Glossa
a journal of general linguistics

Phrasal Spellout and Partial Overwrite: On an alternative to backtracking

RESEARCH

HAGEN BLIX

ubiquity press

Abstract

This paper explores branching vocabulary items as a possible alternative to backtracking in Nanosyntax. Such vocabulary items give rise to the possibility of partial overwrite. Initially, they spell out a node with two phrasal daughters. At a subsequent stage, the right branch is interpreted by a new element, but the original item continues to spell out the left branch of such a derivation. The effects that branching vocabulary items give rise to thus mimic aspects of backtracking, without requiring mechanisms that undo parts of the derivation.

CORRESPONDING AUTHOR:
Hagen Blix
New York University, US
hagen.blix@nyu.edu

KEYWORDS:
Nanosyntax; Phrasal Spellout; Backtracking; Spellout-Driven Movement; Partial Overwrite; Pointers; Lexicalized Tree Structures

TO CITE THIS ARTICLE:
Blix, Hagen. 2021. Phrasal Spellout and Partial Overwrite: On an alternative to backtracking. *Glossa: a journal of general linguistics* 6(1): 62. 1–17. DOI: *https://doi.org/10.5334/gjgl.1614*

DOI: 10.5334/gjgl.1614

1 Introduction

In Nanosyntax (Starke 2018, Caha 2019), vocabulary items are hypothesized to interpret phrasal nodes only. Mediated by a superset-based matching algorithm (a vocabulary item matches all trees that the tree it lexicalizes contains), this leads to the vocabulary driving (parts of) the syntactic derivation: In order for a derivation to converge, it must consist solely of trees with matching vocabulary items. To ensure such convergence upon merging a feature F, the derivational algorithm 'attempts' a variety of operations in an ordered fashion until it finds a candidate that can successfully spell out, as in (1):

(1) Merge (F, XP), then:
   a. Try: Spell out FP
   b. If it fails, try: Move(Spec of XP) to Spec FP, Spell out FP
   c. If it fails, try: Move(XP) to Spec FP, Spell out FP
   d. If it fails: Go to the previous cycle that merged X, and try the next option for that cycle

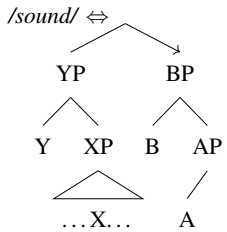
Lightly adapted from Starke (2018), Caha (2019)

Upon merging a feature F, the spellout algorithm will first (1a) attempt to spell out the resulting FP (resulting in F being spelled out with the current 'root' structure). Should that fail (1b), it will attempt spec-to-spec movement (generally resulting in F being spelled out with the current suffix structure). If that, too, fails (1c), F moves its complement to its specifier – roughly corresponding to the "construction" of a new suffixal position. These three steps create a preference for spellout targets that are as large as possible. Should they fail, however, option (1d), the so-called *backtracking* option, returns to the previous cycle, changing the structure of XP itself.

Note however that *backtracking* differs categorically from the other three steps: Step (1d) calls the spellout algorithm itself, i.e., its addition turns spellout into a recursively defined algorithm. The corollary, in terms of computational cost, is that it turns the linear algorithm that is defined by (1a-c), into an exponential algorithm. That is to say, for a derivation of length n (or $n+2$, if we consider first merge), an algorithm without backtracking would consider no more than on the order of $3n$ derivational stages (in the sense of Collins & Stabler 2016). Since backtracking makes the function recursive, however, every feature triples the set of possible derivational stages that may need to be considered in the full algorithm, i.e., the worst-case scenario is a comparatively costly 3^n.[1]

Given this computational cost, it is worth asking if backtracking is a necessary part of the spellout algorithm, or if we can do without a recursive extension of the theory. In this paper, I explore the possibility that the desired effects of backtracking can in fact be implemented in a linear spellout algorithm, if we employ branching vocabulary items and pointers, as in (2) – with A, B and Y part of the extended projection of X.

(2)



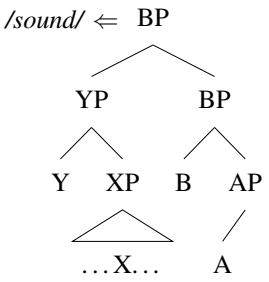
A pointer (indicated by the arrow on the right branch) is a common nanosyntactic device of conjunction: In this case, it conjoins YP with any subtree of BP (here: BP or AP). Hence, the vocabulary item in (2) could be described as follows: It matches any subtree of the set of trees that is formed by conjoining YP with a subtree of BP.

1 Technically, the algorithm is a depth-first search over a decision tree in which any node has three daughters, representing the three ways to introduce and spell out a feature. However, since any node with three daughters that are all impossible to spell out will not have its daughters' daughters explored, the worst-case scenario is, of course, an unlikely edge case. This does not change the fact that the algorithm is exponential with backtracking and linear without backtracking. Section 2 will provide examples and a comparison of decision trees for particular derivations.

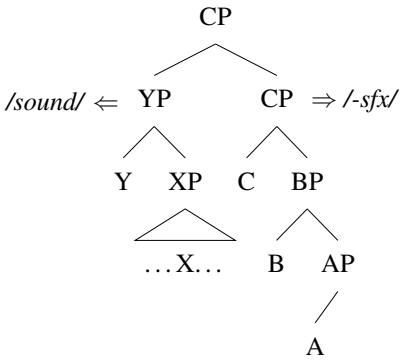
Such branching vocabulary items with a pointer have two properties that will turn out to be crucial for our purposes: First, they are able to drive movement even when the root node is the target of spell out. That is to say, when A is merged with YP, the result of comp-to-spec movement of YP can be matched by the vocabulary item at the root, as can the result of subsequent spec-to-spec movement upon merging B. Second, such vocabulary items can be subject to *partial overwrite*, an effect that turns out to look highly similar to aspects of backtracking.

Both properties are illustrated in (3): After merging A, simple spellout fails. There is no relevant specifier that spec-to-spec movement could target, and thus comp-to-spec is attempted. The resulting AP is matched by */sound/* at the root. Merging of B triggers spec-to-spec movement of YP to Spec BP, and */sound/* again matches the result at the root, and thus spells out the whole BP.[2] Upon merging C, simple spellout fails, and thus spec-to-spec movement is attempted. The structure spells out successfully. However, instead of spelling out the whole BP, */sound/* now targets only the left branch YP – it has been partially overwritten.[3]

(3) a.



b.



If we consider the effects of partial overwrite in terms of a lexicalization table, as in ***Table 1***, we see that our branching vocabulary items are able to create an effect that looks similar to those created by backtracking; they do, however, create this effect without necessitating any undoing of the derivation, or a recursive spellout algorithm.

	XP	Y	A	B	C
AP	*sound*				
BP	*sound*				
CP	*sound*		*-sfx*		

Table 1 Lexicalization Table for Partial Overwrite.

2 The suffix */-sfx/* in (3b) that spells out [C[B[A]]], matches [B[A]] at this stage as well, yet [B[A]] is spelled out by */sound/*. I assume that the preference for */sound/* follows from general Nanosyntactic principles such as the preference for large targets over small ones. Concretely, this could be accomplished simply by having the spellout algorithm check whether the root can be matched before checking whether the right branch can be matched. (A left branch must have spelled out successfully prior to being merged/moved to the specifier position, hence checking whether the right branch can be matched is sufficient.)

3 This idea thus exploits the fact that Nanosyntax provides two ways to successfully spell out a phrasal node X: Either spell out X directly, or spell out both daughters of X.

Both branching vocabulary items and pointers are already part of standard Nanosyntax, and neither generates the computational complexity involved with backtracking.[4] In the interest of a minimalist theory of spellout, both in the sense of having few components, and in the sense of computational complexity, it is thus worth exploring whether the theory can get by without the fourth clause of the spellout algorithm. In what follows, I explore a few cases from Caha's (2019) recent analysis of case competition, as well as a relevant example of Pseudo-ABA patterns. I chose this work because it represents a recent state-of-the-art view of Nanosyntax, and because it contains an explicit argument in favor of backtracking as well as a variety of analyses that make crucial use of this technology.

Section 2 lays out an analysis of the Iron Ossetic pronoun data that Caha uses to motivate backtracking, and shows that a partial overwrite analysis can capture the facts. Section 3 discusses the case of the Digor Ossetic augment, arguing that the backtracking analysis does not, in fact, capture the data perfectly, but needs additional tools. Once such tools are in place, however, an analysis without backtracking is possible. Section 4 takes a look at Caha's (2019; 2020) size-based theory of declension class, and sketches a modified configurational theory of declension class. I suggest that, in addition to capturing the relevant data, branching vocabulary items offer an empirical advantage: Since they 'split' the functional sequence (f-seq) into a lower left and a higher right branch, a configurational theory may model different classes by locating their f-seq split in different positions. Section 5 takes a look at Pseudo-ABA patterns (Middleton 2020), and shows that we can similarly model these as subextraction without backtracking. Section 6 concludes.

2 Iron Ossetic pronouns

Caha motivates the backtracking component on the basis of a comparison between the Iron Ossetic first person plural pronoun *max* and the nominal paradigm of *fyd* 'father' in ***Table 2***. While the latter is affixed with *-y* in the accusative and genitive, the pronoun shows a syncretism between nominative, accusative, and genitive. That is to say, under standard nanosyntactic assumptions, *max* lexicalizes a larger structure than *fyd* does – one that is at least as big as the genitive (or the phrase headed by the feature which builds the genitive from the accusative, F3 in Caha's terminology).

Table 2 Iron Ossetic Pronoun vs Noun (Caha 2009: 74 ff, 119).

	1PL	father, SG
NOM	max	fyd-∅
ACC	max	fyd-y
GEN	max	fyd-y
INS (ABL)	max-æj	fyd-æj
DAT	max-æn	fyd-æn

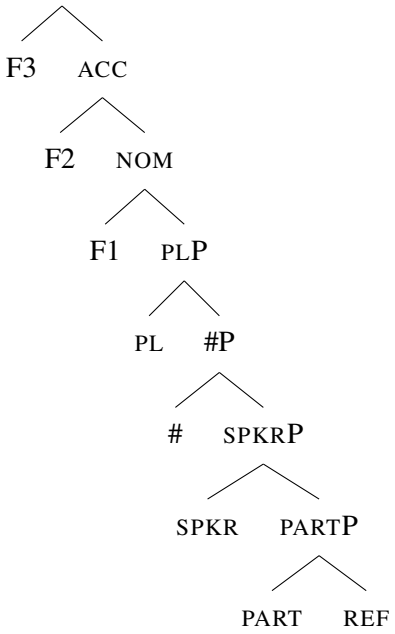
Note that both share the same case suffixes in the larger cases, such as the instrumental/ablative, or the dative. Since the instrumental *-æj* does not appear to co-occur with *-y*, we assume that they compete in some sense, i.e., under the standard assumptions, *-æj* is able to *overwrite* *-y* in the instrumental. For overwriting to take place, *-æj* must therefore lexicalize a superset of the features that *-y* lexicalizes, and *lexicalize the same bottommost feature* (or foot).

Under Caha's proposal, the conjunction of these facts leads to the necessity of backtracking: i) *max* lexicalizes a full GENP, as in (4a). ii) The suffix *-y* that we observe with *fyd* 'father' lexicalizes, minimally, the features F2, F3 that build the genitive from the nominative structure, and iii) *-æj* lexicalizes a superset of the features lexicalized by *-y*, up to the instrumental, as in (4b). Therefore, the foot of *-æj* is lower than the largest structure *max* can spell out – once we reach F4 on top of *max*, we must backtrack in order to anchor *-æj* (say, at F1).

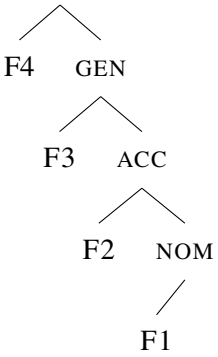
4 The lexicon contains only well-formed syntactic expressions, and branching vocabulary items correspond to expressions that can in principle be derived by the spellout algorithm. Pointers are needed independently for idioms, as well as certain types of cross-categorial syncretisms where a vocabulary item applies to the conjunction of sets (Caha & Pantcheva 2012, Blix 2021).

(4) The vocabulary (backtracking hypothesis)

a. *max* ⇔ GEN

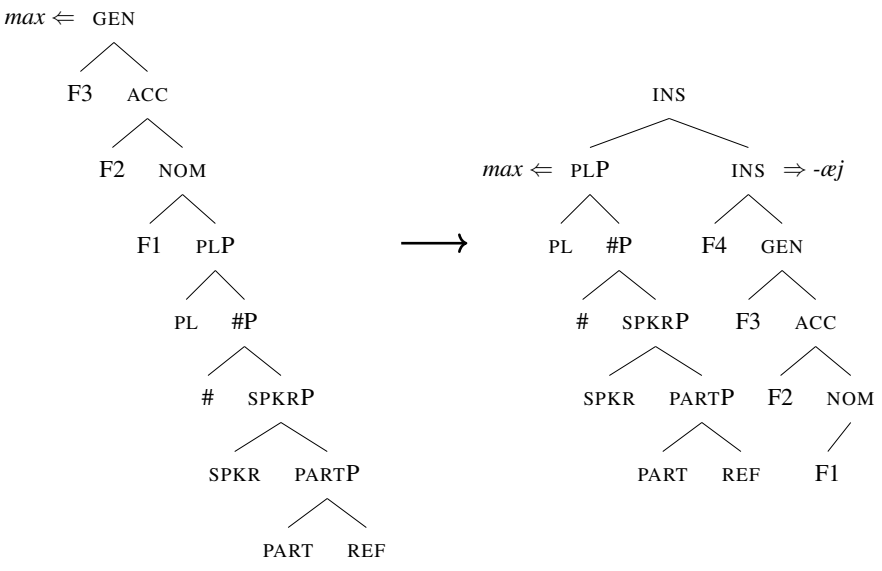


b. INST ⇔ *-æj*



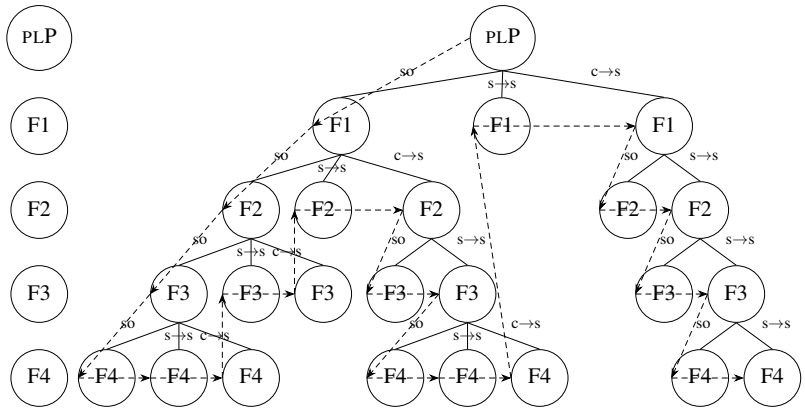
That is to say, backtracking is unnecessary until we reach F4 – at this point, the backtracking analysis suggests that the derivational path taken is incompatible with the spellout of F4, and it walks back step by step, attempting different derivations and checking whether they are compatible with the spellout of F4. The result of backtracking is indicated in (5) – though backtracking itself is not an operation on the tree, its results look just like standard phrasal movement, because it effects a series of standard movement operations.

(5) *Backtracking at F4*



The series of these operations can be visualized in the form of the decision tree in (6). Every node of the tree describes a derivational state, with the feature in the label being the most recently introduced one. The label of the edge connecting it to the mother node describes which of the three basic steps led to the derivational state, i.e., spellout (SO), spec-to-spec movement (s→s), or comp-to-spec movement (c→s). The dashed arrow describes the sequence in which these derivational states are built/explored.

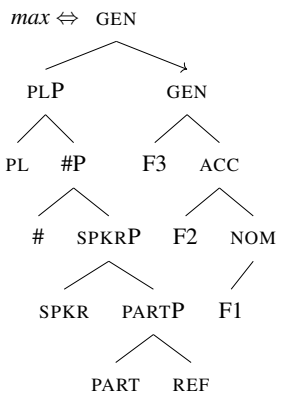
(6) *Decision tree with backtracking*



Essentially, the algorithm first merges F1–F3 on top of the PLP without any movement (SO). However, since this does not lead to a derivation that can successfully spell out F4 (i.e., all three options fail), it traces back its steps, first attempting to change the configuration of F3 where neither alternative to the SO route can be matched. It then reconsiders the configuration of F2. Here, the algorithm is forced to check whether a derivation with *-y* (which lexicalizes [F3[F2]]) may succeed, since F2 with F1P in its specifier can be spelled out by the hypothetical *max-y*. However, this derivation, too, fails to spell out F4, so a different configuration for F1 is explored next. Here, iteration of spec-to-spec movement finally succeeds and successfully derives *max-æj*.

The core idea of the alternative I would like to advance is that *max* is a branching lexical item, as in (7). The left branch corresponds to the phi-structure, and the right branch to the case structure. Crucially, the latter is embedded under a pointer, i.e., the vocabulary item can spell out any subtree of the tree that is formed by conjunction of the PLP with a subtree of the case structure.[5] Let us consider the stage at which we have built the PLP, and not yet merged any case features. Since a vocabulary item matches all subtrees of the tree it lexicalizes, and the PLP is a subtree of the tree in (7), *max* matches PLP, and thus simple SO is the path for the PL feature. After merging F1, spec-to-spec movement is unavailable (since PL does not have a specifier), but comp-to-spec movement is successful and *max* self-overwrites. F1 now has a specifier, PLP, and subsequently merging F2 triggers spec-to-spec movement, as does merging F3. This is due to the fact that *max* can match the respective resulting trees. That is to say, we retain the ability of *max* to spell out F1, F2, F3 that the backtracking approach offered, but we derive a configuration where, for spellout to be successful, the PLP must be in the specifier of F1/F2/F3 respectively.

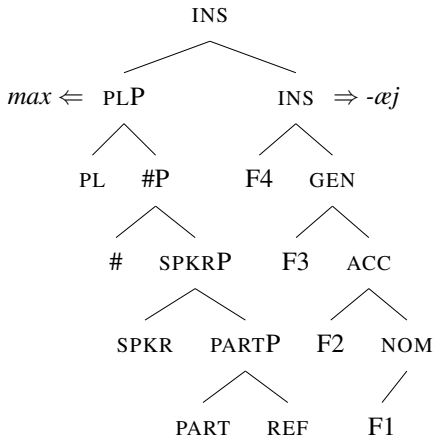
(7)



5 See Caha (2019: Chapter 4.6) for a relevant discussion of mixing grammatical features with pointers to lexical entries. Note that pointed to material is sometimes thought to be obligatorily present in the syntactic structure to be matched, whereas I take it to be optional here.

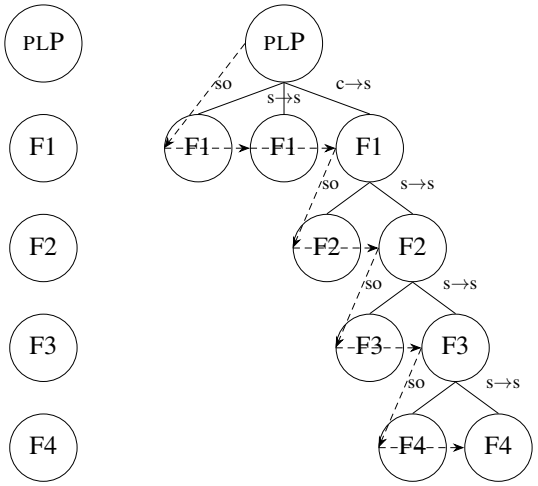
The crucial difference between this analysis and the backtracking one comes about upon merging F4, as in (8). In both cases, simple spellout after merging F4 fails, and spec-to-spec movement is attempted. In the backtracking analysis, this fails, and previous structure building operations are now undone. However, under the current hypothesis, the derivation does spell out successfully after spec-to-spec movement. PLP – now in the specifier of F4 – can be spelled out by *max*, because it is a subtree of the one *max* lexicalizes. The right branch is spelled out successfully by a suffix *-æj* that is identical to the one from the backtracking analysis. This *partial overwrite* analysis takes advantage of a theoretical ambiguity: A node is considered interpreted successfully if a) it was spelled out, or b) both its daughters were spelled out. That is to say, the structure [F4[F3[F2[F1]]]] can now be spelled out by an affix anchored at F1, with no backtracking being necessary: *max* is effectively overwritten with respect to spelling out F1, F2, and F3, but it remains the winning candidate for spelling out the PLP. Note in passing that the latter part is an accident of the lexicon (there is no smaller competitor for the spellout of PLP). I return to cases where partial overwrite leads to the emergence of a smaller competitor for the spellout of the left branch in section 5, which discusses Pseudo-ABA patterns of this type.

(8)



Both analyses derive the data successfully, and we can thus compare them in terms the number of derivational stages that had to be considered. The operation of the spellout algorithm in the analysis without backtracking can be visualized as in (9). Crucially, no backtracking means that there is no walking back up the tree and opening a new branch: At every level, there is a maximum of three nodes to explore, and consequently the derivational stages form a proper subset of those that the backtracking algorithm needs to explore. The maximum number of derivational stages that need to be explored simply grows linearly with the number of features in our tree.

(9) *Decision tree without backtracking*



Though the algorithms differ considerably in the way they achieve their aim, a look at a lexicalization table that describes which features are interpreted by which vocabulary item, as

in ***Table 3***, is a useful comparison to see similarities: The table describes *both* hypotheses equally well. In the backtracking analysis, the spellout of F1, F2, F3 by *max* was undone by returning to previous stages of the derivation and attempting alternative operations. However, the same result was achieved by the backtracking-less analysis by means of *partial overwrite* – just with a less costly algorithm.

Table 3 The lexicalization table for *max*.

	PLP	F1	F2	F3	F4
NOM	max				
ACC	max				
GEN	max				
INS	max	-æj			

3 The Digor Ossetic augment

Digor Ossetic numerals show an augment in the oblique cases, but not the structural cases. As the comparison in ***Table 4*** shows, however, the numerals take the same oblique case suffixes as a regular noun.

Table 4 Digor Ossetic augment (partial paradigm).

	two	horse
NOM	duuæ-∅	bæx-∅
ACC	duuæ-∅	bæx-i
INS (ABL)	duu-em-æj	bæx-æj

The core of Caha's analysis is represented in ***Table 5***. The augment *-em* lexicalizes a low head [X], but a zero affix that lexicalizes [F2[F1[X]]] overwrites it in the structural cases. Any affix that can spell out F3 and subsequent case features, however, must be anchored at F1 – i.e., backtracking occurs. In this case, backtracking has the further effect of uncovering *-em*: The cycles of spellout that overwrote *-em* are undone by backtracking, and thus the augment (re-) surfaces in the larger cases.

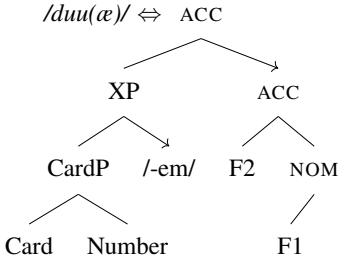
Table 5 The backtracking analysis.

	CardP	X	F1	F2	F3–F6
NOM	duu(æ)	-∅			
ACC	duu(æ)	-∅			
INS	duu(æ)	-em	-æj		

This analysis elegantly captures the distribution of the augment, and I cannot see a way of capturing the same data solely with a backtracking-less algorithm for spellout-driven movement. That is to say, an algorithm without backtracking may need to stipulate feature-driven movement to derive such lexicalization tables. Using such an auxiliary tool/assumption, we could once again postulate a branching vocabulary item with a pointer, as in (10):

(10) The vocabulary (to be revised)

a. */duu(æ)/* ⇔ ACC



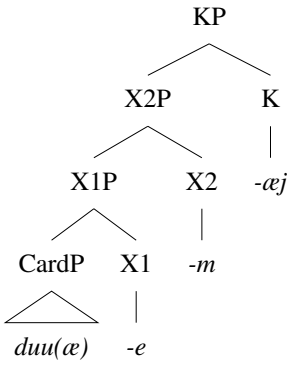
b. */-em/* ⇔ XP
/
X

The derivation continues in the usual way, and upon merging F2, we derive the full tree that *duu(æ)* lexicalizes. The core idea is the following: If, by stipulation, F3 has an EPP feature for CardP, then such a feature may lead to stranding of the structure above CardP, namely X. Once CardP has stranded X, the interpretation by *-em* that was hitherto overwritten by the idiom, resurfaces. As in the cases above, however, *duu(æ)* remains the spellout for the moved CardP, and thus we derive the augment as a resurfacing effect, and capture the data in a way reminiscent of Caha's proposal, but without backtracking.

There is reason to believe that an operation not driven by spellout is needed to derive the Ossetic augment on independent grounds, even for the backtracking analysis, and I will lay out in more detail both why and how to derive the relevant facts with vocabulary entries such as (10) in the remainder of this section.

Let me begin by elaborating on an issue with Caha's analysis that cannot be captured with backtracking alone: Nanosyntax generally takes zero-affixes to be reason for suspicion (see in particular the arguments about zero-distribution in Caha et al. 2019), but they are certainly not reason enough to reject an analysis. There is, however, a deeper problem that is laid bare by Caha's subsequent comparison with the augment in the pronominal and demonstrative system. In particular, Caha (2019: 142ff) argues highly convincingly that the augment must in fact be analyzed as a bimorphemic structure *-e-m*. He revises the structure as represented in the simplified representation in (11) (his 24, p. 144).

(11)



A bimorphemic augment however, cannot be the result *solely* of backtracking undoing the work of a zero affix overwriting the augment. To see why, consider the original analysis in ***Table 5***, in which a zero affix overwrites the augment in the nominative and accusative cases, but backtracking undoes this in the oblique cases. The augment (re-)surfaces in the larger cases, since the spellout cycles that overwrote it were undone. ***Table 6*** shows why this solution is impossible for a bi-morphemic augment: The zero affix would continue to be a better candidate for the spellout of X1, X2, regardless of backtracking to F1, and no augment would re-surface. This limitation follows from core principles of Nanosyntax (the preference for few large morphemes over many small ones, as encoded in the preference for spec-to-spec movement, or the biggest wins theorem): Backtracking that results in the re-emergence of a smaller vocabulary item in place of a bigger one is *necessarily* limited to the re-emergence of a single affix.

		CardP	X1	X2	F1	F2	F3-F6
	NOM	duu(æ)		-∅			
	ACC	duu(æ)		-∅			
*	INS	duu(æ)	-e	-m		-æj	
☞	INS	duu(æ)		-∅		-æj	

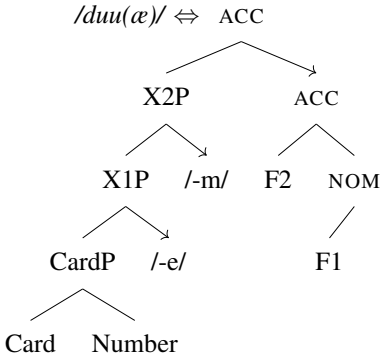
Table 6 The Problem of the Bimorphemic Augment.

The backtracking analysis thus needs some tool(s) beyond backtracking itself to derive the data. One possible analysis employing standard syntactic tools would be to augment the backtracking

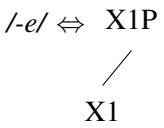
analysis with feature driven movement, as above. If F3 triggers the displacement of X1P, it would block the ability of *-Ø* to overwrite *-e* and *-m*, because they are no longer in a phrasal configuration that could be matched by *-Ø*. Once such feature driven movement is brought into the account, however, we can make due without backtracking, and the analysis sketched above can easily be modified to deal with a bi-morphemic augment.[6]

The non-backtracking analysis I would like to propose builds on Caha's analysis in (11), but includes the fact that there is no overt case marking in the structural cases. Both the augments and the case structure can be analyzed as being spelled out by right branches of the vocabulary item i.e., we simply iterate the approach from before, as in (12).

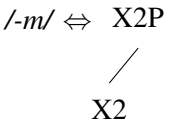
(12) a.



b.

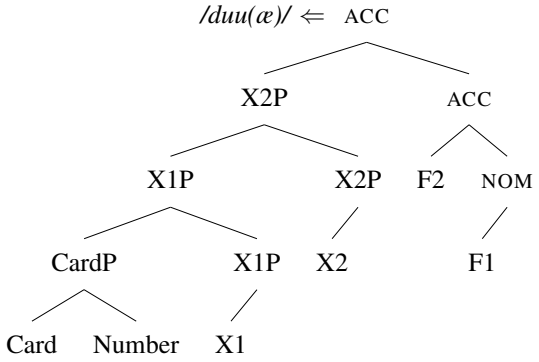


c.



Simply put, this vocabulary item is a complex idiom that can overwrite a phrase that contains the augments */-e/* and *-m/*, as well as any subtree of [F2[F1]]. Since Nanosyntax maximizes targets, it will therefore result in the corresponding derivation after the merger of F2, as in (13):[7]

(13)



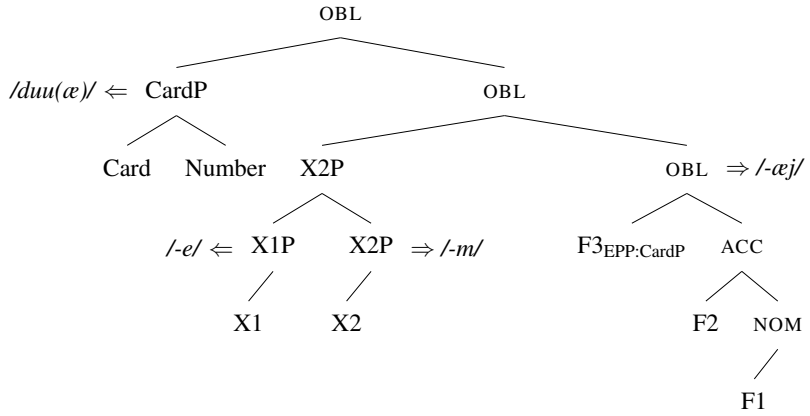
6 That being said, the backtracking analysis still has the advantage of not requiring a stipulation of feature-driven movement in cases where a monomorphemic augment appears. For instance, Caha's (2019:304) analysis of the declension of Russian *žen-a* 'woman' also features the re-surfacing of a single smaller vocabulary item in the instrumental. If this is the correct analysis, the backtracking-less story will need to stipulate a feature-driven movement where the backtracking analysis does not, and would operate in a manner parallel to the one advanced here. An anonymous reviewer helpfully points out that parts of the Finnish case paradigm also feature the resurfacing of a single morpheme, as discussed recently in Kloudová (2020).

7 The derivation occurs as follows: 1. Merge(Card, Number), spell out 2. Merge(X1,CardP), c→s 3. Merge(X2,X1P), c→s 4. Merge(F1,X2P), c→s 5. Merge(F2,F1P), s→s.

The vocabulary item *duu(æ)* matches the whole tree at the root at every single step.

Next, let us again assume that there is an EPP-feature associated with F3 that attracts CardP. This results in feature-driven movement of CardP to the specifier of F3. However, the remainder cannot spell out the newly merged F3, and hence the spellout algorithm continues operating as usual: spec-to-spec movement targets the remnant X2P, and moves it from the specifier of F2 into the specifier of F3. This allows for the spellout of F3 in a constituent [F3[F2[F1]]], essentially parallel to the case with *max* above.[8]

(14)



The case structure thus spells out the same way as above; however, the feature-driven movement has additionally extracted CardP from a larger idiom. Because *duu(æ)* can no longer match X1P and X2P (since they are no longer forming a phrasal node that can be matched by *duu(æ)*), the *pointed to* vocabulary items, i.e., the overwritten idiom chunks re-appear. The analysis is thus quite similar in spirit to Caha's, but derives it with different means.

None of this is to suggest that this ought to taken as a proper analysis of the Digor Ossetic Augment. The point was to show that a) the backtracking analysis runs into an issue that will require a further stipulation, such as feature driven movement, and that b) once such an allowance is made, an analysis without backtracking becomes possible as well. That is to say, the augment case does not offer a good case for the *necessity* of backtracking.

4 Size & Configuration: Declension Class

Caha (2019; 2020) employs backtracking to develop a *size theory of declension classes*: A declension class is determined by the size of the root, and the consequences for anchoring affixes and the point at which affixes need to start backtracking. This is a highly attractive theoretical development, for a variety of reasons. In featural theories of declension class, the theoretical status of the features generally remains unclear – there is no reason to suspect that declension class represents syntactically active features (for instance, gender may be the target of agreement/concord processes, but declension class is not). Yet, a featural conception in a late insertion model requires post-syntactic reference to them, i.e., they seem to violate the inclusiveness condition (Chomsky 1995). In contrast, the size theory does not have declension class features, and models declension class purely in terms of the cyclicity of exponence targeting feature sets (trees) of varying sizes.

A simplified example of Caha's theory is provided in ***Table 7*** (I refer the reader to Caha's work for the actual details; the overview here is meant to lay out the spirit of the proposal). Both *zavód* 'factory' and *mést-o* 'place' share an identical f-seq, but the former lexicalizes are larger part of this structure, reaching all the way up to F2, while the latter lexicalizes only #P. This leads to *mést-o* 'place' having affixes that are anchored at the F1 node in the nominative (F1P) and accusative (F2P), while *zavód* 'factory' is able to spell these out as part of the root. However, for

8 It is of course essential that CardP end up in the outer specifier position in order to derive the proper linear order. There are two obvious paths to ensure this: Either feature-driven movement precedes spellout driven movement and the latter is subject to *tucking in*, or spellout-driven movement precedes feature-driven movement, and we subextract from the inner specifier position to an outer specifier position. In either case, subsequent spellout driven movement may have to move *both* specifiers to create viable spellout targets (and presumably do so in an order preserving fashion). To my knowledge, the relation between feature driven movement and spellout driven movement has not been discussed at much depth (at least publicly), so I will leave these details aside, merely noting that there are options to ensure the desired results.

both items, subsequent affixes must be anchored at the F1 level, i.e., the derivation of *zavód-a* (GEN) involves backtracking.

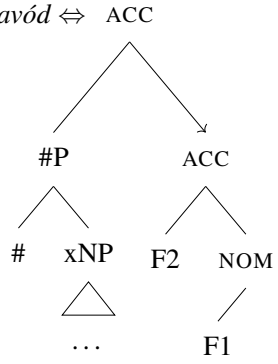
	xNP	#	F1	F2	F3	F4	F5	F6
NOM	zavód							
ACC	zavód							
GEN	zavód		a					
LOC	zavód		e					
DAT	zavód		u					
INS	zavód		om					
	xNP	**#**	**F1**	**F2**	**F3**	**F4**	**F5**	**F6**
NOM	mést		o					
ACC	mést		o					
GEN	mést		a					
LOC	mést		e					
DAT	mést		u					
INS	mést		om					

Table 7 Simplified lexicalization tables for Russian *zavód* 'factory' and *mést-o* 'place'.

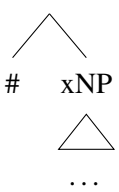
I believe, however, that the featureless conception of declension class can not only be derived without backtracking, but that such a theory may in fact improve upon the size theory of declension class.[9] Consider first how we can replicate the basic aspect of these lexicalization tables, as in (15). As in the previous discussion, this branching conception of *zavód* can spell out nominative and accusative (hence the absence of dedicated surface morphemes vis-a-vis the *-o* in *mést-o*). The fact that it is a branching vocabulary item with a pointer, however, allows for *partial overwrite*, i.e., it is compatible with the lexicalization tables introduced above without any need for backtracking.

(15) *An alternative without backtracking*

a. *zavód* ⇔ ACC



b. *mést* ⇔ #P



9 Technically, both conceptions of declension class make slightly different predictions. Insofar as the backtracking-less theory of spellout is a sub-theory if the one with backtracking, however, the tools of the backtracking-less theory can, of course, be employed to create a theory that can encompass the predictions of both theories.

By employing branching vocabulary items to model declension classes, we can explore the possibility of a *configurational theory of declension class*: If vocabulary items fall into declension classes, they vary by size (and possibly f-seq, in case they have different gender, but I leave this aside for now), as in the size theory. However, they may *additionally* vary in the point at which the f-seq is split into a left and a right branch (and if it is split at all). There is data that suggests that such an approach is necessary, even for a theory that does adopt backtracking. Consider for instance the two strong feminine consonantal inflection classes from Icelandic in ***Table 8*** (Müller 2005: 232), or the Greek declension classes VII and VIII in ***Table 9*** (Alexiadou & Müller 2008: 120).

Table 8 Strong feminine consonantal declension classes (Icelandic).

	geit **'goat' (FC1)**	***vík*** **'bay' (FC2)**
NOM SG	geit	vík
ACC SG	geit	vík
DAT SG	geit	vík
GEN SG	geit-ar	vík-ur

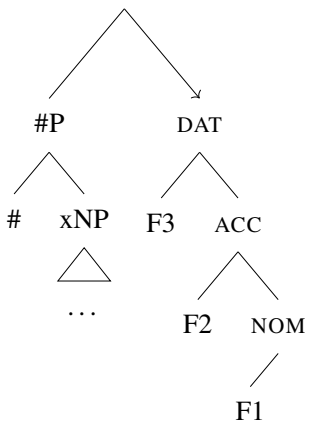
Table 9 Neuter declension classes VII and VIII (Greek).

	spiti **'house' (VII$_n$)**	***soma(t)*** **'body' (VIII$_n$)**
NOM SG	spiti	soma
ACC SG	spiti	soma
GEN SG	spitj-u	somat-os

In both the Greek and the Icelandic case we can observe the following: For two declension classes, nouns of the same gender show zero morphology for the lower part of the case hierarchy. As usual, we conclude that Icelandic *geit* 'goat' and *vík* 'bay' are able to spell out the whole phrase corresponding to a dative. In parallel fashion we conclude that Greek *spiti* 'house' and *soma* 'body' lexicalize the whole accusative structure. They do, however, take different suffixes for the larger cases. Under an analysis that relies solely on backtracking, this is unexpected: They spell out structures of the same size, and presumably share the same f-seq.[10] Under the configurational perspective laid out above, however, two vocabulary items may lexicalize the same set of features, but in a different syntactic configuration. This is not the place to develop a serious analysis of these inflectional systems, but let me illustrate how one can account for these type of data in principle with vocabulary items such as those in (16):

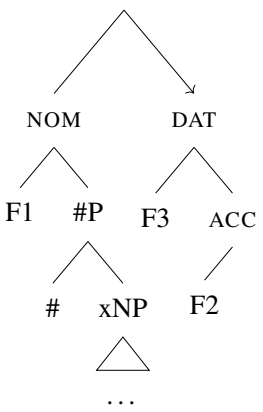
(16) *Configurational Declension Classes*

a. *geit* ⇔ DAT



10 The latter is, of course, not necessarily a given, but examples like these are easy to find, and reducing all such cases to differences in the f-seq risks being indistinguishable from returning to a theory that employs declension class features.

b. *vík* ⇔ DAT



Assuming two suffixes *-ar* [F4[F3[F2[F1]]]] and *-ur* [F4[F3[F2]]] now explains how these two classes differ besides sharing an identical f-seq and being of equal size, as in the lexicalization tables in ***Table 10***: The partial overwrite of the right branch requires a different anchor. While *geit* restricts overwriting of the right branch to affixes anchored at F1, *vík* restricts it to affixes anchored at F2. Crucially, backtracking on its own is unable to derive lexicalization tables such as those in ***Table 10***: Given the existence of *-ur*, backtracking would never result in backtracking all the way to F1. In a configurational theory, however, identity in size does not necessarily imply identity of inflection class.

	xNP	#	F1	F2	F3	F4
NOM	geit					
ACC	geit					
DAT	geit					
GEN	geit			ar		
	xNP	**#**	**F1**	**F2**	**F3**	**F4**
NOM	vík					
ACC	vík					
DAT	vík					
GEN	vík			ur		

Table 10 (Toy) lexicalization tables for Icelandic *geit* 'goat' and *vík* 'bay'.

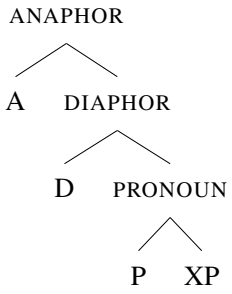
I believe that I have shown that a backtracking-less theory can in principle underly a featureless theory of declension classes. Obviously, this configurational theory of declension classes will need to be put to a broader empirical test. Independently of the question of backtracking, however, I believe to have shown that thinking of declension class not only in terms of size but also in terms of configuration is likely be necessary to account for relations between declension classes – whether a theory deploys backtracking or not.

5 Pseudo-ABA

Middleton (2020; 2021) provides us with a final case of interest: Pseudo-ABA patterns. In her investigation, Middleton uncovers strong cross-linguistic evidence for a general *ABA restriction for pronouns/diaphors/anaphors, i.e., a ban on using the same form for anaphors and pronouns to the exclusion of diaphors. She derives the restriction from a structure along the lines of (17).[11]

11 Middleton (2020; 2021) defines an anaphor as a locally bound variable, a diaphor as a non-locally bound variable, and a pronoun as a free variable. The lexical entries for English pronouns such as *they/she/he* etc are systematically ambiguous between diaphors and anaphors (an AAB pattern) – or, in terms of the theory adopted here, they lexicalize [D[P[XP]]] and English lacks a smaller competitor.

(17)



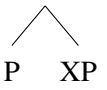
While there are no *ABA patterns, however, there are Pseudo-ABA patterns, A-B-A + x, as shown in (18), from Middleton (2020).[12]

(18) *Three Pseudo-ABA Patterns (Middleton 2020)*

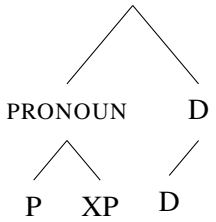
	PRONOUN	DIAPHOR	ANAPHOR	
a.	wén	jì	ə̀wénə̀ wén	(Babanki)
b.	avan	tan	avanavan	(Malayalam)
c.	rẹ́	òun	ara rẹ́	(Yoruba)

In order to account for Pseudo-ABA patterns, we extend our approach slightly. As before, the effect can be captured as a partial overwriting; however, this time a smaller candidate re-emerges. Consider the vocabulary items for Malayalam, as in (19). In the same spirit as the analyses above, the key to making this system work lies in the fact that (19b) enforces comp-to-spec movement subsequent to merging D. In this way – and unlike a vocabulary item that lexicalizes [D[P[XP]]] in a strict head-complement sequence – *tan* makes this derived specifier available for subsequent spec-to-spec extraction.

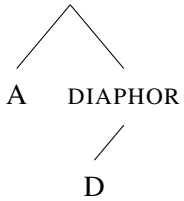
(19) a. *avan* ⇔ PRONOUN



b. *tan* ⇔ DIAPHOR



c. /REDUPL/ ⇔ ANAPHOR

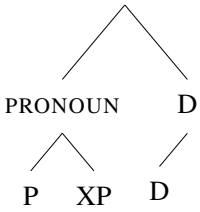


That is to say, *tan* can spell out the whole diaphor, but subsequent merger of the A feature that derives an anaphor leads to sub-extraction of the pronoun. This is shown in (20): In (20a), *tan* is able to spell out the root after the spellout algorithm attempts comp-to-spec movement (after simple spellout and spec-to-spec movement were unsuccessful). Upon merging A (20b), the spellout algorithm can now move this derived specifier by spec-to-spec movement. This movement results in a structure that can be spelled out, with the right branch being matched by the reduplicating vocabulary item REDUPL (19c). On the left branch, the previously overwritten

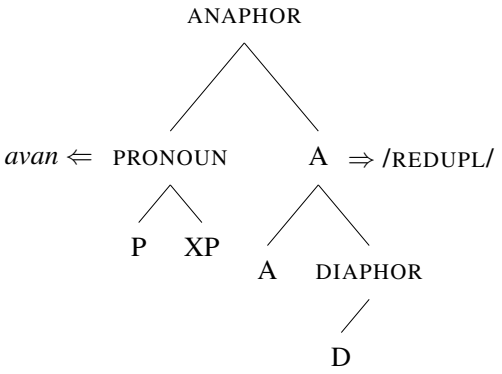
12 While comparable cases are rare in the degrees of adjectives, Bulgarian offers a similar pattern with *mnogo* 'much/many': *mngogo*(POS) – po-veče(CMPR) – naj-mnogo(SPRL), see Bobaljik (2012: 126).

avan re-emerges as the spellout of the PRONOUN structure.[13] The only real difference to the previous analysis is that the left branch of *tan* has an independent candidate for its spellout, but the mechanisms are otherwise identical.

(20) a. *tan* ⇐ DIAPHOR



b.



Once again, an analysis in which a vocabulary item can force movement prior to spellout, despite targeting the root, allows for an account that can model Pseudo-ABA effects without backtracking.

6 Conclusion

In this paper, I have offered an analysis of a variety of phenomena that eschews the notion of *backtracking*. Instead, vocabulary items were argued to take a highly active role in structuring the course of a derivation, frequently enforcing movement even when the spellout target was the root node – a possibility that has not been explored at great depth within Nanosyntax, but one whose availability follows from standard assumptions. I have shown that such vocabulary items can result in *partial overwrite*, and that a variety of empirical phenomena that appeared to necessitate backtracking can in fact be analyzed without backtracking, once such vocabulary driven movement is fully exploited.

I have further argued that regardless of the adoption of *backtracking*, a configurational extension of the size theory of declension class is likely to be necessary, and that such an extension requires the tools I employed as an alternative to backtracking. That is to say, a theory with backtracking will still need pointers, branching vocabulary items, and partial overwrite, but a theory with these features does not necessarily need backtracking.

These results are of core importance, insofar as the proposed analyses all manage to keep the computation of spellout in the realm of a linear algorithm. I believe that this should be a welcome simplification of the theory, and any data that might suggest the need for the adoption of backtracking should be scrutinized carefully before we conclude that the realm of linear algorithms is one we have to abandon.

Competing interests

The author has no competing interests to declare.

13 Note that I am abstracting away from linear order here; the astute reader may notice that this assumption isn't altogether innocent, as the re-emerging smaller element has to be to the right, rather than to the left in Babanki as well as Yoruba – i.e., the analysis here might require the suffix to undergo subsequent movement above the root.

Author affiliation

Hagen Blix *orcid.org/0000-0002-8838-9393*
New York University, US

References

Alexiadou, Artemis & Gereon Müller. 2008. Class Features as Probes. In Asaf Bachrach & Andrew Nevins (eds.), *Inflectional identity* (Oxford Studies in Theoretical Linguistics) 18. 101–155. Oxford: Oxford University Press.

Blix, Hagen. 2021. Spans in South Caucasian agreement: Revisiting the pieces of inflection. *Natural Language & Linguistic Theory* 39(1). 1–55. DOI: *https://doi.org/10.1007/s11049-020-09475-x*

Bobaljik, Jonathan David. 2012. *Universals in comparative morphology: Suppletion, superlatives, and the structure of words* (Current Studies in Linguistics). Cambridge, Mass.: MIT Press. DOI: *https://doi.org/10.7551/mitpress/9069.001.0001*

Caha, Pavel. 2019. Case competition in Nanosyntax: A study of numerals in Ossetic and Russian. Lingbuzz/004875.

Caha, Pavel. 2020. Modeling declensions without declension features. Lingbuzz/005537.

Caha, Pavel, Karen De Clercq & Guido Vanden Wyngaerd. 2019. The Fine Structure of the Comparative. *Studia Linguistica* 73(3). 470–521. DOI: *https://doi.org/10.1111/stul.12107*

Caha, Pavel & Marina Pantcheva. 2012. Datives Cross-Linguistically.

Chomsky, Noam. 1995. *The Minimalist Program.* Cambridge University Press.

Collins, Chris & Edward Stabler. 2016. A Formalization of Minimalist Syntax. *Syntax* 19(1). 43–78. DOI: *https://doi.org/10.1111/synt.12117*

Kloudová, Veronika. 2020. *The interaction of functional morphemes inside the nominal phrase* (LINCOM Studies in Theoretical Linguistics 63). München: Lincom.

Middleton, Hannah Jane. 2020. **ABA syncretism patterns in pronominal morphology*. London, UK: University College London PhD Thesis. DOI: *https://doi.org/10.1007/s11525-021-09377-7*

Middleton, Jane. 2021. Pseudo-ABA patterns in pronominal morphology. *Morphology*. *https://doi.org/10/gjj2vw*. DOI: *https://doi.org/10.1007/s11525-021-09377-7*

Müller, Gereon. 2005. Syncretism and iconicity in Icelandic noun declensions: A Distributed Morphology approach. In Geert Booij & Jaap van Marle (eds.), *Yearbook of Morphology 2004,* 229–271. Dordrecht, Netherlands: Springer. DOI: *https://doi.org/10.1007/1-4020-2900-4_8*

Starke, Michal. 2018. Complex Left Branches, Spellout, and Prefixes. In Lena Baunaz, Karen De Clercq, Liliane M. V. Haegeman & Eric Lander (eds.), *Exploring nanosyntax* (Oxford Studies in Comparative Syntax), New York: Oxford University Press. DOI: *https://doi.org/10.1093/oso/9780190876746.003.0009*

TO CITE THIS ARTICLE:
Blix, Hagen. 2021. Phrasal Spellout and Partial Overwrite: On an alternative to backtracking. *Glossa: a journal of general linguistics* 6(1): 62. 1–17. DOI: *https://doi.org/10.5334/gjgl.1614*

Submitted: 01 March 2021
Accepted: 21 April 2021
Published: 07 May 2021

Glossa: a journal of general linguistics is a peer-reviewed open access journal published by Ubiquity Press.