Natural recursion doesn't work that way: Automata in planning and syntax

Cem Bozsahin

Abstract Natural recursion in syntax is recursion by linguistic value, which is not syntactic in nature but semantic. Syntax-specific recursion is not recursion by name as the term is understood in theoretical computer science. Recursion by name is probably not natural because of its infinite typeability. Natural recursion, or recursion by value, is not species-specific. Human recursion is not syntax-specific. The values on which it operates are most likely domain-specific, including those for syntax. Syntax seems to require no more (and no less) than the resource management mechanisms of an embedded push-down automaton (EPDA). We can conceive EPDA as a common automata-theoretic substrate for syntax, collaborative planning, i-intentions, and we-intentions. They manifest the same kind of dependencies. Therefore, syntactic uniqueness arguments for human behavior can be better explained if we conceive automata-constrained recursion as the most unique human capacity for cognitive processes.

1 Introduction

One aspect of theoretical computer science that is useful in AI and cognitive science is in making ideas about computing explicit, independent of whether we are computationalist, cognitivist, connectionist, dynamicist, or an agnostic modeler.

One such concept in need of disambiguated use in cognitive science and linguistics is recursion. A one-time conference was dedicated solely to the discussion of the role of recursion in language and cognition (Speas and Roeper, 2009). Current work touches on several issues addressed there, such as its role in planning and syntax, and on lack of recursion in the lexicon (which is only true for a certain kind of recursion). The critical issue, I believe, is lack of agreement in what we think we are observing in the processes that are called recursive.

The need for agreement arises because very strong empirical claims have been made about recursion's role and its mechanism, such as that of Hauser et al (2002), Fitch et al (2005), where *syntactic* recursion is considered the most unique human capacity, the so-called core computational mechanism in "narrow syntax." The claim

Cem Bozsahin
Cognitive Science Department, The Informatics Institute, Middle East Technical University, Ankara e-mail: bozsahin@metu.edu.tr

follows Chomsky's recent theorizing, in particular the Minimalist Program (Chomsky, 1995, 2005), which puts a recursive/cyclic merger at its core, not only as a theoretical device but also as an operation of the mind.

We can conceive this syntax-based (and language-centered) argument about cognition in at least two ways. In its first sense we can take the phrase *syntactic recursion* to mean recursion in syntax, which seems to be everybody's assumption,[1] therefore not expected to be problematic. In the second sense we can take it to mean reentrant piece of knowledge, known as recursion by name (or label) in computer science.[2] As will be evident shortly, these two aspects are not the same when we take formal semantics and formal definitions of recursion into account.

The current paper aims to show that either conception of syntactic recursion poses problems for the narrow claim of narrow syntax, and to the so-called generative enterprise (Huybregts and van Riemsdijk, 1982), which claim that syntactic recursion is the unique human capacity. The most uniquely human capacity may be recursion of a certain kind, but it is not limited to recursion in syntax, and it is certainly not syntactic recursion in the second sense above, therefore the conjecture of Chomsky and his colleagues is probably too strong and premature.

The following arguments are made in the paper. The last point is raised as a question. Some of these arguments are quite well-known. I will be explicit about them in the text.

(I) a. Natural recursion in syntax, or recursion by linguistic value, is not syntactic in nature but semantic.
b. Syntax-specific recursion is not recursion by name as the term is understood in AI and theoretical computer science.
c. Recursion by name is probably not natural.
d. Natural recursion, or recursion by value, is not species-specific.
e. Human recursion is not syntax-specific, although the values it operates on are most likely domain-specific, including those for syntax.
f. Syntax seems to require no more (and no less) than the resource management mechanisms of an embedded push-down automaton (EPDA).

---

[1] All but one, that is. Everett 2005 argues that recursion is not a fact for all languages. That may be true, but the fact remains that some languages do have it, and all languages are equally likely to be acquirable. See Nevins et al 2009, Bozsahin 2012 for some criticism of Everett, and his response to some of the criticisms (Everett, 2009). Even when syntactic recursion is not attested, there seems little doubt that semantic recursion, or recursion by value, is common for all humans, e.g. the ability to think where thinker is agent and thinkee is another thought of same type, manifested in English with complement clauses such as *I think she thinks you like me*. But, it can be expressed nonrecursively as well: *I think of the following: she thinks of it; it being that you like me*. We shall have a closer look at such syntactic, semantic and anaphoric differences in recursive thoughts.

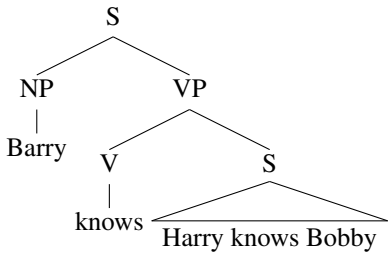
[2] The name is apt because, as lambda-calculus has shown us, reentrant knowledge *can* be captured without names if we want to, and that the solution comes with a price (more on that later). In current work, the term *recursion by name* (or label) is taken in its technical sense in computer science. Confusion will arise when we see the same term in linguistics, for example most recently in Chomsky 2013, where use of the same label in a recursive merger refers to the term 'label' in a different sense, to occurrence of a value.

g. We can conceive EPDA as a common automata-theoretic substrate for syntax, collaborative planning, i-intentions, and we-intentions (Searle, 1990).
h. The most unique human capacity appears to be the use of recursion with a stack of stacks. Arguments from evolution are needed to see whether planning, syntax or something else might emerge as its first manifestation.

2 Recursion by value is semantics with a syntactic label

The kind of recursion manifested in language is exemplified in (2), where a sentence (S) contains another sentence as a complement.

(2)

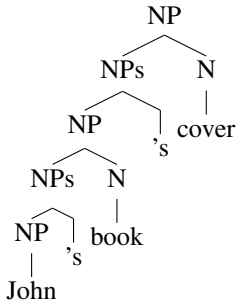


It can be applied unboundedly: *I think you think Barry knows Harry knows Bobby*, etc. The natural capture of this behavior in linguistics is by drawing trees, as above.

It is the property of a class of verbs such as *think, know, ask, claim* that they take such complements. This behavior is constrained by language-particular syntactic properties interacting in complex ways with the argument structure (i.e. semantics) of the event/action/state denoted by the predicate.[3] For example, the English verb *hit* cannot take such complements: **John hits (that) Barry drags Bobby*.

Recursion is possible in the nominal domain as well. For example, a fragment of *John's book's cover's colour* is shown below.

(3)



All the grammatical examples above are recursion by value, where *another* instance (i.e. value) of a predicate is taken as an argument of the predicate. For example, *know* takes a knower and a knowee, and the knowee is another predicate: there

[3] Some comprehensive attempts in linguistics in accounting for the interaction are Grimshaw 1990, Manning 1996, Hale and Keyser 2002.

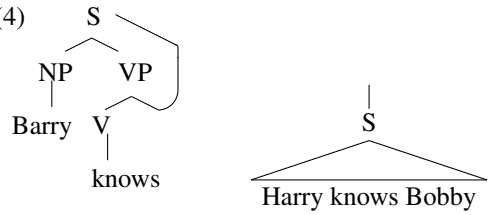
are two acts of knowing in (2), not one, as the use of same label might suggest, and by two different subjects. The same observations apply to two different possessors and their possession in (3).

The constraints on syntactic values—a better term might be *syntacticized values*—such as S and NPs are semantic in nature, as the distinction between *know* and *hit* shows. The same can be said about the possessive construction: it needs the syntactic correlate of a participant and a property or another participant.

3 Syntax-specific recursion is not recursion by name

It follows that the structure below cannot be the right one for *Barry knows Harry knows Bobby*; compare it with the one in (2). In this interpretation, *Harry knows Bobby* would be the base case of recursion by name, as shown separately.

(4)



There is one knower in this structure too, and one knowee, but, unlike (2), it points back to the root predicate-argument structure of the word *know*. Both must be *repeated* at every iteration until the base case (the nonrecursive case written on the right) applies to stop the recursion, giving rise to examples such as *Barry knows Barry knows Barry knows Harry knows Bobby*.

As mentioned, it would be tempting to think of this structure as a generalization of (2), by which we would assume (2) to be the base case of (4) without recursion: *Barry knows Harry knows Bobby*. However, this proposal could not adequately capture the speaker's intuition, that knower can know a knowee, and that, from this she would not infer that knowee's knower must be the same as the knower next level up if multiple embeddings are involved. Neither would she conclude that knower's knowee must be the same predicate until recursion stops, for example *Barry knows John claims Harry knows Bobby*, which is not captured by (4).

It is precisely for this reason that Lexicalized Tree-Adjoining Grammar (LTAG; Joshi and Schabes, 1992) represents the reentrancy implied by (2) not as (4) but as an S tree dominating another S tree, one with a special operation of adjunction rather than substitution. Because it is *another* tree, LTAG captures the right semantics of (2). Generative grammar assumes two trees as well, but makes no such combinatory distinction. Therefore, it is susceptible to recursion by name vs. value arguments.

The structure in (4) is precisely what is called recursion by name in computer science, considered to be a special form of reentrancy. As the preceding argument shows, it is not the same as recursion by value.

We can have a look at formal definitions of recursion, and also at some recursive definitions, to see what is at stake in deciding what kind of recursion is involved.

Below are two different definitions of a potentially recursive data structure, the tree, from Knuth (1968: 314).

(5) a. Tree: (i) a node called *root* is a tree, denoted as T(root). (ii) The subtrees of a tree T, T($T_1, T_2, \cdots T_m$), are partitioned into $T_1, T_2, \cdots T_m$, where each T_i is a tree.
b. Tree: Any tree is a collection of *nested sets*. A collection of non-empty sets is nested if, given any pair X, Y of the sets, either $X \subseteq Y$ or $X \supseteq Y$ or X and Y are disjoint.

The first one is a recursive definition. The second one is not. Knuth shows that they are extensionally equivalent. This is a sign that a definition using recursion by name such as (5a) can be avoided if it is not truly necessary.

It may not be necessary, but is it adequate? The answer depends on what we are studying. One striking discovery in mathematics was that recursion by name (reentrancy) can be written without names or labels, using for example paradoxical or fixpoint combinators. If we define the combinators as (6a–b), we get the characteristic equation of recursive behavior in (6c–d).

(6) a. $\mathbf{Y} \stackrel{\text{def}}{=} \lambda h.(\lambda x.h(x\,x))(\lambda x.h(x\,x))$ Curry and Feys (1958)
b. $\mathbf{U} \stackrel{\text{def}}{=} (\lambda x\lambda y.y(xxy))(\lambda x\lambda y.y(xxy))$ Turing (1937)
c. $\mathbf{Y}h = h\,(\mathbf{Y}h) = h\,(h\,(\mathbf{Y}h)) = \cdots$
d. $\mathbf{U}h = h\,(\mathbf{U}h) = h\,(h\,(\mathbf{U}h)) = \cdots$

Notice that neither **Y** nor **U** are recursive definitions, yet they capture recursion by name. (Incidentally, this is the foundation for compiling functional programming languages, almost all of which are based on lambda calculus. They are all Turing-complete because of this reason.)

The conversion from reentrant (named) recursion to nameless recursion is quite instructive about the powers of recursion by name. Consider the recursive definition of Fibonacci numbers in (7a). It is shown in one piece in (7b), which is then turned into nameless recursion by a series of equivalences in (7c). Notice that h is not recursive by name (f is now a bound variable, which in principle can be eliminated). Its recursion is handled by **Y**.

(7) a. $fib(n) = fib(n-1) + fib(n-2)$ $fib(0) = 0, fib(1) = 1$
b. Let $fib = \lambda n.$if $(n == 0)$ 0 else if $(n == 1)$ 1 else $fib(n-1) + fib(n-2)$
c. Let $h = \lambda f.\lambda n.$if $(n == 0)$ 0 else if $(n == 1)$ 1 else $f(n-1) + f(n-2)$
Then $fib = h\ fib$ because $fib\ n = h\ fib\ n$, $\forall n \geq 0$
And $fib = \mathbf{Y}h$ because $fib\ n = \mathbf{Y}hn, \forall n$, and $\mathbf{Y}x = x(\mathbf{Y}x), \forall x$

But this solution comes with a price. **Y** and **U** are not finitely typeable, therefore their solution space cannot be enumerated. Function h is finitely typeable, but fib is not just h but $\mathbf{Y}h$, which is not finitely typeable.

This result seems to fly in the face of the fact that *know-, think*-like verbs, and possession-like predicates, are lexical items, therefore they must be finitely typeable and representable. In other words, capturing the meaning of *know* by the formula

Y*know′* or **U***know′*, where *know′* is the meaning of *know*, could not stand in for the native speaker's understanding of *know*. Therefore, it is not adequate to use recursion by name in any form to represent competent knowledge of words. Because that kind of knowledge in words is the building block of meaning for syntactic trees, where the meaning of a phrase is combined from the meaning of the parts and the way they are combined, it is not adequate to use recursion by name for syntactic trees of natural strings of words either.

In summary, we have two kinds of evidence that recursion in syntax is not recursion by name, or recursive reentrancy. One is theoretical, as just seen. The other one is empirical, as argued after the example in (4).

4 Recursion by name is probably not natural

Is recursion by name good for anything? It is indeed. The point is slightly tangential to the purpose of current work, but it allows us to see that in places where recursion by name is necessary and adequate, it is difficult to see a natural phenomenon.

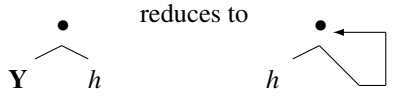
One such domain is programming. With the exception of Fibonacci, the examples we have seen so far are all peripheral recursion, i.e. the recursive value appears on the edge of a tree, which then reiterates, branching on the right edge (2), or the left (3). However, nontail or nonperipheral recursion is possible, and it is not reducible to traversing one periphery of a tree. For example, the pseudo-code below traverses a tree in what is called 'in-order':

```
visit(tree):
  if tree is not nil:
    visit(tree.left)
    print(tree.root)
    visit(tree.right)
end
```

Theory of compiling has shown that we can indeed eliminate such recursion altogether as well, but at the expense of manipulating an auxiliary stack for *every* nonperipheral use of recursion. Such solutions also need elaborate run-time mechanisms to keep track of sequence of computations. It will be clear later that this is fundamentally different than managing an auxiliary stack *once*, per rule, which seems to have natural counterparts.

Thus we either face elimination of recursion by name by fixpoint combinators, which are not finitely typeable, or its elimination by auxiliary devices where every recursive call needs extra stack management. None of these seem natural mechanisms.

It is also worth noting the semantics of recursion in programming:

(8) 

Peyton Jones (1987: 218)

We can see the pseudo-code above as a realization of this syntax and semantics, applied twice. Notice the identity of the general mechanism (8) to the structure we considered to be inadequate for natural language recursion, shown in (4). Unlike natural resources such as words, a piece of code can be reentrant; it can point back to its root (and note that the pseudo-code itself is finitely representable), with the understanding that auxiliary mechanisms (such as activation records of recursive calls) and other conventions take care of the rest. None of these mechanisms or their functional equivalent have been attested in cognition, in language, vision, planning, music, or reasoning.

5 Recursion by value is not species-specific

We can now assume that natural recursion is recursion by value. Given this conception, it is not difficult to see rudiments of recursion in close cousins of ours (if not in other higher animals), especially in planning.

Planning has a long history in AI; see e.g. Ghallab et al (2004) for an extensive coverage of techniques and tools. Collaborative plans and their relation to psychological states have been extensively studied too; see for example Lochbaum (1998), Bratman (1992), Petrick and Bacchus (2002), Steedman and Petrick (2007), Grosz and Kraus (1993), Grosz et al (1999).

The field has devised ingenious ways to capture the act and knowledge of plans as states, search, knowledge representation, and inference. For our purposes, it seems convenient to classify planning in an automata-theoretic way, independent of the aspects above, to highlight its close ties to language.

From this perspective, we can conceive plans at three levels of resource management (with automata-theoretic substrates in parentheses):

(9) a. Reactive planning (finite-state automata—FSA)
    b. Instrumental planning (push-down automata—PDA)
    c. Collaborative planning (embedded PDA—EPDA)

Finite-state plans deliver whatever organization can be afforded by a finite history and non-embedded behavior. This is not much; for example we cannot capture a scenario where separate actions of an agent match step by step, or a case where a step of the plan needs the result of another plan, either by the same planner or by someone else.

We can model such organized behavior to some extent with PDAs. An example from Jaynes in animal cognition is on the mark (he used it to show deceit as a form of animal consciousness): Jaynes (1976: 219) reports of a chimpanzee in captivity filling his mouth with water in order to penalize a not-so-friendly keeper. The chimpanzee coaxes the keeper, and tries to lure him to proximity to spit water in his face. Sometimes the plan fails, and we would expect the chimpanzee not to spit water. (He might spit water, but not for that purpose. Spit therefore means something more

as part of a plan.) His actions depend on how much the keeper conforms to his role as part of the chimpanzee's plan. In this sense it is instrumental planning.

It is worth formalizing some aspects of this planned action to see that at this level of instrumentalizing we are dealing with context-free agent-centered dependencies. Below is a context-free grammar on behalf of the chimpanzee for some potential ways to get what he wants.

(10) S → FillWater LureKeeper Spit
LureKeeper → Coax | Hail
Coax → Stalk Coax | AskBanana

Spitting depends on achieving LureKeeper, which might enter the chimpanzee's plan by perceiving and interpreting the actions of the keeper. If coaxing him fails, we might still have some acts of stalking the keeper by the chimpanzee, but they would presumably not amount to a plan of spitting at him. This is a dependency that—let's say—he himself established as part of the plan.

From an external observer's point of view the plan-action sequences suggested by this grammar may appear to be finite-state, and indeed this grammar captures a finite-state (regular) language. From the chimpanzee's point of view, it is context-free. I will not extend this way of thinking to suggest that chimpanzees (and bonobos and gorillas) are capable of devising grammars that are strictly context-free from both the observer's and the planner's point of view, but the distinction remains that, unlike reactive planning, an instrumental grammar can be context-free in some way.

This way of thinking coincides with a change of mind in cognitive science. Tomasello and Call (1997) had argued earlier that chimpanzees don't have a mind, but they changed their position in Tomasello et al (2003), where the finding is that they might have a mind. Crucially, it depends on being aware of other agents, and of potential results that might suggest alternate courses of action if the other agents' actions do not meet the expectations of the concerned chimpanzee from them. This is semantic recursion, or recursion by value, and it is instrumental.

It is sometimes suggested that strict context-freeness and nontrivial recursion can be observed in birdsong as well, in the sense that they sing phrases that seem to consist of subphrases, in one claim to the extent of beyond context-freeness (Stabler, 2013). It is not clear to me that we are facing semantic *re*cursion here, because it is not clear that this is not a phonological skill (Berwick et al, 2011, 2013). For it to be semantically compositional the internal phrases must have semantics all the way up, which would indeed be recursion by value. Birdsong might have global semantics, such as happiness, gathering, etc., or make use of very simple rules (Van Heijningen et al, 2009).

6 Human recursion

Availability of other kinds of recursion in humans is not contested by Hauser et al (2002), Fitch et al (2005).[4] They acknowledge that spatial reasoning, among other

[4] See Jackendoff and Pinker 2005, Parker 2006 for counterarguments on evolutionary basis of syntactic recursion.

things, is recursive as well, for example *((((the hole) in the tree) in the glade) by the stream)*, from Fitch et al (2005). But, once we have a closer look at nonsyntactic recursion, and subsequently at its striking *computational* similarity to language (§6.2), the Chomskyan argument that what makes syntax unique to humans—which I do not dispute—is recursion in it, weakens. A certain kind of recursion may be the most unique human capacity.

As we have seen in §5, instrumental planning can be taken for granted for humans. For a single agent not collaborating with anyone, but perhaps interacting with others, it is easy to see what Searle (1990) called i-intentions, for example scurrying in the park because of rain, to use an example of Searle's. Everyone in the park might target the same shelter, but these would not be we-intentions but a collection of i-intentions. A collection of i-intentions does not constitute a we-intention, Searle claims. This makes perfect sense when we consider dancing in the rain, which might involve the same set of movements as scurrying in the rain from an external observer's point of view, but we know a collaborative dance when we see one, as a collective intention, therefore behavioral equivalence is not the right criterion. Even if a dancer makes a mistake, we would not equate that with an independent act such as someone failing to reach shelter.

The point of collaborative planning and action is that a collection of individuals may intend to pursue a common goal, although they may serve it by different courses of action. In American football, another of Searle's examples, a quarterback and a runningback may have the same intent and execute the same plan, while carrying out different actions.

We can formally incorporate i-intentions, we-intentions, i-plans, and we-plans. Consider the following grammars as proxies for modeling such behavior arising from an internal mechanism. Let us assume extensionally identical (or equivalent) behavior, i.e. dancing and scurrying involve the same actions and are distinguishable only by the intent. The first grammar below is meant for i-intentions and i-plans, and the second one for we-intentions, we-plans, and i-actions.

(11) $S_i \rightarrow \alpha_i \mid A_i$ where α_i is a plan with a base case A_i (Scurry-in-rain grammar)
$A_1 \rightarrow$ run, and do S_1's work
$\vdots$
$A_n \rightarrow$ run, and do S_n's work

(12) $S_i \rightarrow S'_i[\pi\{S_1,..,S_n\}]$ πx: an ordering of set x (Dance-in-rain grammar)
$S'_i[S_j\ldots] \rightarrow \alpha_j S'_i[\ldots]\beta_j$
$\vdots$
$S'_i[S_i] \rightarrow$ run
and do S_i's work A_i

Here is my convention: the grammars are individuated per person i, with the start symbol S_i. We-intentions first make a note of the 'we', using the first rule in (12). In principle it can be of indeterminate number. These rules make use of the Linear-Indexed Grammar (LIG) convention (and the choice is not accidental; cf.

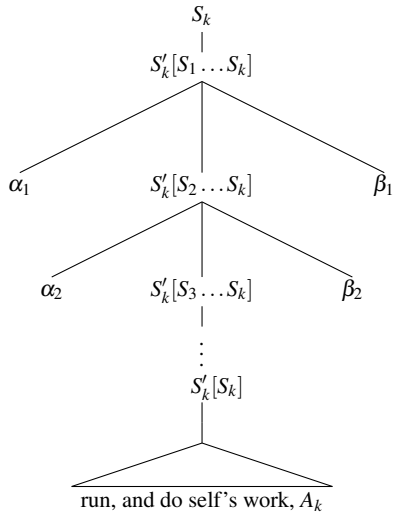


Fig. 1 Participant k's grammar for collaborative dancing in the rain.

subsequent sections). The stack associated with a nonterminal B is denoted $[x \ldots]$ after B, where x is the top. What can enter a stack is planner's decision. Plans and intentions are the righthand sides of rules. Actions and assumptions (or knowledge states) are members of the righthand sides; α and β stand in for contextualizing a participant's action with others, to be differentiated from her own actions, A_i, but related to it structurally, as the unfolding of the mechanism exemplifies for participant k in Fig.1.[5]

Therefore, every participant can go her own way of carrying out the plan, symbolized by the base cases of her own grammar (the '$S'_i[S_i]$' rules), but not behaving incognizant of the overall plan, symbolized by the top rule, or impervious to other participants' plans and actions, symbolized by the left and right contexts of her own actions/states at the bottom.

If somebody's participation goes wrong, say p's, another participant would know this by observing the failure of S_p for the participant, and recovery may be attempted. (Therefore we assume that the grammar is solving a hidden-variable problem, where it is couched between perception and inference from the world, under the guidance of LIG-automata providing the search space. Zettlemoyer and Collins 2005 started explicit modeling of acquisition of this nature for the case of language.)

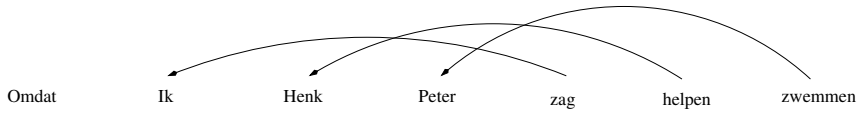
No such mechanism is manifest in an instrumental plan such as (11). Thus we can representationally distinguish we-intentions from i-intentions.

[5] I am not suggesting that (12) is the universal schema for all plans. It is meant to show that collaborative plans may be LIG-serializable. LIG-plan space remains to be worked out. For example, base cases of individuated grammars, the $S'_i[S_i]$ rules, doing running—as part of a dance—and A_i would be by definition LIG-serializable too, but in a manner different than what α and β are intended to capture, viz. contextualized knowledge states of the group constituting the we-intention. A_is may be LIG-realized action sequences, making the whole collection a we-plan.

6.1 Embedded push-down automata for syntactic recursion

It has been known since the work of Shieber (1985) that human languages are not context-free. (Most of the earlier proofs turned out to be problematic. Shieber's proof is the one everyone accepts.) The emerging formal characteristics that are adequate for natural languages are summed up in the mild context-sensitivity hypothesis of Joshi (1985): constant growth of string length, i.e. incremental build-up, polynomial parsability, limited cross-serial dependencies, and proper inclusion of context-freeness.

Cross-serial dependencies are the most challenging automata-theoretic aspect to context-freeness. PDAs cannot handle the following behavior, from Dutch:



'because I saw Henk help Peter swim ...'

However, not all cross-serial dependencies are mildly context-sensitive. Although there are mildly context-sensitive grammars for $\{a^n b^n c^n \mid n \geq 0\}$, which is strictly not context-free, there are no such grammars for $\{www \mid w \in \{a,b,c\}^*\}$, or for $\{w \mid w \in \{a,b,c\}^*$ and $|w|_a{=}|w|_b{=}|w|_c\}$.

The last one, called MIX, symbolizes the result that there is probably no human language which is truly scrambling in every word order if mild context-sensitivity is the upper bound, which was an early conjecture of Joshi (1983), which was recently proven by Kanazawa and Salvati (2012). The first one, the "copy" language, shows that human languages do not need queue automata.[6] Together they show that the automata-theoretic approach to linguistic explanation captures some natural boundaries of human syntax and parsing without further stipulation.[7]

Formally speaking, mild context-sensitivity does not define a formal class but makes explicit some desirable and discernible properties. The least powerful extension of context-freeness is a formal class, called linear-indexed languages (Gazdar, 1988). Lexicalized tree-adjoining grammars and Combinatory Categorial Grammars (CCG; Steedman, 2000) are provably linear-indexed (Joshi et al, 1991).

---

[6] We note that the language $\{ww \mid w \in \{a,b,c\}^*\}$ is fundamentally different than double-copy $\{www \mid w \in \{a,b,c\}^*\}$. The first one allows stack processing. Here is a LIG grammar for it: $S_{[...]} \rightarrow x\, S_{[x...]}, S_{[...]} \rightarrow S'_{[...]}, S'_{[x...]} \rightarrow S'_{[...]}\, x, S'_{[\,]} \rightarrow \varepsilon$, for $x \in \{a,b,c\}$.

[7] Continuing in this way of thinking, we could factor recursion and other dependencies in a grammar, and incorporate word order as a lexically specifiable constraint. It might achieve the welcome result of self-constraining recursion and levels of embedding in parsing: see Joshi 2004:662.

Both LTAG and CCG avoid recursion by name, LTAG by employing adjunction in addition to substitution, and CCG by avoiding any use of paradoxical combinators such as **Y**, or generalized composition. That is how they stay well below Turing equivalence that might otherwise have been achieved because of recursion by name; see also Joshi 1990, Vijay-Shanker and Weir 1993, Bozsahin 2012 for discussion of these aspects. Their restrictiveness (to LIG) becomes their explanatory force.

Linear-indexed languages have an additional property: not only are they polynomially parsable (all MCSLs are), they are efficiently parsable, which means they incur a small polynomial cost in parsing. Dutch and Swiss German data, which exhibit strictly noncontext-free dependencies, can be given a linear-indexed treatment; there are LTAG and CCG grammars for them.[8] Swiss German and Dutch cases can be shown to be abstractly equivalent to $n_1 n_2 n_3 \cdots v_1 v_2 v_3 \cdots$ where n_i is an argument of the verb v_i. In linear-indexed notation we get:

$$\begin{aligned} S_{[\ldots]} &\rightarrow n_i \; S_{[i\ldots]} \\ S_{[\ldots]} &\rightarrow S'_{[\ldots]} \\ S'_{[i\ldots]} &\rightarrow \; S'_{[\ldots]} \; v_i \\ S'_{[\,]} &\rightarrow \varepsilon \end{aligned}$$

The algorithmic substrate of linear-indexed grammars is the Embedded PDA of Vijay-Shanker (1987), Joshi (1990), which is a stack of stacks (and crucially, not two stacks, a system which we know is Turing-equivalent if they can exchange values). Its grammar formalism passes a single stack among the nonterminals to preserve the dependencies (from left to one symbol on the right, hence the term *linear*). The following grammar is for $\{a^n b^n c^n d^n \mid n \geq 0\}$.[9]

$$\begin{aligned} S_{[\ldots]} &\rightarrow a \; S_{[i\ldots]} \; d \\ S_{[\ldots]} &\rightarrow S'_{[\ldots]} \\ S'_{[i\ldots]} &\rightarrow b \; S'_{[\ldots]} \; c \\ S'_{[\,]} &\rightarrow \varepsilon \end{aligned}$$

6.2 Embedded push-down automata for human recursion

The mechanism that was devised to surpass the context-freeness boundary in syntax is the same as the one we need to move from i-intentions to we-intentions, or from instrumental planning to collaborative multi-agent planning. That seems natural given the relation between organized behavior and serializability, which was first observed in psychology. In Karl Lashley's words:

> Temporal integration is not found exclusively in language; the coordination of leg movements in insects, the song of birds, the control of trotting and pacing in a gaited horse, the rat running the maze, the architect designing a house, and the carpenter sawing a board

[8] The Swiss German facts are more direct because the language has overt case marking and more strict word order; see Bozsahin 2012 for a CCG grammar of some Swiss German examples.

[9] Notice that $\{a^n b^n c^n d^n e^n \mid n \geq 0\}$ is not a linear-indexed language, hence such grammars make no use of a linear distance metric, or simple induction from patterns; see Joshi 1983.

> present a problem of sequences of action which cannot be explained in terms of successions of external stimuli. Lashley (1951: 113)

Some internal mechanism appears to be at work, and, from the current perspective, LIGs may be the most explicit proposal for the unified problem of characterizing behaviors that are complex enough to rise above data in unexpected ways compared to other kinds of computations by other species.

This is not a resemblance, or reasoning by analogy. It shows that natural languages and natural plans of humans may reduce to the same class of automata-theoretic resource management. If natural computation is what we seek to understand, there seems to be an identifiable mechanism of its management, with many ways to materialize depending on the nature of categories. This does not put humans with language on a par with singing birds and maze-running rats in terms of complexity of organized behavior, but it helps us to understand what added computational explanation is brought in by identifying a class of automata with these behaviors, and the ensuing kinds of recursion that these species are assumed to be capable of.

7 Discussion

The preceding argument is not a conjecture that humans must have a general problem solving ability, one omnipotent induction machine without resource boundedness, and that language and planning fall under it. Quite the contrary, there are unique language-specific constraints, much of which have been worked out theoretically. (We cannot say the same thing about we-intentions and collaborative planning. But there is one conjecture of this way of thinking: possible plans may be the LIG-serializable ones.) Languages do not differ arbitrarily, modulo their lexicons. And even there we can expect to see some predictability once we clarify the concept of natural recursion, such as finite representability of lexical items, which in effect rules out any use of recursion by name in the lexicon.

And clearly, language cannot work with action categories, or action with linguistic categories, or music with linguistic categories or with visual ones. Perception is not an omnipotent mechanism. What makes the cognitive processes learnable may be the specialized categories, sort of Humean rise above experience. It does not follow that we learn how to combine in each cognitive domain, rather than combine to learn with some specialized categories.

Deacon (1997) argued that language and brain co-evolved. This proposal bears on the claims for a common substrate. Brain areas that are taken over by language, over at least two million years, are related to planning and action sequencing. Jaynes (1976) had a different agenda, to explain consciousness, which he claimed happened much more recently compared to language, but nevertheless tapping onto the same parts of the brain, and onto the same functionality: combinatory competence. In this regard the practice of writing grammars for language and for planning is not just a historical accident or convention. Grammars are hidden-variables, where the

observed form and deduced meaning are hypothesized to be indirectly related by an unobservable grammar.

The automata-theoretic approach to the problem suggests that maybe what we are dealing with is not deduction, or grammar induction in a naive sense, but induction under resource boundedness constraints (thus making an attempt to avoid the problems of induction). Out of a possible space of grammars predicted by the identified substrate of automata, it will carve out those which are maximally consistent with perceived data under the constraints of computational complexity. Probably Approximately Correct (PAC) learning is very relevant in this regard: "Inherent algorithmic complexity appears to set serious limits to the range of concepts that can be learned." Valiant (1984). The class of automata properly identified makes the hypothesis space enumerable for grammars and plans (and for recursion in them), which is one requirement for PAC learning. Finding a hypothesis in polynomial time which is maximally consistent with data is another requirement, and we can maintain P property (for a problem which would be in NP if it is decidable), if we look at likely meanings for words and plans, rather than possible meanings as Quine (1960) did. This is the task of obtaining a grammar by solving a hidden-variable problem, in effect saying that recursion by value is learnable too.

Another purported impediment to PAC learnability of such knowledge is the assumed identity of the data distribution for the sampling of training *and* novel examples, as pointed by Aaronson (2013: 291). However, all that PAC class of learners requires is that the distribution is known, not necessarily *derived* only from early experience. (In this sense, his example of "learning of an *Einstein*" might stray PAC into weird corners of the distribution compared to a mere mortal, which means Nature would have to sample a bit more for him or someone like him to arrive again, but it would sample from the then-current population just like before. In a more recent reassessment of PAC, Valiant 2013 elaborates on Invariance and Learnable Regularity assumptions in relation to natural phenomena such as evolution and mind.)

A PAC-like mechanism can safely depend on recursion by value because of its finite representability and its empirical foothold (after all, it is a *value*). The other alternative, reentrancy, or recursion by name as conceived in theoretical computer science, is difficult to assess naturally. There is something unnatural about it. Empirically, it does not correspond to other natural dependencies, which seem to be resource-sensitive and finitely representable. Theoretically, it can be reduced to nameless recursion, which means reduction to recursion by value by a sequence of base cases, which is not enumerable without them.

The last point is equivalent to being uncomputable, for we currently know no way of computing with transfinite representations.[10] We can compute indices of π indefinitely, but we cannot entertain questions regarding the next number after π. Nor can we ask questions about what happens after a computational process fails to halt, and expect an answer.

[10] Notice that lazy evaluation is not a remedy here. By lazy evaluation, we can represent infinite streams by finite means (Abelson et al, 1985, Watt, 2004), but for that to work infinite streams must be enumerable.

8 Conclusion

From a computer science perspective, natural language syntax does not seem to operate on recursion by name. The kind of dependencies we capture in linguistics when we draw trees of hierarchical structures is recursion by value, which is semantic in nature, but clearly syntacticized. The same can be said about plans, which fall into environment- and object-orientation by affordances (Gibson, 1966), syntactically corresponding to type-raising, and to event-orientation by combinatory composition, which is, syntactically, function composition (Steedman, 2002).

Recursion-by-value assumption is commonplace in all of cognitive science, also assumed by those who insist it is not needed in syntax (see Everett's commentaries after the recursion conference—Speas and Roeper 2009). I believe that Everett's view is not sustainable (footnote 1, also Bozsahin 2012), but its failure will not vindicate Hauser et al (2002), Fitch et al (2005).

Humans appear to be uniquely capable of recursion by value, of the kind that can be afforded by a stack of stacks. Various predictions about syntax and other cognitive processes follow from an automata-theoretic way of thinking about them. Therefore, uniqueness of syntax arguments to humans, which I take to be a fact, can be better explained if we conceive automata-constrained recursion as the most unique human capacity for cognitive processes.

Acknowledgements Thanks to PT-AI reviewers and the audience at Oxford, İstanbul, and Ankara, and to Julian Bradfield, Aravind Joshi, Simon Kirby, Vincent Müller, Umut Özge, Geoffrey Pullum, Aaron Sloman, Mark Steedman, and Language Evolution and Computation Research Unit (LEC) at Edinburgh University, for comments and advice. I am to blame for all errors and for not heeding good advice. This research is supported by the GRAMPLUS project granted to Edinburgh University, EU FP7 Grant #249520.

References

Aaronson S (2013) Why philosophers should care about computational complexity. In: Copeland BJ, Posy CJ, Shagrir O (eds) Computability: Turing, Gödel, Church, and Beyond, MIT Press

Abelson H, Sussman GJ, Sussman J (1985) Structure and Interpretation of Computer Programs. MIT Press

Berwick RC, Okanoya K, Beckers GJ, Bolhuis JJ (2011) Songs to syntax: the linguistics of birdsong. Trends in Cognitive Sciences 15(3):113–121

Berwick RC, Friederici AD, Chomsky N, Bolhuis JJ (2013) Evolution, brain, and the nature of language. Trends in Cognitive Sciences 17(2):89–98

Bozsahin C (2012) Combinatory Linguistics. De Gruyter Mouton, Berlin/Boston

Bratman ME (1992) Shared cooperative activity. The Philosophical Review 101(2):327–341

Chomsky N (1995) The Minimalist Program. MIT Press, Cambridge, MA

Chomsky N (2005) Three factors in language design. Linguistic Inquiry 36(1):1–22

Chomsky N (2013) Problems of projection. Lingua 130:33–49

Curry HB, Feys R (1958) Combinatory Logic. North-Holland, Amsterdam

Deacon TW (1997) The Symbolic Species: The co-evolution of language and the human brain. The Penguin Press: London

Everett DL (2005) Cultural constraints on grammar and cognition in Pirahã. Current Anthropology 46(4):621–646
Everett DL (2009) Pirahã culture and grammar: A response to some criticisms. Language 85(2):405–442
Fitch T, Hauser M, Chomsky N (2005) The evolution of the language faculty: Clarifications and implications. Cognition 97:179–210
Gazdar G (1988) Applicability of indexed grammars to natural languages. In: Reyle U, Rohrer C (eds) Natural Language Parsing and Linguistic Theories, Reidel, Dordrecht, pp 69–94
Ghallab M, Nau D, Traverso P (2004) Automated planning: theory and practice. Morgan Kaufmann, San Francisco
Gibson J (1966) The Senses Considered as Perceptual Systems. Houghton-Mifflin Co., Boston, MA
Grimshaw J (1990) Argument Structure. MIT Press, Cambridge, MA
Grosz B, Kraus S (1993) Collaborative plans for group activities. In: IJCAI, vol 93, pp 367–373
Grosz BJ, Hunsberger L, Kraus S (1999) Planning and acting together. AI magazine 20(4):23
Hale K, Keyser SJ (2002) Prolegomenon to a Theory of Argument Structure. MIT Press, Cambridge, MA
Hauser M, Chomsky N, Fitch WT (2002) The faculty of language: What is it, who has it, and how did it evolve? Science 298:1569–1579
Huybregts R, van Riemsdijk H (1982) Noam Chomsky on the Generative Enterprise. Foris, Dordrecht
Jackendoff R, Pinker S (2005) The nature of the language faculty and its implications for language evolution. Cognition 97:211–225
Jaynes J (1976) The Origin of Consciousness in the Breakdown of the Bicameral Mind. Houghton Mifflin Harcourt, New York
Joshi A (1985) How much context-sensitivity is necessary for characterizing complex structural descriptions—tree adjoining grammars. In: Dowty D, Karttunen L, Zwicky A (eds) Natural Language Parsing, Cambridge University Press, Cambridge, pp 206–250
Joshi A (1990) Processing crossed and nested dependencies: An automaton perspective on the psycholinguistic results. Language and Cognitive Processes 5:1–27
Joshi A, Schabes Y (1992) Tree-adjoining grammars and lexicalized grammars. In: Nivat M, Podelski A (eds) Definability and Recognizability of Sets of Trees, Elsevier, Princeton, NJ
Joshi A, Vijay-Shanker K, Weir D (1991) The convergence of mildly context-sensitive formalisms. In: Sells P, Shieber S, Wasow T (eds) Foundational issues in Natural Language Processing, MIT Press, Cambridge, MA, pp 31–81
Joshi AK (1983) Factoring recursion and dependencies: an aspect of tree adjoining grammars (TAG) and a comparison of some formal properties of TAGs, GPSGs, PLGs, and LPGs. In: Proceedings of the 21st annual meeting on Association for Computational Linguistics, pp 7–15
Joshi AK (2004) Starting with complex primitives pays off: complicate locally, simplify globally. Cognitive Science 28(5):637–668
Kanazawa M, Salvati S (2012) MIX is not a tree-adjoining language. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1, Association for Computational Linguistics, pp 666–674
Knuth DE (1968) Fundamental Algorithms, The Art of Computer Programming Vol. 1. Addison-Wesley, Reading, MA
Lashley K (1951) The problem of serial order in behavior. In: Jeffress L (ed) Cerebral Mechanisms in Behavior, Wiley, New York, pp 112–136, reprinted in Saporta (1961)
Lochbaum KE (1998) A collaborative planning model of intentional structure. Computational Linguistics 24(4):525–572
Manning CD (1996) Ergativity: Argument Structure and Grammatical Relations. CSLI, Stanford, CA
Nevins A, Pesetsky D, Rodrigues C (2009) Pirahã exceptionality: A reassessment. Language 85(2):355–404

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, Singapore: World Scientific Press, pp 239–246

Petrick RP, Bacchus F (2002) A knowledge-based approach to planning with incomplete information and sensing. In: AIPS, pp 212–222

Peyton Jones SL (1987) The Implementation of Functional Programing Languages. Prentice-Hall, New York

Quine WvO (1960) Word and Object. MIT Press, Cambridge MA

Saporta S (ed) (1961) Psycholinguistics: A Book of Readings. Holt Rinehart Winston, New York

Searle JR (1990) Collective intentions and actions. In: Philip R Cohen MEP Jerry L Morgan (ed) Intentions in communication, MIT Press

Shieber S (1985) Evidence against the context-freeness of natural language. Linguistics and Philosophy 8:333–343

Speas M, Roeper T (eds) (2009) Proceedings of the Conference on Recursion: Structural Complexity in Language and Cognition. Univ. of Mass, Amherst, forthcoming

Stabler E (2013) Copying in mildly context sensitive grammar. Informatics Seminars, Institute for Language, Cognition and Computation, Univ. of Edinburgh, October

Steedman M (2000) The Syntactic Process. MIT Press, Cambridge, MA

Steedman M (2002) Plans, affordances, and combinatory grammar. Linguistics and Philosophy 25:723–753

Steedman M, Petrick RP (2007) Planning dialog actions. In: Proceedings of the 8th SIGDIAL Workshop on Discourse and Dialogue (SIGdial 2007), pp 265–272

Tomasello M, Call J (1997) Primate Cognition. Oxford University Press

Tomasello M, Call J, Hare B (2003) Chimpanzees understand psychological states—the question is which ones and to what extent. Trends in Cognitive Sciences 7(4):153–156

Turing AM (1937) Computability and λ-definability. J of Symbolic Logic 2(4):153–163

Valiant L (1984) A theory of the learnable. Communications of the ACM 27(11):1134–1142

Valiant L (2013) Probably Approximately Correct: Nature's Algorithms for Learning and Prospering in a Complex World. Basic Books

Van Heijningen CA, De Visser J, Zuidema W, Ten Cate C (2009) Simple rules can explain discrimination of putative recursive syntactic structures by a songbird species. Proceedings of the National Academy of Sciences 106(48):20,538–20,543

Vijay-Shanker K (1987) A study of tree adjoining grammars. PhD thesis, University of Pennsylvania

Vijay-Shanker K, Weir D (1993) Parsing some constrained grammar formalisms. Computational Linguistics 19:591–636

Watt DA (2004) Programming Language Design Concepts. John Wiley and Sons, Chicester

Zettlemoyer L, Collins M (2005) Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In: Proc. of the 21st Conf. on Uncertainty in Artificial Intelligence, Edinburgh