we claim that a property of natural language is physically uncountable finity and not discrete infinity.

E. Luuk (✉)
Institute of Computer Science, University of Tartu,
Tartu, Estonia
e-mail: erkkil@gmail.com

H. Luuk
Department of Physiology, University of Tartu, Tartu, Estonia

H. Luuk
Department of Clinical Biochemistry, Bispebjerg Hospital,
Copenhagen, Denmark

E. Luuk
Postimaja Pk 149, Tartu 50002, Estonia

Keywords Recursion · Iteration · Language · Brain · Infinity · Embedding · Arithmetic

Introduction

An influential line of thought claims that a hallmark of human cognitive processing is recursion (Christiansen and Chater 2003; Fitch et al. 2005; Hauser et al. 2002; Hurford 2004; Premack 2007). The proponents of this view, most famously Noam Chomsky, appeal to the facts that (a) there is no non-arbitrary upper bound to sentence length, (b) the set of natural numbers **N** is infinite, (c) humans are able to perform arithmetic on **N**, and to the conjecture, drawn from Wallace's paradox, that natural language and arithmetic processing are mediated by the same faculty (Chomsky 2010; Fitch et al. 2005; Hauser et al. 2002). From this, they draw a more general conjecture that the faculty of language must include a neurally implemented recursive process to account for these facts. In this paper, we analyze natural language and arithmetic performance and conclude that there are no compelling reasons to posit neurally implemented recursion. We show that since recursion is a process or procedure, it cannot, in principle, be inferred from structure alone. When this stance is adopted, several popular arguments for recursion in natural language and arithmetic performance are downplayed. In addition, we emphasize that the infinity in natural language and arithmetic competence reduces to imagining infinite embedding or concatenation, which is independent from a neurally implemented recursive process. Due to a number of theoretical and practical considerations, we conclude that neurally implemented recursion might be altogether impossible to find evidence for.

Recursion and embedding

Hauser et al. (2002) drew a distinction between the whole language faculty, including the aspects shared with other species or faculties (the faculty of language in the broad sense) and the unique aspects of the language faculty (the faculty of language in the narrow sense). They hypothesized that the unique aspects of the language faculty comprise "only the core computational mechanisms of recursion as they appear in narrow syntax and the mappings to the Sensory-Motor and Conceptual-Intentional interfaces" (Hauser et al. 2002, p. 1573). Lately, this hypothesis has been vigorously challenged (Jackendoff and Pinker 2005; Parker 2006; Pinker and Jackendoff 2005). Interestingly, none of the challenges question the need for recursion (or infinity) in natural language. Although the viability of a biologically determined Universal Grammar (and thus, the faculty of language in the narrow sense) has been recently challenged from evolutionary (Christiansen and Chater 2008) and linguistic typological (Evans and Levinson 2009) viewpoints, recursion is still almost universally accepted as the explanation to the generativity (i.e. open-endedness) in language. The only exceptions appear to be Lieberman (2008, 2010) and Bickerton (2009), both of whom raise the notion of iteration, contesting recursion from a neuroscientific and linguistic perspective, respectively. In this section, we suggest that recursion is not required for natural language. First, we will focus on the logical contingencies of recursion and embedding. Second, as the narrower claim of Hauser et al. that recursion is unique to our species has subsequently been questioned (Marcus 2006; Okanoya 2007; Watanabe and Huber 2006), we will argue that recursion in non-human animal communication has so far not been attested.

Recursion

Defining recursion

First, in order to bring the notion of recursiveness into a more general perspective, we will inspect the definition of recursion in its original context—mathematical logic. Rogers Jr. (1987, pp. 5–6) gives the following description of a Gödelian recursive definition: "A recursive definition for a function is, roughly speaking, a definition wherein values of the function for given arguments are directly related to values of the same function for "simpler" arguments or to values of "simpler" functions." For example, in the recursive function for defining Fibonacci numbers (for integers $n > 1$), Fib(n) is directly related to Fib(n-1) and Fib(n-2): Fib(n) = Fib(n-1) + Fib(n-2). For a mathematical definition of primitive recursion, see Odifreddi (1992, p. 20). For the present discussion, the most important aspect of recursion lies in its ability to describe an infinity of (input, output) pairs by a finitely definable set of operations in an effectively computable manner (see Odifreddi, 1992, pp. 34–36, for details). Thus, recursion has a potential to explain the generation of open-ended symbol combinations in natural language and arithmetic performance.

Recursion differs from iteration (another form of repetition) in two essential respects. First, definitions employing iteration do not involve self-reference. Second, without self-reference, every (input, output) pair needs to be defined explicitly, rendering it impossible to define infinite sets by finite means other than by a control flow loop. Computationally, there is a clear difference between a procedure that invokes another instance of itself (recursion) and a procedure that repeats itself either mechanically or with a control flow loop (iteration). Recursion and iteration are the only computational solutions for handling repetition.

As a rule, implementations of recursive functions are slower than those of iterative because recursive functions must allocate memory for their multiple instances. In Abelson et al. (1996), the difference between recursive and iterative *processes* is captured as follows. Recursive processes are characterized by a chain of deferred operations and require that the interpreter keep track of the operations to be performed later on. An iterative process is one whose state can be summarized by a fixed number of state variables, together with a fixed rule that describes how the state variables should be updated as the process moves from state to state. All this being said, recursive *procedures* (or definitions) tend to be formally and notationally more elegant than iterative ones. The difference between process and procedure is explained in 2.1.2.

Returning to linguistics, we will first attempt to identify the essential properties of recursion that have led to the view that recursion is a component of the language faculty (Christiansen and Chater 2003; Fitch et al. 2005; Hauser et al. 2002; Hurford 2004; Jackendoff and Pinker 2005; Pinker and Jackendoff 2005; Premack 2007). For this, we return to the influential paper by Hauser et al. (2002), which was the first to explicitly formulate this view (regrettably, no definition of recursion was given in the article). After reviewing all statements about recursion in the paper, the following definition emerges (numbers in brackets refer to the pages in Hauser et al. (2002) where the statements appear): recursion is a neurally implemented (p. 1574) computational mechanism (p. 1573) that yields a potentially infinite array of discrete expressions (pp. 1570, 1571, 1574) from a finite set of elements (p. 1571). Crucially, a computational mechanism with finite input and potentially infinite output, described here, does not imply recursion, as it is possible to generate a potentially infinite output from a finite set of elements without recursive operation by implementing $n \rightarrow \infty$

operations iteratively. Hence, it is not clear whether the mechanism described by Hauser et al. (2002) is recursive or, perhaps, merely iterative (cf. Tomalin 2007). While it is plausible that the notion of recursion as applied by Hauser et al. (2002) refers to the 'recursion' in the Minimalist Program (Chomsky 1995), the latter allows for a range of interpretations. Tomalin (2007, p. 1797) has, usefully, distinguished between five different interpretations of 'recursion' in formal sciences, with 'inductive definition' as the most broad one (and thus the safest for syntactic theory). Tomalin's (2007) theoretic variants and computational equivalents of recursion (λ-definability, Turing computability et al.) are more or less on the same level of abstraction but recursion can also be defined in five different levels (in the order of decreasing abstractness) (Table 1).

We suggest that, for cognitive and neural modeling, the procedural level (4) is of central importance. For the following discussion, it is crucial that the recursion we are looking for is implemented in the brain (Hauser et al. 2002, p. 1574), i.e. it is not something that is posited at the level of computational theory only (Marr's (1982) level 1 of his three levels of information processing). It is also important to note that levels 4 and 5 in Table 1 correspond to Marr's level 2 (algorithm and input/output representation) of information processing. In the next section, we will examine whether and how any of the levels in Table 1 can be connected to Marr's level 3 (hardware implementation) in the brain.

Recursion, iteration, and inductive definition

All open-ended sets (e.g. language expressions, **N**) can be defined inductively, i.e. recursively in the broadest sense. For example, one can have the following inductive definition of 'bear': (a) *Ted is a bear*; (b) *All entities that share at least 98% of Ted's genome are bears*. Observe that, although the set of potential 'bears' is open ended and inductively defined, no recursion is computationally necessary to determine its contents. An iterative process that compares Ted's genome to that of potential 'bears' would do the job. Importantly, the difference between iteration and recursion pertains to levels 4–5 only. Thus, even if recursion were used in levels 1–3, the involvement of a recursive process or procedure would not be implied, as it can be implemented with a purely iterative computational process (e.g. on a Turing machine). Thus, at the very least, one must distinguish between the definition and processing levels.

In some instances, induction may be necessary (or at least the optimal way) for defining a set. However, this is not the case with the outputs of recursive rewrite rules for which a general rule coupled with iteration gives an alternative definition. Below is a strip of iterative pseudocode that defines the infinite set of finite strings …[X[X[X[XY]]]] that can be also defined with the recursive rewrite rule Y → XY:

```
while true:   //infinite loop (with a condition that always holds true):
  Y = concatenate(X, Y)   //append Y to X, assign the result to Y
```

From the viewpoint of effective calculability, general recursion and Turing computability are equivalent (Kleene 1952, p. 300), and a universal Turing machine can enumerate any recursively enumerable formal language (an infinite set of finite strings) as can general recursion (Sipser 1997). Turing machine is based on iteration, whereas general recursion is based only on recursion. Over finite outputs, recursion and infinite iteration are computationally equivalent (Turing-equivalent).

We must also distinguish between recursion as a formal explanation and recursion as something that is implemented in the brain. We cannot rule out recursion as a parsimonious formal explanation for, example, adjective left-branching in NPs, i.e. we cannot preclude the rule NP → A NP as a formal explanation. It is a completely different matter; however, whether the rule is somehow implemented in the brain. Crucially, (a.1) a neurally implemented procedure (that defines a neurally implemented process) is impossible to find evidence for with present-day methods. As there are no explicit formulations as to what a neural description of a recursive procedure could look like, we are unable to isolate it in the brain and, accordingly, unable to verify its existence. Thus, an evidence for (b.1) a neurally implemented process which satisfies (a.1) is ruled out as well. On the other hand, Lieberman (2008, p. 527) has recently suggested that "neural circuits linking local operations in the cortex and

Table 1 'Recursion' in five different levels

1. Inductive (or recursive) definition: a definition with a base rule, specifying that certain "simple" objects are in the class, and an inductive (recursive) rule, stating that if certain objects are in the class, then so are certain other objects formed or derived from them (Minsky 1972)
2. Recursive definition for a function: a definition wherein values of the function for given arguments are directly related to values of the same function for "simpler" arguments or to values of "simpler" functions (Rogers Jr. 1987)
3. Recursive function: a function that is defined recursively (see level 2)
4. Recursive procedure: a procedure that, when executed, invokes another instance of itself (Abelson et al. 1996)
5. Recursive process: an execution of a recursive procedure (see level 4)

the basal ganglia confer reiterative capacities, expressed in seemingly unrelated human traits such as speech, syntax, adaptive actions to changing circumstances, dancing, and music", thus obviating the need for a neural implementation of recursion. Of course, the distinction between neurally implemented recursive and iterative processes is rather opaque for present-day methods, i.e. the possibility of a neurally implemented recursion cannot be ruled out. We argue for iteration only as a simpler and equipotent computational alternative to recursion.

In sum, we have seen that inductive definitions (level 1 in Table 1) are not necessary for defining the outputs of recursive rewrite rules. Moreover, even if they were necessary, there would be no sense in (and no obvious way of) implementing level 1 in the brain *separately from levels 4 and 5*. As for levels 2 and 3, it would be outlandish to assume that the brain somehow (and apparently redundantly) implements *equations* of the processes it carries out. Thus, implementations of these levels can be ruled out as well. We have also argued that implementations of levels 4 and 5 would be, even theoretically, extremely difficult to isolate in the brain with present-day knowledge and methods.

Recursion and self-embedding: a confusion

Regrettably, the biggest confusion with 'recursion' is yet to be described. In computer science, on the procedural level, recursion denotes the syntactic property that a procedure definition refers to the procedure itself (Abelson et al. 1996). In Chomsky's (1956, 1964 [1957], 1975) phrase structure grammar, recursion is a property of rewrite rules (all that is on the left side of the rewrite arrow repeats on the right side of the arrow, e.g. A → AB). For some other theorists, recursion is a structural property: a situation where an instance of an item is embedded in another instance of the same item (Heine and Kuteva 2007; Jackendoff and Pinker 2005; Parker 2006; Premack 2004). For clarity, let us call the former recursion and the latter self-embedding.[1] Recursion and self-embedding are logically independent for the following reasons. First, a self-embedded structure (an NP within an NP, a box within a box etc.) does not have to be recursively generated. Jackendoff and Pinker (2005) submit a picture of a rectangular form within another rectangular form as an example of 'recursion in visual grouping'. Obviously, this has no bearing whatever on recursion. It would be outlandish to assume that recursion is necessary to put a box in a box, or for understanding that a box is in a box. Yet, for conspicuous (but nonetheless insufficient) reasons, this assumption is held with syntactic categories like sentence and NP. The reasons are, of course, the recursive rewrite rules of generative grammar (e.g. NP → A NP), and they are insufficient as there are at least three mathematically explicit alternatives to generative grammar (for details, see Pullum and Stolz 2010). Moreover, even if generative grammar with its recursive rewrite rules is the best theory around, the output of recursive rewrite rules can be generated iteratively as well (see "Recursion, iteration, and inductive definition"). Furthermore, iteration is defined as the repeated application of a transformation (Chang and Sederberg 1998; Weisstein 2003), which is something that Chomsky's (1956, p. 113) description of his early transformational version of generative grammar explicitly incoporates: "/—/phrase structure is limited to a kernel of simple sentences from which all other sentences are constructed by repeated transformations". The confusion with recursion can be traced back to Chomsky (1971 [1957], p. 24; 1956, p. 115), who refers to loops as 'recursive devices'. The source that the formalism is taken from refers to the loops as 'circuits' ("a closed series of lines in the graph with all arrows on the lines pointing in the same orientation"—Shannon and Weaver 1964 [1949], p. 47). The circuits pertain to a graphic representation of finite-state Markov chains, and to call them 'recursive' is jumping to the conclusion, as Markov chains do not prescribe an algorithmic realization for the circuits. In Chomsky (1959, p. 143), 'recursive function' is used as a synonym for 'Turing computable function'. Again, this is confusing, as computational equivalence does not imply algorithmic equivalence (which is absent in this case).

Fitch (2010) claims that iterative functions are inadequate for generating center- and/or self-embedding. As an example of the superiority of recursion over iteration, he presents a recursive center-embedding program generating A^nB^n. Below is an iterative pseudocode that does the same:

```
C = ''''                              //evaluate C to the empty string
for i = 0 to n do :                   //loop n times
C = concatenate(''A'', C, ''B'')      //concatenate ''A'', C and ''B'',
                                      //and assign the result to C
```

This is center-embedding: the program embeds *C* between "A" and "B" *n* times, with *i* indicating the depth of embedding in each cycle of the loop. It is true that "recursive functions take their own past output as their next input" (Fitch 2010, p. 75) but this feature is not unique to recursion—in our above examples, concatenate() coupled with iteration does the same, and so does the addition function in the sequence $a_1 + a_2 = a_3$, $a_3 + a_4 = a_5$, $a_5 + a_6 = a_7$….

[1] Observe that, in Chomsky's (1959) phrase structure grammar, self-embedding has a different, precise technical meaning. In phrase structure grammar, self-embedding entails recursion but not vice versa. A grammar is self-embedding if it contains a derivation A ⇒ xAy, where A is a single non-terminal and {x,y} are non-empty strings (Chomsky 1959, p. 148). For example, S → ASB and {S → ASB, S → AB} are self-embedding grammars that generate …AASBB… and …AABB…, respectively.

Recursion and self-embedding are already in principle very different. Recursion pertains to a process or procedure, self-embedding pertains to a structure. A recursive process or procedure is something that, more often than not, cannot be directly observed. Self-embedding, on the other hand, is usually salient and an offshoot of a cognitive phenomenon we term 'parallel interpretation'. Parallel interpretation is the interpretation of elements of a single communication system and modality at multiple levels[2] (e.g. perceiving a rectangle within a rectangle, detecting a noun phrase within a noun phrase). A crucial difference between parallel interpretation and non-parallel interpretation is that only parallel interpretation allows the same element to be interpreted simultaneously as a type (i.e. category) and as a token (i.e. instance), hence implying additional interpretative correlates not present in the input. The type/token distinction is a precondition for self-embedding, where tokens are embedded under the same type (e.g. NP or clause). An example of parallel interpretation is natural language. Linguistic interpretation is sequential and compounding, merging smaller units that are per se meaningful in the code (Chomsky 1995; Hauser et al. 2002; Studdert-Kennedy 1998). As far as we know, linguistic code is unique among natural communication systems in stipulating semantic compositionality, whereby meaningful units are combined into diversely meaningful higher-order units (e.g. words into phrases, sentences and compound words, phrases into sentences, and higher-order phrases).

As an illustration that self-embedding is possible in parallel interpretation only, consider the following example: the recursive center-embedding rule AB → AABB generates the strings AABB, AAABBB etc. It is impossible to tell by looking at these strings whether their generation procedure (or process) was recursion, iteration, or neither (cf. Corballis 2007a; Fitch 2010, and the example above). Furthermore, it is impossible to tell whether the strings exhibit self-embedding. The definition of self-embedding was "an instance of an item is embedded in another instance of the same item". In other words, "an instance of a type (i.e. a token) is embedded in another instance of the same type (i.e. in another token)". Without any a priori assumptions about the generative mechanism (e.g. stipulation of a certain phrase structure grammar), it is undecidable whether a string …AABB… is embedded, concatenated, or elementary (assuming that different generative mechanisms may allow for different elementary strings).

[2] Parallel interpretation may be closely related to multitasking (consciously managing two or more operations)—an ability admittedly unique to humans (Donald 1998).

Embedding

Embedding is a situation where an item is embedded in any item (with infinity not implied). Embedding is logically independent from recursion (i.e. there can be one without the other). First, embedding does not have to be generated by a recursive rule. It can be created iteratively or by any other function with relevant output. Second, a recursive process or procedure does not have to yield (relevant) output. Assuming that we cannot witness a recursive process or procedure in situ (e.g. in the brain), two conditions must be met for attesting it: (1) it must generate output, and (2) there must be a one-to-one correspondence between the values of the recursive procedure and its output. Logically, self-embedding is a situation where an instance of an item is embedded in another instance of the same item (with infinity not implied); thus, self-embedding is a proper subset of embedding. The fact that embedding is hierarchical has frequently raised speculations about a putative underlying recursive process or procedure (or more unfortunately, resulted in confusing embedding with recursion). By definition, self-embedding does not imply recursively generated embedded output. As explained above, a hierarchical or embedded structure is insufficient to decide on its generative mechanism. Hence, self-embedding can be unambiguously equated with recursion only in the context of recursive rules, the output of which can be organized into a containment hierarchy. As Suzuki et al. correctly remark in discussing humpback whale song, "Hierarchical grammars may be efficiently represented using recursion, although recursion is not necessarily implied by hierarchy" (Suzuki et al. 2006, p. 1863).

Recursion in non-human communication?

Following the widely cited paper by Hauser et al. (2002), there have been some claims as to the possibility of recursion in non-human communication (Marcus 2006; Okanoya 2007; Watanabe and Huber 2006). However, the study that the claims refer to does not imply recursion in non-humans. In the experiments by Gentner et al. (2006), European starlings learned to classify the sequences from the recursive, center-embedding grammar AB → AABB with respect to finite-state grammar and agrammatical sequences (with A corresponding to 'rattle' and B to 'warble' motif from their song repertoire). As explained in "Recursion and self-embedding: a confusion", although the grammar AB → AABB is recursive, its output strings, when judged by themselves (as in the experiment), exhibit neither recursion nor self-embedding. The starlings learned to differentiate the AB → AABB sequences AABB, AAABBB etc. from the others like AAAA, BBBB, ABBA, BAAB, ABABAB, and ABABABAB We argue that this

distinction could have been made by pattern recognition mechanisms capable of subitizing the number of identical adjacent symbols up to 4 and rejecting all strings where the numbers for A and B differ.[3] Given the complexity of their song and the fact that they are excellent mimics (Chaiken et al. 1994), it is axiomatic that European starlings have a capacity for auditory pattern memorization and recognition (after all, they would have to recognize the song of their conspecifics from all others). Since the number of consecutive As or Bs never exceeded 4, the starlings probably relied on their "counting" ability (Corballis 2007b). As Köhler's classic experiments have shown, pigeons and jackdaws can "count" a simultaneously or successively presented set up to 6, and ravens and parrots up to 7 elements (Köhler 1956). Thus, there are two plausible cues that the birds could have used: rhythm and number. Given this, it would be surprising if nine out of eleven birds would not have learned to reliably discriminate AABB, AAABBB etc. from other sequences after 10,000–50,000 trials. As the birds were only trained on AABB vs. ABAB, we need an explanation why their response to the novel sequence AAABBB was different from that to AABBBB. We suggest that pattern recognition and/or counting are more simple and plausible explanations than learning a recursive rule. While it is true that a more generalized pattern or rhythm recognizer will have trouble rejecting A^*B^*, an exact one will reject all except A^nB^n. In addition, though a sufficient degree of exactness is not very high for strings of up to 4 As and 4 Bs, the birds still made mistakes (albeit their classification was significantly better than chance performance). If the birds had learned the recursive rule, there would have been only chance mistakes, if any. After all, learning a rule can be differentiated from pattern recognition only by extremely stable and nearly perfect performance in situations where new patterns are introduced.

We submit two general points about attesting self-embedding and recursion in communication. First, it is easier to attest self-embedding than recursion. Second, as described in "Recursion and self-embedding: a confusion", self-embedding subsumes parallel interpretation. To our knowledge, neither recursion nor self-embedding has been attested in non-human animal communication. This is in concordance with the observations that no non-human animal communication system known shows evidence of syntactic recursion (Fitch et al. 2005; Hurford 2004).

[3] In a well-known experiment, pigeons were trained to reliably discriminate between novel paintings by Picasso and Monet (Watanabe et al. 1995).

Infinity in natural language and arithmetic competence

The central claim of Hauser et al. (2002) and Chomsky (2010) is that a neurally implemented recursive process introduces infinity to natural language and arithmetic. An example of (potential) infinity in natural language and arithmetic competence is the *knowledge* that one can add 1 to n, append a natural language expression to text or embed clauses indefinitely. Of course, we are incapable of *performing* infinitely in any of these tasks (hence the famous competence/performance distinction—Chomsky 1975, 1995).

In the Minimalist framework (Chomsky 1995), it is not hard to show that if the lexicon is reduced to a single element, then unbounded Merge (an operation that takes a pair of syntactic objects and replaces them with a new combined syntactic object) will yield arithmetic (Chomsky 2010). Chomsky's derivation of neurally implemented recursion for operating on **N** is as follows. A) Any formal definition of the set of natural numbers **N** = {0, 1, 2, 3,…} incorporates recursion by means of the successor function, where 1 = S(0); 2 = S(S(0)) etc. B) We have knowledge of the properties of **N** (i.e. given enough time and space, we can compute the sum of two numbers and distinguish the right from the wrong answer). From premises A and B, he conjectures that neurally implemented recursion is required for operating on **N** (e.g. for adding 4555 to 7884). Thus, we have the following: (1) infinity in natural language and arithmetic competence (to motivate neurally implemented recursive process in the first place), (2) neurally implemented recursion is required to operate on **N**, and (3) **N** is an offshoot of the language faculty. From these premises, it follows that (4) neurally implemented recursion underlies both **N** and natural language.

On the face of it, the above argumentation for neurally implemented recursion is consistent and logically sound. However, the points (1)–(4) themselves are rather shaky, and in "Arithmetic performance" we argue in more detail that neurally implemented recursion is not necessary to operate on **N**. Here, we make only some general observations. (4) is the only feasible conjecture from premises {(1), (2), (3)} only insofar as we assume that an inputting brain function can be formally and straightforwardly derived from its final output (in this case, supposedly, infinity in natural language and arithmetic competence). For separate reasons, this assumption seems erroneous both in this particular case and in principle. Firstly, as will be explained below, infinity in natural language and arithmetic competence reduces to *imagining* infinite embedding or concatenation and thus does not qualify as an output of a recursive process or procedure (as there is no reason to assume that conceptualizing infinity requires recursion). Secondly, it is not clear whether or to what extent some

fractions or levels of the universe comply with the laws of classical logic. These laws are derived from empirical evidence. Thus, they do not necessarily hold for the domains that we have had very limited or no access to—for example, it has been shown that the distributive law breaks down in quantum mechanics (Birkhoff and von Neumann 1936; Putnam 1975). Thirdly, as recursive processes assume serial processing (the product of a recursive function with a certain argument value depends on its product for the previous argument value, except in the base case), their mappability to a neurobiological interface is problematic since the brain integrates information via massively parallel circuits (Felleman and Van Essen 1991).[4] The evidence available on brain function so far indicates that, while it may be a computer in a very loose sense, it is radically different from those that we have built (rather analog than digital, with predominantly parallel processing, not computationally equivalent to a universal Turing machine etc.), and vastly more complex (Douglas and Martin 2007; Leng and Ludwig 2006). We suspect that, in the brain, there are too many mediating layers between the final output and the input procedure for the latter to be derived from the former with any certainty. Thus, it is not clear whether it is correct to derive a neurally implemented recursive process from premises A and B above. In particular, we suggest that it is not clear whether monotonicity of entailment holds in deriving a neurally implemented recursive process from A and B above—i.e. it is not clear that if neurally implemented recursive process infers from A,B, it would infer from A,B,X, where X is an additional premise stipulated by brain function. A and B yield the assertion that exact arithmetic (premise B) stems from conceptualizing the set of natural numbers as defined by the successor function (premise A). However, premise A is not necessary for arithmetic. Computers implement arithmetic by dedicated logic circuits that combine bits from two binary inputs to yield the result (von Neumann 1948). Children, on the other hand, do arithmetic initially by using their limited ability to count and later acquire rules for reducing calculations with large decimal numbers to intermediate steps involving smaller ones. Ultimately, all efforts to verify our calculations must resort to counting which is, in principle, rather an iterative than a recursive process (see "Defining recursion").

The concept of neurally implemented recursion is largely motivated by the 'discrete infinity' property of natural language (Chomsky 1995; Hauser et al. 2002). In fact, the whole distinction between the broad and the narrow language faculties, as originally proposed by Hauser et al. (2002), can be derived from this property. However, the maximum possible natural language "corpus"—everything that has ever been and will be processed—is not infinite but a finite, just physically uncountable set. We propose that this is precisely the nature of language as it should be accounted for. This substantial correction (physically uncountable finity instead of infinity) is suggested for the sake of unambiguity and exactitude. Physically uncountable sets can be finite or infinite. Set-theoretically, potential and actual infinity (Moore 1990) are proper subsets of physical uncountability. The evidence that Hauser et al. (2002, p. 1571) submit for discrete infinity covers also physically uncountable finity: "There is no longest sentence (any candidate sentence can be trumped by/—/ embedding it in "Mary thinks that …"). It would be a contradiction to assume that the size of a finite, physically uncountable array can be compared to the size of all others, or that such array can be embedded (the mere fact that we can imagine embedding such an array does not account for its capacity of being embedded). "There is no non-arbitrary upper bound to sentence length." This is as true for an infinite as it is for a finite, physically uncountable array.

We conclude with the following points. 1. There is no data available on brain function that would implicate neurally implemented recursion. 2. It is not at all clear that an input procedure in the brain can be formally and straightforwardly derived from its final output with any certainty (much less by the laws of classical logic, as in the Chomskyan derivation of neurally implemented recursion). 3. The fact that natural language and arithmetic processing is by definition finite (although potentially physically uncountable) downplays the motivation for neurally implemented recursive process. 4. As recursion is primarily used for defining infinite sets, the only motivation for neurally implemented recursion is infinity in natural language and arithmetic competence. As explained above, infinity in natural language and arithmetic competence reduces to imagining, not processing infinite embedding or concatenation. A recursive process or procedure is not needed to account for this. In fact, there is no compelling evidence that the brain behaves in any recognizable way like the models of computation we use in formal logic.

Embedding depth in natural language

In this section, we focus on the parameter of syntactic embedding that is most relevant to the present discussion, viz., the depth of the embedding. On the face of it, the depth of syntactic (and, correspondingly, conceptual) embedding in natural language can be physically uncountable. Although there are no such examples in normal usage, we can artificially create one, e.g. [[[[[*John's*] *mother's*] *dog's*] *parents'*] *ears'*]… Language-specific differences exist, however. Everett (2005), for

[4] The same problem applies to iterative processes as well.

example, has hypothesized that Pirahã lacks self-embedding altogether.

It is well known that embedding depth in normal natural language usage is very limited, with the exact limit depending on the language and construction. For English, Finnish, French, German, Latin, Swedish, and Danish, the maximal depth of initial clausal embedding is two, and the maximal depth of center-embedding is three[5] (Karlsson 2007a,b). Note that these are 'soft constraints' exhibited in the corpora of the respective languages. It has been argued that a variety of center-embedding, 'double object relativization', exemplified by the sentence [*the mouse* [*the cat* [*the dog chased*] *caught*] *squealed*], though OK in theory, is frequently judged as unacceptable beyond the first level of embedding (Christiansen and Chater 2003; Corballis 2007a; Johansson 2006; Karlsson 2007a). However, double object relativizations with first-/second-person pronouns in the innermost subject position are less complex than the same sentences with other NPs in that position (Warren and Gibson 2002, p. 85). While the depth of left- and right-embedding[6] is sometimes thought to be constrained by performance considerations only (time, communicative payoff etc.), there is psycholinguistic evidence that from a certain depth onward the parsing algorithm for such sentences is not embedding but concatenation, i.e. though the logical structure of the sentence presumes embedding, it is processed like concatenation (Christiansen 1992; Karlsson 2007a).

In sum, it is sensible to conclude that embedding in natural language processing is restricted to a maximum of 3–7 levels at a time. The reason why embedding appears to be restricted to 3–7 levels at a time could be related to constraints imposed upon the primary memory. According to Unsworth and Engle (2007), primary memory is defined as the part of working memory which serves to maintain a distinct number of separate representations active for ongoing processing by means of continued allocation of attention. Primary memory is thought to have an upper bound of approximately four items (Cowan 2001; Unsworth and Engle 2007).

Arithmetic performance

The set of natural numbers can be given a primitive recursive definition by a recurring application of the successor function S, where $S(n) = n+1$ ('primitive recursive' means recursive with respect to a single variable). Thus, the recursive definition of natural numbers presupposes the capacity to perform arithmetic operations. But in order to perform arithmetic operations in the first place, one needs to conceptualize the properties of numbers. These properties are given, e.g. in the definition of natural numbers. Accordingly, we find ourselves in a situation where we are able to conceptualize numbers, as evidenced by our capacity to do arithmetic, and we are able to give numbers a recursive definition, but it is not clear whether the concept of number depends on a recursive definition of numbers (on the other hand, a recursive definition of numbers uncontroversially depends on the concept of number).

If recursion were involved in conceptualizing numbers, our brain would execute something like a successor function …S(S(S(S(0))))… for natural numbers and maybe also n*n*n*n… for base-n integer exponents (since we normally use base-10 numeral system, n would normally equal 10). While n*n*n*n… can be coded and implemented recursively as well as iteratively, it is unlikely that anything approximating …S(S(S(S(0))))… or n*n*n*n… would be run in our brains for conceptualizing numbers and performing arithmetic on them. If it were, our arithmetic performance should be significantly better than it tends to be. If, on the other hand, our inferior arithmetic skills are down to general performance limitations and/or penalties for (other) arithmetic operations, there would be no apparent use for running these procedures for our mathematical capacity. The only remaining justification for …S(S(S(S(0))))… and n*n*n*n… would be recursion in the language faculty. However, for this concession to make sense, there would first have to be some evidence for recursion in the language faculty. As we have argued at length above, at present we have merely conjectures built on invalid premises (cf. "Infinity in natural language and arithmetic competence").

How is arithmetic performed, then? The set of natural numbers exhibits regularities by which a complex arithmetic task can be reduced to a number of simpler tasks involving small numbers. When calculating 54 + 68, people can rely on the following strategy: they add 5 to 6, append 0 to the sum, memorize the result (110), add 4 to 8, and add the sum (12) to the memorized result (e.g. by adding 1 to 11, appending 0 to the sum, and adding 2 to

[5] An example of initial clausal embedding: [[*If* [*when I'm 38*] *Metallica ends*] *I don't think*…] (Karlsson 2007b, p. 112). An example of center-embedding: [*A person* [*who,* [*when riding a cycle,* [*not being a motor vehicle,*] *on a road or other public place,*] *is unfit to ride through drinks or drugs,*] *shall be guilty of an offence*.] (British Road Traffic Act, 1972; Karlsson 2007a, p. 372). Gibson (1998, 2000) has shown that the acceptability of such constructions depends not only on embedding depth but also on the locality and characteristics of the discourse events and participants introduced in the constructions.

[6] An example of left-embedding (initial-embedding or left-branching): [[[[[[[*John's*] *mother's*] *dog's*] *parents'*] *ears'*] *hairs'*] *follicles*.]. An example of right-embedding (final-embedding or right-branching): [*This is* [*to confirm* [*that I would like* [*to enquire* [*whether it would be possible* [*to apply this technique on humans* [*as we have run out of animals*.]]]]]]].

120). Theoretically, it is still possible that 2, 4, 5, 6, and 8 are generated recursively, but the upshot is the following: the power of recursion, required to generate a potentially infinite set, is redundant in accounting for human arithmetic performance.

It is easy to demonstrate that conceptualizing a principle (recursion) for producing a pattern of output (self-embedding) does not entail (1) that the principle is necessary for producing the pattern (as self-embedding can be also produced iteratively or by any other function with relevant output—see "Embedding") and (2) that the principle itself must be neurally implemented for us to be able to conceptualize it. For example, we can conceive that all natural numbers are derived from the number 23098 by ±1 operations, i.e. each time we conceptualize a natural number *x* that is less than 23098, we subtract 1 from 23098 until we get x, and each time we conceptualize a natural number y that is greater than 23098, we add 1 from 23098 until we get y. We can conceive this principle. Does it follow that the "23098 ±1" principle must be neurally implemented for us to be able to conceive it in the first place? Surely not. Observe that the situation with the "23098 ±1" principle is similar to the recursive one: we can conceptualize the principles but both are at odds with human arithmetic performance. To circumvent the latter problem in the neurally implemented recursion hypothesis, the competence/performance distinction has been called into effect. Unquestionably, the competence/performance distinction raises a non-parsimonious psychological duality as to the conceptualization of relevant syntactic and arithmetic properties—we can conceptualize that the properties are given to us by a recursive principle, but we do not seem to follow the principle neither in linguistic nor arithmetic processing/performance. Furthermore, it seems dubious to explain the apparent discontinuity between competence and performance in arithmetic and language by limitations in primary memory—what potential advantage could a neurally implemented recursive principle bestow if its effects are subject to so severe constraints?

Conclusion

We conclude with the following points. Computationally, both recursion and iteration allow for finite definitions of infinite sets. Moreover, iterative solutions are frequently less resource demanding than recursive ones (cf. "Defining recursion"). Second, there are two logically independent notions of recursiveness in linguistics: (A) a recursive procedure or rule (Chomsky 1956, 1964 [1957], 1975), and (B) an instance of an item embedded in another instance of the same item (Jackendoff and Pinker 2005; Parker 2006; Premack 2004). As only (A) is compatible with recursion in mathematical logic, (B) is comparatively recent and synonymous with self-embedding, and (A) and (B) are logically independent, we conclude that (B) is an instance of conflating and confusing self-embedding with recursion. Third, since recursion is a property of certain phrase structure rules and not of terminal strings generated by them, Gentner et al.'s (2006) study does not attest recursion in non-human communication. Furthermore, the technical preciseness of the notion of recursion makes it next to impossible to find evidence for it in the brain by the present-day methods, and there is no reason to assume neurally implemented recursion by default (cf. below). Fourth, contrary to Chomsky and colleagues (Chomsky 1995; Hauser et al. 2002), we argue that a property of natural language is not discrete infinity but physically uncountable finity. Fifth, we reject the received opinion, articulated by Chomsky et al. (Chomsky 2010; Fitch et al. 2005; Hauser et al. 2002), that neurally implemented recursion is necessary to explain natural language and arithmetic competence and performance. The only motivation for neurally implemented recursion is infinity in natural language and arithmetic competence (e.g. the knowledge that one can add 1 to n, append a natural language expression to text or embed clauses indefinitely). We claim that infinity in natural language and arithmetic competence reduces to imagining infinite embedding or concatenation, which is completely independent from an algorithmic capacity for infinite processing, and hence, completely independent from neurally implemented recursion or iteration. As the only purported motivators for recursion in human cognitive processing are natural language and arithmetic, the need for recursion in human cognitive processing is effectively downplayed. Crucially, as recursive and iterative processes cannot be distinguished by their input/output, neurally implemented recursive processes cannot be discerned from the iterative ones with present-day methods. Moreover, with all evidence for neurally implemented recursive and/or iterative processing, the more parsimonious (i.e. the default) assumption should be iteration, which is a generally simpler and less resource demanding process than recursion. In sum, there is no infinity in natural language and arithmetic processing, but even if there were, iteration would be sufficient for generating it.

Acknowledgments We thank Noam Chomsky and Margus Niitsoo for extremely helpful, thorough and critical discussions, and Michael Corballis, Panu Raatikainen, Tim Gentner, Haldur Õim, Jaan Valsiner, Märt Muts, and the anonymous reviewers for their comments and suggestions. All remaining mistakes are, of course, our own. Erkki Luuk was supported by the target-financed theme No. 0180078s08, the National Programme for Estonian Language Technology project "Semantic analysis of simple sentences 2", and the European Regional Development Fund through the Estonian Center of Excellence in Computer Science, EXCS.

References

Abelson H, Sussman GJ, Sussman J (1996) Structure and interpretation of computer programs, 2nd edn. MIT, Cambridge

Bickerton D (2009) Recursion: core of complexity or artifact of analysis? In: Givón T, Shibatani M (eds) Syntactic complexity: diachrony, acquisition, neuro-cognition, evolution. John Benjamins, Amsterdam, pp 531–543

Birkhoff G, von Neumann J (1936) The logic of quantum mechanics. Ann Math 37(4):823–843

Chaiken M, Bohner J, Marler P (1994) Repertoire turnover and the timing of song acquisition in European starlings. Anim Behav 128(1–2):26–39

Chang G, Sederberg TW (1998) Over and over again. The Mathematical Association of America, Washington, DC

Chomsky N (1956) Three models for the description of language. IRE Transact Inform Theor 2:113–124

Chomsky N (1959) On certain formal properties of grammars. Inform Control 2:137–167

Chomsky N (1964 [1957]) Syntactic structures. Mouton, The Hague

Chomsky N (1971 [1957]) Syntactic structures. Mouton, The Hague

Chomsky N (1975) The logical structure of linguistic theory. Plenum, New York

Chomsky N (1995) The minimalist program. MIT, Cambridge

Chomsky N (2010) Some simple evo-devo theses: how true might they be for language? In: Larson RK, Yamakido H, Deprez V (eds) Evolution of human language: biolinguistic perspectives. Cambridge University Press, Cambridge, pp 45–61

Christiansen MH (1992) The (non)necessity of recursion in natural language processing. In: Proceedings of the 14th annual conference of the Cognitive Science Society. Cognitive Science Society, Indiana University, Indiana, pp 665–670

Christiansen MH, Chater N (2003) Constituency and recursion in language. In: Arbib MA (ed) The handbook of brain theory and neural networks, 2nd edn. MIT, Cambridge, pp 267–271

Christiansen MH, Chater N (2008) Language as shaped by the brain. Behav Brain Sci 31:489–558. doi:10.1017/S0140525X08004998

Corballis MC (2007a) On phrase-structure and brain responses: a comment on Bahlmann, Gunter, and Friederici (2006). J Cogn Neurosci 19:1581–1583

Corballis MC (2007b) Recursion, language, and starlings. Cogn Sci 31:697–704

Cowan N (2001) The magical number 4 in short-term memory: a reconsideration of mental storage capacity. Behav Brain Sci 24(1):87–114 (discussion 114–185)

Donald M (1998) Mimesis and the executive suite: missing links in language evolution. In: Hurford JR, Studdert-Kennedy M, Knight C (eds) Approaches to the evolution of language: social and cognitive bases. Cambridge University Press, Cambridge

Douglas RJ, Martin KA (2007) Mapping the matrix: the ways of neocortex. Neuron 56(2):226–238. doi:10.1016/j.neuron.2007.10.017

Evans N, Levinson SC (2009) The myth of language universals: language diversity and its importance for cognitive science. Behav Brain Sci 32:429–492. doi:10.1017/S0140525X09990094X

Everett DL (2005) Cultural constraints on grammar and cognition in Pirahã. Curr Anthropol 46(4):621–646

Felleman DJ, Van Essen DC (1991) Distributed hierarchical processing in the primate cerebral cortex. Cereb Cortex 1(1):1–47

Fitch WT (2010) Three meanings of "recursion": key distinctions for biolinguistics. In: Larson RK, Deprez V, Yamakido H (eds) The evolution of human language: biolinguistic perspectives. Cambridge University Press, Cambridge, pp 73–90

Fitch WT, Hauser MD, Chomsky N (2005) The evolution of the language faculty: Clarifications and implications. Cognition 97(2):179–210. doi:10.1016/j.cognition.2005.02.005 (discussion 211–125)

Gentner TQ, Fenn KM, Margoliash D et al (2006) Recursive syntactic pattern learning by songbirds. Nature 440(7088):1204–1207. doi:10.1038/nature04675

Gibson E (1998) Syntactic complexity: locality of syntactic dependencies. Cognition 68(1):1–76

Gibson E (2000) The dependency locality theory: a distance-based theory of linguistic complexity. In: Miyashita Y, Marantz AP, O'Neil W (eds) Image, language, brain. MIT, Cambridge, pp 95–126

Hauser MD, Chomsky N, Fitch WT (2002) The faculty of language: what is it, who has it, and how did it evolve? Science 298(5598):1569–1579. doi:10.1126/science.298.5598.1569

Heine B, Kuteva T (2007) The genesis of grammar: a reconstruction. Oxford University Press, New York

Hurford JR (2004) Human uniqueness, learned symbols and recursive thought. Eur Rev 12(4):551–565

Jackendoff R, Pinker S (2005) The nature of the language faculty and its implications for evolution of language (reply to Fitch, Hauser, and Chomsky). Cognition 97(2):211–225. doi:10.1016/j.cognition.2005.04.006

Johansson S (2006) Working backwards from modern language to proto-grammar. In: The evolution of language: proceedings of the 6th international conference on the evolution of language (evolang6). World Scientific, Singapore, pp 160–167

Karlsson F (2007a) Constraints on multiple center-embedding of clauses. J Linguist 43(2):365–392. doi:10.1017/S0022226707004616

Karlsson F (2007b) Constraints on multiple initial embedding of clauses. Int J Corpus Linguist 12(1):107–118. doi:10.1075/ijcl.12.1.07kar

Köhler O (1956) The ability of birds to "count". In: Newman JR (ed) The world of mathematics, vol 1. Simon and Schuster, New York

Leng G, Ludwig M (2006) Jacques Benoit lecture: information processing in the hypothalamus: peptides and analogue computation. J Neuroendocrinol 18(6):379–392. doi:10.1111/j.1365-2826.2006.01428.x

Lieberman P (2008) Cortico-striatal-cortical neural circuits, reiteration, and the "Narrow language faculty". Behav Brain Sci 31:527–528

Lieberman P (2010) The creative capacity of language, in what manner is it unique, who had it? In: Larson RK, Deprez V, Yamakido H (eds) The evolution of human language: biolinguistic perspectives. Cambridge University Press, Cambridge, pp 163–175

Marcus GF (2006) Language: startling starlings. Nature 440(7088):1117–1118

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

Minsky M (1972) Computation: finite and infinite machines. Prentice-Hall International, London

Moore AW (1990) The infinite. Routledge, London

Okanoya K (2007) Language evolution and an emergent property. Curr Opin Neurobiol 17(2):271–276. doi:10.1016/j.conb.2007.03.011

Parker AR (2006) Evolving the narrow language faculty: was recursion the pivotal step? In: The evolution of language: proceedings of the 6th international conference on the evolution of language. World Scientific, Singapore, pp 239–246

Pinker S, Jackendoff R (2005) The faculty of language: what's special about it? Cognition 95(2):201–236. doi:10.1016/j.cognition.2004.08.004

Premack D (2004) Psychology. Is language the key to human intelligence? Science 303(5656):318–320. doi:10.1126/science.1093993

Premack D (2007) Human and animal cognition: Continuity and discontinuity. Proc Natl Acad Sci USA 104(35):13861–13867. doi:10.1073/pnas.0706147104

Pullum GK, Stolz BC (2010) Recursion and the infinitude claim. In: Hulst Hvd (ed) Recursion in human language (studies in generative grammar 104). Mouton de Gruyter, Berlin, pp 113–138

Putnam H (1975) Mathematics, matter, and method. Cambridge University Press, London

Rogers H Jr (1987) The theory of recursive functions and effective computability. MIT, Cambridge

Shannon CE, Weaver W (1964 [1949]) The mathematical theory of communication. The University of Illinois Press, Urbana

Sipser M (1997) Introduction to the theory of computation. PWS Publishing Company, Boston

Studdert-Kennedy M (1998) The particulate origins of language generativity: from syllable to gesture. In: Hurford JR, Studdert-Kennedy M, Knight C (eds) Approaches to the evolution of language: social and cognitive bases. Cambridge University Press, Cambridge, pp 202–221

Suzuki R, Buck JR, Tyack PL (2006) Information entropy of humpback whale songs. J Acoust Soc Am 119(3):1849–1866. doi:10.1121/1.424990

Tomalin M (2007) Reconsidering recursion in syntactic theory. Lingua 117:1784–1800. doi:10.1016/j.lingua.2006.11.001

Unsworth N, Engle RW (2007) The nature of individual differences in working memory capacity: active maintenance in primary memory and controlled search from secondary memory. Psychol Rev 114(1):104–132

von Neumann J (1948) The computer and the brain. Yale University Press, New Haven

Warren T, Gibson E (2002) The influence of referential processing on sentence complexity. Cognition 85:79–112

Watanabe S, Huber L (2006) Animal logics: decisions in the absence of human language. Anim Cogn 9(4):235–245

Watanabe S, Sakamoto J, Wakita M (1995) Pigeons' discrimination of paintings by Monet and Picasso. J Exp Anal Behav 63(2):165–174

Weisstein EW (2003) CRC concise encyclopedia of mathematics, 2nd edn. Chapman & Hall/CRC, Boca Raton