Top-down derivation, recursion, and the model of grammar

Jan-Wouter Zwart

1. Introduction[1]

An aspect of human cognition that may be central to language is the ability to treat a complex structure as a single item, and to move back and forth between the complex and atomic interpretation of such elements depending on the cognitive task at hand. This ability is identified by Hofstadter (2007:83) as crucial to the development of the mental lexicon, where complex substructures in e.g. family relations may be subsumed under a single concept like 'uncle' or 'family'. Hofstadter identifies the ability to manipulate such semantic loops as marking a crucial step in the evolutionary development of human cognition.

I would like to suggest that the very same ability is involved in linguistic recursion, understood as the syntactic treatment of a complex string as a single item within another complex string. The simplex/complex ambiguity of such strings can be formalized if we understand them to be construed in a separate derivational sequence (a derivation layer), feeding into the alphabet (numeration, in minimalist terminology) for the next derivational sequence. The string under discussion, then, is complex in the context of the earlier derivation, and simplex in the context of the later derivation. If human cognition is able to jump back and forth between a complex and a simplex treatment of strings, the grammar creating those strings ('narrow syntax') may be of the minimal complexity of a finite-state grammar. If so, the complexity of natural language is not to be expressed in the type of grammar rules, but in terms of interaction among derivation layers, made possible by the atomic interpretation of complex structures.

This paper has the following structure. Section 2 establishes that the alphabet/numeration may be nonhomogeneous, in the sense that some of its symbols may be the output of a previous derivation. Section 3 recapitulates arguments for thinking that the phrase structure rules generating structure must be of a higher complexity than finite-state grammar rules or context-free grammar rules. Section 4 then proposes that the structure building rules, in their most minimalist conception, are of the finite-state type. After a discussion of the diagnostics for determining that certain symbols must be the output of a previous derivation in section 5, we use those diagnostics in section 6 to revisit the arguments concerning the type of grammar rules, and argue that once the principle of derivation layering is taken into account, the arguments lose their force. Section 7 concludes with a discussion of the nature of recursion in the model of grammar contemplated here, arguing that recursion does not reside in the rules of narrow syntax, but in the interaction among derivation layers.

2. The nonhomogenous alphabet of natural language

A key entity in formal language theory is 'symbol', which is generally not formally defined (cf. Hopcroft and Ullman 1979:1), even if it features crucially in the definition of other key concepts, such as string, alphabet, or (formal) language. A 'string of symbols' is sometimes called 'sentence' (Kimball 1973:2), and an 'alphabet', defined as a finite set of symbols (Hopcroft and Ullman 1979:2), is sometimes called 'vocabulary' or 'lexicon'. It corresponds to what is called 'numeration' in linguistic minimalism (Chomsky 1995:225), the input to the rules of grammar. A 'grammar' of a language L is a finite specification of the sentences (strings of symbols) of L (Kimball 1973:2), so that we may state that a grammar specifies sequences of symbols of a language (typically in the form of rewrite rules).

All this implies that we have some understanding of what a symbol is, involving, I think, the common implicit assumption in (1).

(1) An alphabet is a homogeneous set of symbols.

With 'homogeneous' I mean 'of a single type', where 'type' ranges over the set of common linguistic concepts in (2).

(2) { phoneme, morpheme, word, phrase, clause }

More specifically, it appears to be tacitly assumed that symbols are invariably words (3) and the question of what type of grammar we may construe (4) or what kind of language we observe (5) typically starts from that assumption.

(3) The vocabulary is simply a list of words of English (Kimball 1973:2)
(4) ...the task of constructing a finite-state grammar for English can be considerably simplified if we take [the set of symbols generated] as the set of English morphemes or words [as opposed to phonemes] (Chomsky 1956:114-115)
(5) ...when we consider the human languages purely as sets of strings of words (...), do they always fall within the class of context-free languages? (Pullum and Gazdar 1982:471)

There is, however, no reason why 'symbol' (or 'word' as an entity of formal grammar rules) should be equated with 'word' (in the sense of a natural language object), or in fact with any single type of natural language object. Intuitively, it is clear that the rules of grammar often combine elements of different types, as discussed at length in Ackema and Neeleman (2004). Ignoring phonemes, we observe that elements of all types in (2) may be combined with each other (examples from Dutch):[2]

(6) a. **word + morpheme**
vader 'father' + *-je* DIM >
vader-tje 'dear/little father'
b. **phrase + morpheme**
vader en moeder 'father and mother' + *je* DIM >
[*vader-en-moeder*]-*tje* 'playing house'
c. **phrase + word**
syntaxis en semantiek 'syntax and semantics' + *groep* 'group' >
[*syntaxis en semantiek***]-***groep* 'syntax and semantics group'
d. **clause + word**
ik weet niet wat ik moet doen 'I don't know what to do' + *gevoel* 'feeling' >
[*ik weet niet wat ik moet doen*]-*gevoel* 'feeling of not knowing what to do'
e. **clause + morpheme**
ik weet niet wat ik moet doen 'I don't know what to do' + *ge* FREQ/ITER >
ge-[*ik weet niet wat ik moet doen*] 'constantly saying that you don't know what to do'

Since words and phrases can be combined with morphemes, it would have to be the case that the alphabet (numeration) either consists of morphemes or is nonhomogeneous.

If the alphabet is nonhomogeneous, the rules of grammar may combine simplex and complex elements. But the complex elements, being complex, are structured and hence must themselves be derived by the rules of grammar. I am assuming the uniformity hypothesis in (7), referring to the structure-building process of linguistic minimalism ('Merge').

(7) *Uniformity hypothesis*
Every structured complex is created by Merge.

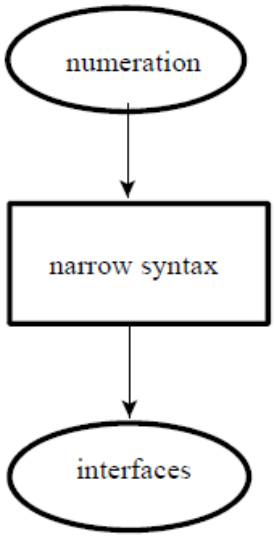
We return to the merge mechanism in section 4; for now it suffices to think of Merge as the operation that combines two elements in a string (constituent).

It follows from (7) that if the alphabet is nonhomogeneous, containing some complex element, that complex element must itself have been created by Merge. The derivation of elements like (6), therefore, must be punctuated: there must have been an operation (or sequence of operations) Merge creating a complex element and feeding that into the alphabet for the operation Merge that yields the elements in (6).

On the other hand, if the alphabet is homogeneous, consisting of morphemes only, we need subroutines creating complex elements out of a subset of elements in the alphabet. This can be done in a number of ways, either enriching the alphabet as we go along (as in Bobaljik 1995) or utilizing simultaneous workspaces ('parallel Merge', Citko 2005). (In both cases, the added complexity of the process entails that the grammar can no longer be finite-state.)

But what needs to be accounted for is that the complex elements thus derived acquire idiosyncratic properties as to form and interpretation, of the kind that in current minimalism are established at the interface components dealing with sensory/motor (sound) and conceptual/intentional (meaning) aspects of language. This assumes a model of grammar as pictured in (8), where the sequence of operations Merge is contained within the box marked 'narrow syntax' (cf. Hauser et al. 2002:1570).

(8)



Thus, whereas a complex like *handbook* may be derived within narrow syntax (i.e. via Merge) from a numeration containing the elements *hand* and *book*, its noncompositional meaning '(short) book giving all the important information' is not just a function of Merge, but must be established at the interfaces. Using *handbook* in a clause, then, requires a loop feeding *handbook*, with its special properties, from the interface component into another numeration.

It follows that the subroutines needed if the numeration/alphabet is homogeneous must involve the interface components, and hence that these subroutines are in fact full-fledged derivations of the type in (8). Therefore, derivations must be 'layered', and there is no need to maintain that the alphabet must be homogeneous at all times.

3. Arguments relating to the typing of grammar rules

The possibility of nonhomogeneous alphabets affects the argumentation regarding the nature of the rules of (phrase structure) grammar in a significant way.[3] As is well known, the earliest discussions of transformational generative grammar (e.g. Chomsky 1956) contain proofs that the rules of grammar of natural languages like English are not of the finite-state type, and later discussions have centered around the question of whether these rules are of the context-free or context-sensitive type (Huybregts 1976, Pullum and Gazdar 1982, Bresnan et al. 1982, and others).[4]

Discussions of the type of grammar rules for natural language invariably assume the alphabet to be homogeneous. As we will see in section 6, allowing for nonhomogeneous alphabets (in combination with the recursive loop discussed in section 2) undermines the proof that the rules of grammar cannot be finite-state. Here we just want to present that proof, as well as a proof showing that the rules cannot be context-free.

The commonly (though not universally) accepted proof demonstrating that English is not a finite-state language (from Chomsky 1956:115; see also Kimball 1973:22f, Partee et al. 1990:480) is in (9), quoted from Langendoen (2003:26-28).

(9) to account for the syntactic dependency between the words *if* and *then* in a sentence like *if it rains then it pours*, English grammar must contain a rule like $S \rightarrow$ *if S then S*. If so (..), then English also contains the sentences *if if it rains then it pours then it pours* and *if if if it rains then it pours then it pours then it pours*, but not **if it rains then it pours then it pours* nor **if if it rains then it pours*. That is, English contains every sentence of the form $\{(if)^m$ *it rains (then it pours*$)^n \mid m = n\}$, but no sentence of the form $\{(if)^m$ *it rains (then it pours*$)^n \mid m \neq n\}$. Hence English is not a finite-state language.

Chomsky (1956: 115) likens sentences containing such a (recursive) *if/then* dependency to sentences of the type in (10), i.e. "sentences consisting of *n* occurrences of *a* followed by *n* occurrences of *b*, and only these".

(10) $a^\frown b$
$a^\frown a^\frown b^\frown b$
$a^\frown a^\frown a^\frown b^\frown b^\frown b$
etc

A finite-state grammar contains rules turning an initial (nonterminal) state *A* into a terminal *a* or into a string of a terminal *a* and a nonterminal *B* (see (11); the '|' symbol indicates disjunction).

(11) *finite-state grammar rule*
$A \rightarrow a \mid a\ B$

The nonterminal *B* in (11) may be rewritten by a subsequent application of the finite-state grammar rule, creating perhaps a terminal *b*, but nothing guarantees the balanced and potentially infinite accretion of *a*'s and *b*'s we see in (10) (see Chomsky 1956:115).

The fact that English has sentences that have exactly the property in (10) shows that English is not a finite-state language, where a finite-state language is defined as the set of strings generated by finite-state grammar rules. We return to this argument in section 6.

It has been argued that phrase structure rules generating natural language sentences have to be context-sensitive, rather than context-free, based on examples like (12), from Swiss German (Huybregts

1984, extending an argument from Huybregts 1976; Bresnan et al. 1982).

(12) *wil* *mer* *de* *maa* *em* *chind*
because 1PL:NOM DET:ACC man DET:DAT child

lönd *hälffe* *schwüme* (Zurich German; Huybregts 1984:91)
let:PAST.3PL help:INF swim:INF

'because we let the man help the child to swim'

In (12), causative *lönd* 'let' selects the accusative noun phrase *de maa* 'the man', and *hälffe* 'help' selects the dative noun phrase *dem chind* 'the child', yielding crossing case dependencies. Adding further embedded clauses adds further crossing dependencies, in such a way that the noun phrases and the verbs constitute potentially infinite series of crossing dependencies. (The facts in Dutch are the same, except that the dependencies are not expressed in case morphology.) The sensitivity to context that these crossing dependencies display cannot be expressed in context-free rewrite rules, indicating that natural languages are not context-free languages (Huybregts 1984:93).[5]

Both arguments assume that the grammar of a language contains just rewrite rules (i.e. is a phrase structure grammar). As Chomsky (1956) argued, phrase structure grammars are inadequate in that rules of phrase structure grammar fail to express natural language regularities captured by transformation rules. Cutting corners, we might state that English is not a finite-state (or context-free, or context-sensitive) language because to describe sentences of English, we need transformations in addition to phrase structure rules (see also note 3).

Transformations range from morphological adjustment via reordering (displacement) to deletion and insertion. In linguistic minimalism, some of these processes are no longer considered to be part of narrow syntax. Morphological adjustment (or morphological expression in general) is taken to be postsyntactic (Chomsky 1995:229), and deletion may just be failure to spell out at the interface component dealing with sound (cf. Merchant 2001). On the other hand, insertion and displacement are unified in the single operation Merge, joining some element to the structure being derived (extracted from out of that structure, in the case of displacement).

In minimalism, then, the neat separation between structure building (rewrite rules) and structure manipulation (transformations) is gone. Structure is created by a single process Merge, which may involve movement.[6] In the context of the present discussion, this leads to the question what kind of operation Merge is. I argue in what follows that Merge may be equated to a finite-state grammar rule, and that the complexity of natural language illustrated in (9) and (12) is brought in by the recursive loop process that turns the output of one derivation layer into part of the input for the next derivation layer.

4. The nature of Merge

Let us assume that the rules of grammar of a language *L* must derive at least the information in (13) about the sentences of *L*. (In what follows, we ignore movement entirely; for issues arising in this connection, cf. Zwart 2009.)

(13) a. constituency
b. hierarchy
c. dependency

We derive constituency if the rules allow us to single out a string as a unit. This is typically done by

conceiving of the rules ('Merge') as combining two elements in a set (or parts in a whole):

(14) *common conception of Merge (1)*
- (i) There is a numeration N
- (ii) There is an operation ('Merge') such that
  - a. Merge takes two elements *x*, *y* from N, and
  - b. Merge combines *x* and *y*

Hierarchy is derived if we allow Merge to combine an element from N with the newly constructed *x*-*y* combination, which I call the 'object under construction' ((15) supplementing (14)):

(15) *common conception of Merge (2)*
- (i) There is a numeration N
- (ii) There is an object under construction *A*
- (iii) There is an operation ('Merge') such that
  - a. Merge takes an element z from N, and *A*, and
  - b. Merge combines either z and *A*

Dependency is derived if a dependent element needs to be in a hierarchical relation to its antecedent, i.e. a relation defined in terms of (15), as in Epstein (1999).

The fact that Merge, in its common conception, is either (14) or (15) looks like an imperfection. In fact, the only time we need (14) is the first time Merge applies, as there is no object under construction at that point (Jaspers 1998:109). But if we allow the object under construction to be empty at that point (Fortuny 2008:18, Zwart 2009), (14) is redundant, and we can simplify the conception of Merge to (15). Hierarchy and dependency still follow.

Still, this conception of Merge seems overly complicated in its two-step process of selecting elements and combining them. To use a common image, it is like both the numeration and the object under construction (the derivation) are in separate spatial locations, and Merge takes an element from the numeration and transfers it to the derivation. I find myself in agreement with Bobaljik (1995:47), who argues that the common conception of Merge as involving transfer is an artifact of the way the operation is notated, while what is going on instead is that Merge articulates a particular relation among the elements of the numeration.

One implementation of this idea, discussed in Zwart (2009, 2011a), is that Merge (now a misnomer) starts with an unordered numeration set, and with each step orders one element from that set with respect to the residual members of the set ('top-down Merge').[7]

(16) *Top-down Merge*
- a. There is a numeration N
- b. There is an operation ('Merge') converting N into $\langle (x \in \mathrm{N}), (\mathrm{N} - x) \rangle$ [8]

This operation yields a constituent (N − *x*), as well as hierarchical structure, if the operation can subsequently apply to (N − *x*), i.e. N becomes (N − *x*). The result is a nested sequence of ordered pairs, and dependency can again be defined in terms of the hierarchical structure thus derived (or, more exactly, in terms of the sequence of operations Merge).

The top-down conception of Merge essentially splits a nonterminal element into a terminal element and another nonterminal element. The operation ends when the final remaining element is split off from the residue set, yielding just a terminal. The rule of grammar, then, is the finite-state grammar rule (11), yielding either a string of a terminal and a nonterminal, or a terminal.

The discussion would be complicated if we wanted the rules of grammar to express movement. For

the common conception of Merge, movement implies a further situation, not covered in (14)-(15), where Merge combines the object under construction *A* with an element from *A*. For the top-down conception of Merge (16), some representation of the element *x* split off from N would have to remain part of N, either as a copy or a trace or a feature, to be split off later, in order to account for the observation that *x* 'belongs' in a position where we do not see it. These complications take us too far afield at this point, so I will continue on the assumption that they will not jeopardize the approach contemplated here, an assumption that may well be off the mark.

In what follows, then, I take (16) to be the minimalist conception of the structure-building process. Narrow syntax (cf. [8]) merely involves a sequence of operations of this type, suggesting that at least this part of grammar is finite-state.[9] With that in mind, let us return to the argumentation in section 3, concerning the proper characterization of the rules of grammar for natural language, now assuming the alphabet/numeration to be nonhomogeneous.

5. The composition of the numeration

If the alphabet/numeration can be nonhomogeneous, a simple sentence like (17) may receive a number ([i]-[iv]) of string-set analyses.

(17) *The man left*
(i) (the) (man) (left)
(ii) (the man) (left)
(iii) (the) (man left)
(iv) (the man left)

The corresponding alphabets are:

(18) (i) { the, man, left }
(ii) { the_man, left }
(iii) { the, man_left }
(iv) { the_man_left }

If (17) is analysed as in (17iv), the grammar (the top-down merge machine) turns the initial state (18iv) into (17) in one step. The grammar then involves a single operation (16b), articulated in (20), which is a finite-state grammar rule of the type in (21), turning the initial state *S* (= [18iv]) into a single terminal *a* (= [17iv]).

(19) { the_man_left } → ⟨ [the man left], ∅ ⟩

(20) $S \rightarrow a$

In the remaining cases, the machine turns the initial state into one or more intermediate stages before reaching the end stage. The rules of the grammar then all have the form in (21)(= [11]), where *A* is a nonfinal state, *a* is a terminal, and *B* an intermediate stage.

(21) $A \rightarrow a\,B \mid a$

For example, to derive (17i) from (18i), the rules, terminals, and intermediate stages are as in (22).

(22)

	initial stage	rule	terminal	intermediate stage
1.	{ the, man, left}	(21)	*the*	{ man, left }
2.		(21)	*man*	{ left }
3.		(20)	*left*	∅

For our purposes, there is the additional question of which of the possible analyses is the correct one, i.e. is the one yielding the correct structure.

A structure is cognitive reality brought out by experiments known as constituency tests, indicating which substrings are perceived as units by the speaker. If these units must be brought out by the grammar, a nonambiguous string has a single correct derivation. Since constituency tests identify *the man* in (17) as a unit, the correct derivation of (17) is the one that yields (17ii), i.e. the one that takes the alphabet for this sentence to be (18ii).

Importantly, the string *the man* itself can be derived by a finite-state machine from the alphabet {(the), (man)}. If so, the alphabet for each string consists of symbols that are primitive relative to the derivation of that string, but not in an absolute sense.

It is easy to see that in a system with nonhomogeneous (relativized) alphabets, recursion takes the form of 'derivational interaction', in the sense that a symbol in the alphabet for derivation D_1 may itself be a string of terminals generated in derivation D_2 (see Zwart 2011b).

With this in mind, let us return to the question of sentences of English referred to in (9), proving that English is not a finite-state language.[10]

6. Revisiting the question of grammar rule types

Sentences of the structure *if X then Y* obviously have the structure in (23):

(23) ((if A) (then (B)) = *a b*,
where *a* = *if A*
b = *then B*

An additional conditional construction (i.e. a pair consisting of a conditional clause and a consequent clause), as illustrated in (9), would substitute for *A* in (23). Since *A* is a term of *a*, the analysis is not crucially altered by multiplying the conditionals and consequents.

Assuming nonhomogeneous alphabets, then, the string *if it rains then it pours* can be the output of a derivation over the alphabet in (24), and *if if it rains then it pours then it pours* can be the output of a derivation over the alphabet in (25).

(24) { if_it_rains, then, it, pours }
(25) { if_if_it_rains_then_it_pours, then , it, pours }

Of course, the symbols *if_it_rains* in (24) and *if_if_it_rains_then_it_pours* in (25) can themselves be analysed as outputs of derivations over alphabets, such as (26) for (24) and (27) for (25), and the unanalyzed complex symbols can likewise be further analysed.

(26) { if, it_rains }
(27) { if, if_it_rains_then_it_pours }

The point is that *if (if it rains then it pours) then it pours* does not have the structure $a^\frown(a^\frown b)^\frown b$, or $(a^\frown a)^\frown(b^\frown b)$, with infinite parallel accretion of *a*'s and *b*'s, but the structure $a^\frown b$, regardless the complexity of *a*. The parallel accretion of *a*'s and *b*'s results from the recursive inclusion of an $a^\frown b$ structure (a conditional-consequent pair) inside *a*, and the relevant class of constructions is not correctly characterized as in (28), but as in (29), where *S* can be any clause (including another instance of *if S*).

(28) $\{(if)^m$ *it rains (then it pours)*$^n \mid m = n\}$

(29) *(if S) then it pours*

What remains is the *local* dependency of a consequent clause and a conditional clause, which is repeated within *S* in (29) if it contains a conditional construction (a conditional-consequent pair). But such a dependency can be handled within a finite-state grammar, if dependency is a function of Merge, as we have assumed (see section 4).

Thus, if we allow symbols to be the complex output of separate derivation layers, and we define recursion as the interaction between derivation layers, the rules of narrow syntax ('Merge') can be of the ultimate minimalist finite-state type. It remains the case that English is not a finite-state language (see note 3), but given the definition of recursion as derivational interaction, the narrow syntax component of English grammar may still be finite-state. For this conclusion to hold, we have to assume that the human cognitive capacity to treat objects as both complex and atomic, identified by Hofstadter (2007), supplements the faculty of language in the narrow sense as defined in Hauser et al. (2002).

It remains to discuss the argument based on (12) showing that the rules of grammar are context-sensitive. Constructions of the type in (12), typical of the Continental West-Germanic languages, are notoriously complicated, with no single analysis of the various types and variations being currently uniformly accepted (see Zwart 1996 and Wurmbrand 2005 for a survey of the phenomena and their analyses). For our purposes, the interesting question is whether particular substrings in examples of this type can be characterized as outputs of separate derivation layers.

In Dutch, a verb-second language, strings that appear before the finite verb in independent clauses can be identified as constituents (Zwart 2011c:21). This test allows us to identify verb clusters as constituents:

(30) a. *Ik* *heb* *hem* *de kinderen* *niet*
1SG:NOM have:1SG 3SG.MASC:OBJ DETchild:PL NEG

helpen ***leren*** ***zwemmen***
help:INF teach:INF swim:INF

'I did not help him to teach the children to swim.'

b. ***Helpen*** ***leren*** ***zwemmen*** *heb* *ik* *hem*
help:INF teach:INF swim:INF have:1SG 1SG:NOM 3SG.MASC:OBJ

de kinderen *niet*
DETchild:PL NEG

(same as a.)

The constituency of the verb cluster *helpen leren zwemmen* [help teach swim] is consistent with the idea that the verb cluster is the output of a separate derivation, hence a single symbol in the alphabet (numeration) for the derivation of the clause containing it (30a). Moreover, subparts of the verb cluster cannot be separately fronted:

(31) a. * ***Leren*** ***zwemmen*** *heb* *ik* *hem*
teach:INF swim:INF have:1SG 1SG:NOM 3SG.MASC:OBJ

de kinderen *niet* ***helpen***
DETchild:PL NEG help:INF

b. * ***Zwemmen*** *heb* *ik* *hem*
swim:INF have:1SG 1SG:NOM 3SG.MASC:OBJ

de kinderen *niet* ***helpen*** ***leren***
DETchild:PL NEG help:INF teach:INF

(both intended the same as [30a])

This argues against an alternative analysis, where the fronted constituent in (30b) is the remnant of a phrase depleted by extraction of some of its subparts (i.e. an accidental constituent; cf. den Besten and Webelhuth 1987). This kind of derivation, not available in a top-down model (cf. [16]), and also otherwise contested (cf. Fanselow 2002), does not account for the observation that the cluster as a whole behaves like an atomic constituent. In terms of the top-down derivational machine (16): subparts of the verb cluster may not be split off from the numeration independently, which is explained if the cluster is a single symbol in the relevant numeration.[11]

If verb clusters are created in separate derivation layers, noun phrases associated semantically with the verbs in the cluster must be linked with those verbs via c-command (dependency created as a function of merge), i.e. must be 'base-generated' in their surface position, again entirely consistent with the top-down approach of section 4. The relation between the noun phrases and the verb cluster, then, is no longer one of cross-serial dependency, but a many-to-one relation. In this context, it is relevant to note that the order of the verbs inside the cluster is subject to considerable variation (both within and among dialects), so that the particular cross-serial dependency configuration is just one of many possibilities, hence arguably more coincidental than systematic.

7. The place of recursion in the model of grammar

The top-down merge machine (16) essentially performs a series of identical steps, each step identifying a different symbol x from the alphabet/numeration N and in doing so reducing the set of unordered symbols $(N - x)$. Both x and $(N - x)$ are constituents, and the result at each step is an ordered pair. As I have argued elsewhere, dependency phenomena of natural language may be the linguistic interpretation of the asymmetry between the members of such an ordered pair (Zwart 2009:165f).

It is common to think of a sequence of operations Merge as recursive (e.g. Nevins, Pesetsky, and Rodrigues 2009:366). This is because Merge (under any conception of it) is a rule creating an output that may be subjected to an application of the same rule. However, it is not the case that the structured objects created by Merge (hierarchical syntactic structures) are *necessarily* created by a recursive process (see e.g. Arsenijević and Hinzen 2012).

For example, assuming Merge to involve transfer from a numeration to an object under construction

(essentially [16]), hierarchical structures are created by transfering elements one by one, an iterative procedure. Similarly, under the top-down conception of Merge (16), elements are split off from the numeration one by one. In each case, we derive hierarchical structures by means of an iterative procedure.

On the other hand, the loop that results from taking the complex output of one derivation layer to be the atomic input to a next derivation layer is inherently recursive, in the sense that a symbol in the alphabet for a procedure *P* is itself the output of *P*. Here recursivity is not an artifact of the notation of the procedure, but is the unavoidable correlate of the simplex/complex ambiguity of linguistic objects alluded to in the introduction.

This view of recursion shows some resemblance to the earliest conception of recursion in generative grammar, involving generalized transformations (Chomsky [1955]1975:383, 518; Chomsky 1961:134; Chomsky 1966:52f). Generalized transformations are used to derive complex sentences; they are overarching transformations that operate on fully developed phrase markers. For example, a clause may function as an object in another clause by substituting for a dummy symbol in that other clause (Chomsky 1961:52-53 note 2). In the cyclic, layered derivations approach contemplated here, there is no need for a dummy symbol, as the embedded clause functions as a single symbol in the alphabet for the derivation of the embedding clause.

In what Palmatier (1972:ix) calls 'second-generation transformational grammar' (roughly from the mid-1960s), generalized transformations were abandoned, and recursion was written straight into the phrase structure rules, which were allowed to reintroduce the start symbol *S* (Chomsky 1966:63). Since generalized transformations were typically substitutions targeting a dummy symbol, taking the dummy symbol to be *S* (or some other rewritable category) allows one to eliminate generalized transformations altogether. This development was made possible after Katz and Postal (1964:120ff) had shown that the generalized transformations themselves made no semantic contribution.

In minimalism (e.g. Chomsky 1993:21), the need to introduce elements of arbitrary complexity directly (i.e. via Merge) was interpreted as motivating a device like generalized transformations once more (essentially in the context of abandoning the distinction between D-structure and surface structure; see also Frank 2002:9). But while this gave rise to exploitation of parallel derivations, the simple consequence of allowing derivations to feed into the numeration for further derivations was not, to my mind, sufficiently explored, nor were the consequences for the definition of recursion.

8. Conclusion

In this article, I have argued that common views on the nature of phrase structure rules (finite-state or of a higher complexity) suffer from an unmotivated and unnecessary hidden assumption, namely that the rules of grammar are fed by a homogeneous set of symbols, the alphabet/numeration. Once it is understood that the symbols in a numeration may themselves be the output of a separate derivation, arguments against the finite-state character of the phrase structure rules ('Merge') lose their force.

I have argued that the minimalist conception of Merge involves the iterative conversion of the alphabet/numeration into a sequence of ordered pairs, each consisting of an element split off from the numeration and the residue of the numeration at that point, i.e. a terminal/nonterminal pair, mimicking a finite-state machine. Viewed this way, narrow syntax is just the conversion of an unordered set into an ordered n-tuple (cf. Fortuny 2008), and the complexity of the elements involved is not created by recursivity in the rules of narrow syntax, but by the circumstance that some of these elements have their own derivational history, being the output of a separate derivational sequence. I submit that the cognitive ability to treat elements simultaneously as simplex and complex underlies much of the complexity of natural language, allowing us to consider the rules of grammar to be maximally simple.

It follows that if Hauser et al. (2002) are right in identifying recursion as the single species-specific property of the human faculty of language, and if we are right here, human language derives its unique

properties not from the structure building procedure Merge, but from the circumstance that sequences of operations Merge (derivation layers) may interact, yielding atomic symbols of potentially infinite complexity.

Notes

1. Thanks to Gertjan van Noord, Riny Huybregts, Jordi Fortuny, Jorike van Werven, and the audience at the Konstanz *Complex sentences, types of embedding, and recursivity* workshop, March 5-6, 2012, as well as to two anonymous reviewers. This article reports on research carried out within the *Dependency in Universal Grammar* research program, sponsored by the Netherlands Organization for Scientific Research (program number 360-70-200), which is gratefully acknowledged.
2. Abbreviations used in the glosses: ACC = accusative case, DAT = dative case, DET = determiner, DIM = diminutive, FREQ = frequentative, INF = infinitive, ITER = iterative, MASC = masculine gender, NEG = negation, NOM = nominative case, OBJ = objective case, PL = plural number, SG = singular number, 1 = first person, 3 = third person.
3. A note of clarification is in order here. Discussion on the formal grammar of natural languages typically focuses on the type of language, rather than the type of rules. That is, the question is whether the sentences of a language (the string-sets) fall within a language-type class. In the model of grammar assumed here (cf. [8]), the sentences of a language are derived by a complex of operations, involving a structure-building process ('Merge') that can be described as a set of phrase structure rules, and a variety of interface processes, involving morphophonology, linear ordering, and the establishment of particular sound-meaning pairings, but also unpredictable processes such as reanalysis and recategorization (cf. Zwart 2009). On top of that, we must allow for derivation layering, allowing the output of one derivation to feed the next. My concern here is not with language as a whole, but with a small but important component of the model of grammar, narrow syntax, and with the nature of the rules needed to generate the type of structures we take to be its output. This is relevant to the hypothesis of Hauser et al. (2002) that narrow syntax represents the faculty of language in the narrow sense, a component of the human cognitive system that is arguably species-specific.
4. The assumption of transformational rules, already in Chomsky (1956), significantly reduced the urgency of these questions in mainstream generative grammar, a point that I will side-step here.
5. For discussion of an alternative conclusion, that languages apparently vary as to the complexity of the rewrite rules of their grammars, see Sauerland (2013).
6. *Sentences*, on the other hand, are derived by Merge in combination with a range of additional processes, see note 3.
7. See Phillips (2003) and Chesi (2007) for an earlier top-down generative model, and Zwart (2009: 165) for discussion of the differences with the present proposal.
8. Informally, the rule splits an element off from the numeration and yields an ordered pair consisting of that element and the numeration minus the element split off from it.
9. Importantly, the conclusion that this part of the grammar, Merge, is finite-state, does not entail that the entire grammar, or the language generated by that grammar, is finite-state. If we are correct, a sentence is derived by a network of derivations, in which little finite-state units interact. It follows that the question of formal complexity cannot be resolved simply by inspecting sentences. Rather, the network of derivations involved needs to be identified, and the question of complexity can be asked of each subderivation. See Trotzke and Zwart (to appear) for discussion.
10. The idea of describing complex strings as involving separate (finite-state) subroutines is familiar from the literature on natural language parsing (Woods 1970, Bates 1978, Abney 1996, Roche 1997). As Roche (1997:269) observes, this cuts into the argument that to describe expressions with a certain level of complexity, we need more powerful formalisms, such as context-free grammars.
11. A complication is that we have to allow for the finite verb to escape from the cluster to obtain the verb second position (e.g. *heb* in [31] in the text is not or no longer part of the verb cluster, whereas it is string-adjacent to or included in the cluster in embedded clauses, not illustrated here). This, however, may not be a problem if Chomsky (2001) is correct in identifying verb-second as a phenomenon of the interface component dealing with sound (see also Anderson 1993, Zwart 2005), and if we allow interface phenomena to affect parts of

strings that are treated as atomic symbols in narrow syntax (as we must to account for morphological or prosodic marking anyway).

References

Abney, Steven
1996 Partial parsing via finite-state cascades. *Natural Language Engineering* 2, 337-344.
Ackema, Peter and Ad Neeleman
2004 *Beyond morphology: interface conditions on word formation*. Oxford: Oxford University Press.
Anderson, Stephen
1993 Wackernagel's revenge: clitics, morphology, and the syntax-semantics interface. *Language* 69: 68-98.
Arsenijević, Boban and Wolfram Hinzen
2012 On the absence of X-within-X recursion in human grammar. *Linguistic Inquiry* 43: 423-440.
Bates, Madeleine
1978 The theory and practice of augmented transition network grammars. In *Natural language communication with computers*, Leonard Bolc (ed.), 191-254. Berlin: Springer Verlag.
Bobaljik, Jonathan D.
1995 In terms of merge: copy and head movement. *MIT Working Papers in Linguistics* 27: 41-64.
Bresnan, Joan, Ronald Kaplan, Stanley Peters, and Annie Zaenen
1982 Cross-serial dependencies in Dutch. *Linguistic Inquiry* 13: 613-635.
Chesi, Christiano
2007 An introduction to phase-based minimalist grammars: why *move* is top-down and from left-to-right. *CISCL Working Papers on Language and Cognition 1 (Studies in Linguistics)*: 38-75.
Chomsky, Noam
1956 Three models for the description of language. *IRE transactions on information theory*, vol. IT-2 (*Proceedings of the symposium on information theory*): 113-124.
1961 On the notion 'rule of grammar'. *Proceedings of the Symposium in Applied Mathematics* 12: 6-24.
1966 *Topics in the theory of generative grammar*. The Hague: Mouton.
1975 *The logical structure of linguistic theory*. New York: Plenum Press.
1993 A minimalist program for linguistic theory. In *The view from Building 20: essays in linguistics in honor of Sylvain Bromberger*, Kenneth Hale and Samuel J. Keyser (eds.), 1-52. Cambridge: MIT Press.
1995 *The minimalist program*. Cambridge: MIT Press.
2001 Derivation by phase. In *Ken Hale: a life in language*, Michael Kenstowicz (ed.), 1-52. Cambridge: MIT Press.
Citko, Barbara
2005 On the nature of merge: external merge, internal merge, and parallel merge. *Linguistic Inquiry* 36: 475-497.
Den Besten, Hans and Gert Webelhuth
1987 Remnant topicalization and the constituent structure of the VP in Germanic SOV languages. *GLOW Newsletter* 18: 15-16.
Epstein, Samuel D.
1999 Un-principled syntax: the derivation of syntactic relations. In *Working minimalism*, Samuel D. Epstein and Norbert Hornstein (eds.), 317-345. Cambridge: MIT Press.
Fanselow, Gisbert
2002 Against remnant movement. In *Dimensions of movement: from features to remnants*, Artemis Alexiadou, Elena Anagnostopoulou, Sjef Barbiers, and Hans-Martin Gärtner (eds.), 91-127. Amsterdam: John Benjamins.
Fortuny, Jordi
2008 *The emergence of order in syntax*. Amsterdam: John Benjamins.
Frank, Robert

2002 *Phrase structure composition and syntactic dependencies*. Cambridge: MIT Press.
Hauser, Marc, Noam Chomsky, and W. Tecumseh Fitch
2002 The faculty of language: what is it, who has it, and how did it evolve? *Science* 298: 1569-1579.
Hofstadter, Douglas
2007 *I am a strange loop*. New York: Basic Books.
Hopcroft, John E. and Jeffrey D. Ullman
1979 *Introduction to automata theory, languages, and computation*. Reading: Addison-Wesley Publishing Company.
Huybregts, Riny
1976 Overlapping dependencies in Dutch. *Utrecht Working Papers in Linguistics* 1: 24-65.
1984 The weak inadequacy of context-free phrase structure grammars. In *Van periferie naar kern*, Ger J. de Haan, Mieke Trommelen, and Wim Zonneveld (eds.), 81-99. Dordrecht: Foris Publications.
Jaspers, Dany
1998 Categories and recursion. *Interface: journal of applied linguistics* 12: 81-112.
Kimball, John P.
1973 *The formal theory of grammar*. Englewood Cliffs: Prentice-Hall.
Langendoen, D. Terrence
2003 Finite state languages and grammars. In *The Oxford International Encyclopedia of Linguistics (2nd edition)*, William J. Frawley (ed.), Vol. 2, 26-28. Oxford: Oxford University Press.
Merchant, Jason
2001 *The syntax of silence*. New York: Oxford University Press.
Nevins, Andrew, David Pesetsky, and Cilene Rodrigues
2009 Pirahã exceptionality: a reassessment. *Language* 85: 355-404.
Palmatier, Robert A.
1972 *A glossary for English transformational grammar*. New York: Appleton-Century-Crofts.
Partee, Barbara, Alice ter Meulen, and Robert E. Wall
1990 *Mathematical methods in linguistics*. Dordrecht: Kluwer Academic Publishers.
Phillips, Colin
2003 Linear order and constituency. *Linguistic Inquiry* 34: 37-90.
Pullum, Geoffrey K. and Gerald Gazdar
1982 Natural languages and context-free grammars. *Linguistics and Philosophy* 4: 471-504.
Roche, Emmanuel
1997 Parsing with finite-state transducers. In *Finite-state language processing*, Emmanuel Roche and Yves Schades (eds.), 241-281. Cambridge: MIT Press.
Sauerland, Uli
2013 Against complexity parameters. In *Syntactic complexity across interfaces*, Andreas Trotzke and Josef Bayer (eds.), this volume. Berlin: De Gruyter Mouton.
Trotzke, Andreas and Jan-Wouter Zwart
to app. The complexity of Narrow Syntax: minimalism, representational economy, and simplest merge. In *Measuring syntactic complexity*, Frederick J. Newmeyer and Laurel B. Preston (eds.). Oxford: Oxford University Press.
Woods, William A.
1970 Transition Network Grammars for natural language analysis. *Communications of the ACM* 13, 591-606.
Wurmbrand, Susi
2005 Verb clusters, Verb Raising, and restructuring. In *The Blackwell companion to syntax*, Martin Everaert, Henk van Riemsdijk, Rob Goedemans, and Bart Hollebrandse (eds.), Vol. 5, 227-341. Oxford: Blackwell.
Zwart, Jan-Wouter
1996 Verb clusters in Continental West-Germanic dialects. In *Microparametric syntax and dialect variation*, ed. James R. Black and Virginia Motapanyane (eds.), 229-258. Amsterdam: John Benjamins.
2005 Verb second as a function of merge. In *The function of function words and functional categories*, ed. Marcel den Dikken and Christina M. Tortora (eds.), 11-40. Amsterdam: John Benjamins.
2009 Prospects for top-down derivation. *Catalan Journal of Linguistics* 8: 161-187.

2011a Structure and order: asymmetric merge. In *The Oxford handbook of linguistic minimalism*, Cedric Boeckx (ed.), 96-118. Oxford: Oxford University Press.
2011b Recursion in language: a layered-derivation approach. *Biolinguistics* 5: 43-56.
2011c *The syntax of Dutch*. Cambridge: Cambridge University Press.