Pinzin, Francesco & Mattiuzzi, Tommaso. 2025. Word-order information in the Lexicon. *Glossa: a journal of general linguistics* 10(1). pp. 1–49. DOI: https://doi.org/10.16995/glossa.18521

Open Library of Humanities

Word-order information in the Lexicon

Francesco Pinzin, Università degli Studi di Padova Goethe Universität Frankfurt am Main, francesco.pinzin@unipd.it

Tommaso Mattiuzzi, Goethe Universität Frankfurt am Main, mattiuzzi@em.uni-frankfurt.de

We propose that crosslinguistic word-order variation (Greenberg 1963; Cinque 2005; Abels 2016) reflects information stored in the lexicon in the format of Lexical Items, and that the operations yielding word-order are the same that guide lexicalisation more generally in Nanosyntax (Starke 2009). We show that this lexicalisation-based approach shares the empirical coverage of current analyses (Cinque 2005; Abels & Neeleman 2012; Cinque 2023) without making reference to traditional syntactic movement. This spares the system from stipulating semantically vacuous structural dependencies triggering movement, and correctly separates subtypes of movements subject to different constraints. The proposal also favours a novel perspective on how the relevant Lexical Items are acquired and activated during processing, shedding new light on the different typological frequency of word-order patterns and structural priming.

1 Introduction

Crosslinguistic variation in basic word-order as in (1) necessarily depends on differences in the word-order information stored in the speakers' competence.

(1) a. black dog
b. *Italian*
cane nero
dog black
'black dog'

It is crucial for a formal approach to language to make explicit where such information is stored, in what format, and how it interacts with the other components of the grammar.

Beyond formal adequacy, any model of word-order variation must also account for two core typological observations: i) some word-order possibilities are not attested, and ii) among the attested orders, some are overwhelmingly more frequent, while others are typologically rare. These longstanding observations (Greenberg 1963) have been at the centre of much theoretical work, starting from the analysis of the word-order possibilities in the nominal domain by Cinque (2005; 2023). To make a concrete and widely adopted example, among all the 24 logical ordering possibilities between demonstrative (Dem), numeral (Num), adjective (Adj) and noun (N), only 14 are attested (absolute constraint). Moreover, among these 14 attested word-orders, some are extremely frequent, while others are rare (typological tendency). Following Greenberg's (1963) label, this is usually referred to as Universal 20 (**Table 1**).

In this contribution, we propose that word-order information is stored in the lexicon, in the format of independently needed ontological entities, i.e. the lexical entries (or Lexical Items, LIs henceforth) of a given language. We take these entities and their interaction with the rest of the grammar to be regulated by a modified version of the mechanisms proposed in Nanosyntax (Starke 2009; 2014; Caha 2009; 2019), whereby a syntactic structure is licensed when all its labelled constituents are matched by an LI. Unlicensed syntactic structures undergo a fixed sequence of syntactic movements to find a match (the Lexicalisation Algorithm, see Section 3).

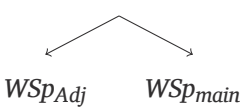
In line with Nanosyntax, we assume that LIs can bind together three types of information: form (externalisation), syntax (contextual distribution) and extra-grammatical meaning (2). Since there is no principled reason why all three such bits of information should always be present, we take one or more to be potentially missing. This enables us to situate in the lexicon what we label second-order LIs, which are ontologically equal to (2), but only contain syntactic information regulating the structural arrangement of a given constituent with respect to the rest of the syntactic derivation. To make a concrete example in connection to (1), the English lexicon, but not the Italian lexicon, will contain the LI in (3).

a.	✓	Dem	Num	Adj	N	very many languages
b.	✓	Dem	Num	**N**	Adj	many languages
c.	✓	Dem	**N**	Num	Adj	very few languages
d.	✓	**N**	Dem	Num	Adj	few languages
e.	*	Num	Dem	Adj	**N**	Ø
f.	*	Num	Dem	**N**	Adj	Ø
g.	*	Num	**N**	Dem	Adj	Ø
h.	*	**N**	Num	Dem	Adj	Ø
i.	*	Adj	Dem	Num	**N**	Ø
j.	*	Adj	Dem	**N**	Num	Ø
k.	✓	Adj	**N**	Dem	Num	very few languages
l.	✓	**N**	Adj	Dem	Num	few languages
m.	*	Dem	Adj	Num	**N**	Ø
n.	✓	Dem	Adj	**N**	Num	very few languages
o.	✓	Dem	**N**	Adj	Num	many languages
p.	✓	**N**	Dem	Adj	Num	very few languages—possibly spurious
q.	*	Num	Adj	Dem	**N**	Ø
r.	✓	Num	Adj	**N**	Dem	very few languages
s.	✓	Num	**N**	Adj	Dem	few languages
t.	✓	**N**	Num	Adj	Dem	few languages
u.	*	Adj	Num	Dem	**N**	Ø
v.	*	Adj	Num	**N**	Dem	Ø
w.	✓	Adj	**N**	Num	Dem	very few languages
y.	✓	**N**	Adj	Num	Dem	very many languages

Table 1: Attested and unattested orders of nouns and modifiers (Cinque 2005: 320–321).

(2) LI_{45}: ⟨ /ˈblæk/, ADJP, [CONCEPT] ⟩

(3) LI_{58}: ⟨ ADJP ⟩



The LI in (3) stores information concerning the structural arrangement of an adjectival constituent labelling the root as AdjP and the constituent it is merged with (i.e., the rest of the syntactic derivation, or *main* derivation). Both constituents are referred to *via* the workspaces (*WSp*) in which they are built (see Section 3.1 for details). The presence of (3) in the lexicon of a language licenses the structure [$_{\text{AdjP}}$ WSp_{Adj} WSp_{main}]. We assume a universal linearisation procedure whereby the content of WSp_{Adj} (e.g. *black*) is linearised before the content of WSp_{main} (e.g. *dog*), deriving (1a). Absence of such an LI triggers instead the set of syntactic movements defined by the Lexicalisation Algorithm, deriving cases like (1b). Word-order variation is then reduced in our proposal to the content and distribution of second-order LIs similar to (3), which varies crosslinguistically, as expected for any element of the lexicon.

Crucially for testing the empirical adequacy of the approach, the movements defined by the Lexicalisation Algorithm can be shown to restrict the derivational options to the typology proposed in Cinque (2005; 2023). This replicates its main result, namely its ability to generate all and only the attested patterns (Section 4).

Unlike existing approaches that formally model typological observations like the one in **Table 1** (Cinque 2005; 2023; Abels & Neeleman 2009; 2012), in our proposal syntactic movement deriving basic word-order is a reaction to a lexicalisation problem and does not hinge on a structural dependency between two positions (whether fully – Cinque 2005; 2023 – or partially – Abels & Neeleman 2009; 2012; see Section 2). In other words, the derivation of basic word-order makes no reference in our system to traditional syntactic movement. This provides a rationale for why the operations involved are semantically *meaningless* (Cinque 2023: §5.4) and obey different constraints than *bona fide* syntactic movement (as noted in Abels & Neeleman 2009; 2012). Moreover, this system allows to maintain a universal linearisation procedure (as Kayne's 1994 *Linear Correspondence Axiom* adopted in Cinque 2005; 2023) and a single locus for syntactic variation, i.e. the lexicon.

On the empirical side, the substantiation of 'word-order rules' as LIs with a specific structure and a contextual derivational activation opens new questions on how these objects are acquired and their role in processing. We argue that this enables a novel perspective on two empirical domains: 1) it helps highlight a concrete acquisitional rationale for the higher frequency of word-orders that show homomorphy with the abstract functional hierarchy (Martin et al. 2024), and 2) it suggests that what is activated under structural priming is nothing else than the lexical entities that license the relevant configurations, a hypothesis which fits with previous results in the literature on syntactic priming (e.g., the possibility of priming syntactic information autonomously and the lexical-boost effect, see Section 6.2) and lends itself to further experimental investigation.

The article is structured as follows: Section 2 situates our proposal with respect to the existing syntactic approaches to word-order variation. In Section 3, we describe the general properties

of the Nanosyntactic system we build on and our reconceptualisation, introducing the logic of second-order LIs. In Section 4, we show how this system captures the U20 generalisation. Section 5 elaborates on different possible implementations of the system, addressing some open issues. In Section 6 we outline how the approach enables a new perspective on typological markedness and structural priming, which we argue paves the way for further empirical testing. Section 7 concludes.

2 The format of word-order instructions, previous approaches and Nanosyntax

Two main families of syntactic approaches to word-order variation are present in the literature. The first grants a prominent role to the syntactic computation (Kayne 1994; Cinque 2005; 2010; Cinque & Rizzi 2010), the second to the interface with the externalisation module (Chomsky 2001; 2013; Abels & Neeleman 2009; 2012; Manzini Submitted). We refer to the first set of approaches as "syntax-based", to the latter as "externalisation-based".

In "syntax-based" approaches, the mapping from syntactic structure to linear word-order is universal, as per the *Linear Correspondence Axiom* (LCA; Kayne 1994). Hence, a given structure has only one possible linearisation. At the same time, the Cartographic perspective (Rizzi 1997; Cinque 1999; 2005; 2010; Cinque & Rizzi 2010, a.o.) generally assumed by these approaches posits a universal order of merge among functional elements. As a consequence, any difference in linear word-order reflects a difference in the syntactic configurations derived from the same universal structure. In Cinque (2005; 2023), these differences result from syntactic movement. Concretely, (1a) is different from (1b) because *black* asymmetrically c-commands *dog* in English, while the opposite holds for Italian *nero* and *cane*. Assuming a universal order of merge whereby the adjective is merged as the specifier of a functional projection dominating the noun, this is captured by taking the noun to move across the adjective in Italian, but not in English.

The core of the proposal originally put forth in Cinque (2005) is that the absolute constraints on the possible word-orders shown in **Table 1** follow from this set of assumptions if the relevant type of movements are subject to the requirement in (4) (see Cinque 2023 for details, and Cinque 2005; 2010 for previous formalisations).

(4) (Cinque 2023: 4)
… only the head of each (sub)hierarchy can move (by itself or in one of the possible ways movement can take place).

Which options a given language takes among the permissible movements is encoded by dedicated "pied-piping" features, which therefore represent the format in which word-order information is stored. In Cinque's formalisation, each functional projection is followed by an AGR projection,

and the "pied-piping" feature in this projection imposes a given type of movement. Hence, all movements deriving linear word-order happen within core syntax, in that both the movement and its trigger (the "pied-piping" feature) are present within this module (Manzini Submitted). "Pied-piping" features lack any semantic interpretation, and the movements they trigger do not alter semantic scope relations. On these grounds, such movements are labelled *meaningless*, in opposition to *meaningful* movements as the ones connected to Wh, Focus, etc. dependencies (Cinque 2023: §5.4).

Crucially, meaningless movements involve a structural dependency between the base and the derived position of the moved element like any other instance of syntactic movement. This is problematic, to the extent that meaningless and meaningful movement obey different constraints (Abels & Neeleman 2009; 2012). Concretely, the meaningless version of syntactic movement must include the lexical head of the relevant functional sequence (4), and must be allowed to violate anti-locality (Grohmann 2011; Abels 2003) and strand pied-piped material.[1] Conversely, the same does not hold for the meaningful version, which can be independently argued to obey anti-locality and other constraints proposed in the literature (see Abels & Neeleman 2012: §5.2).

Building on this, Abels & Neeleman (2009; 2012) propose to abandon the LCA and allow variation at externalisation/Transfer (Chomsky 2001; 2013; Manzini Submitted), that is at the moment in which syntactic structure is linearised. Such variation takes the format of *linearisation statements* like (5) (where AP = AdjP).

(5) (Abels & Neeleman 2012: 66)
In the structure $[_{NP}$ AP NP], order AP before/after NP.

By (5), the same hierarchical syntactic structure $[_{NP}$ AP NP] can be linearised with either the AP or the NP first. Crosslinguistic variation depends on whether the language chooses one or the other option.

Linearisation statements like (5) can however only account for part of the typologically attested orders. As noted by Abels & Neeleman (2012: 33–35), only 8 out of the 14 ordering possibilities in **Table 1** can be reduced to different linearisation statements applying to one and the same underlying structure. The remaining 6 orders require derivations that also involve syntactic movement, which must be leftward and – as in Cinque's formulation – contain the lexical head.[2] To the extent that no independently motivated syntactic dependency justifies these operations, this amounts to a residue of meaningless movement.

[1] Relevant derivations involving subextraction are discussed in Section 4.3.

[2] Manzini (Submitted), building on Chomsky (2013; 2021), proposes a second type of linearisation operation covering cases of so-called "head movement". However, even with this extension, the system cannot account for orders involving extraction of the head *plus* additional material (e.g. N-A-Dem-Num and A-N-Dem-Num).

Abels and Neeleman's system eliminates the need for the subset of meaningless movements that violate the constraints expressed in the literature on meaningful movement (see Abels & Neeleman 2012: §5.2 for a discussion). However, it does not eliminate the need for meaningless movement itself, and therefore shares two issues with Cinque's system. First, it requires the stipulation of semantically vacuous syntactic dependencies. Second, it implies an asymmetry that has no independent motivation: only meaningless movement is constrained to apply to constituents containing the lexical head, unlike its meaningful counterpart.

The approach we develop addresses these issues by proposing that while all operations deriving basic word-order are syntactic (in line with Cinque 2005 and ff.), their trigger is syntax-external. We build on a Nanosyntactic theory of externalisation (Starke 2009; 2014; 2018; Caha 2009; 2019), and argue that this set of operations apply to syntactic constituents as a result of a failed lexicalisation. When a given syntactic constituent fails to find a matching LI in the language-specific lexicon, a fixed series of syntactic movements is attempted to find one (the Lexicalisation Algorithm, see Section 3). Since such movements encode a version of the restriction in (4), this correctly derives all and only the attested patterns.

As mentioned in Section 1, couching the analysis of word-order variation in this system allows to maintain a universal linearisation procedure (in the spirit of the LCA) without the need for the problematic conflation of meaningful and meaningless movement. The first is associated to a syntactic dependency that has semantic/functional correlates, whereas the latter amounts to a lexicalisation-driven operation. This provides a rationale for the different profiles of the two types of operation. Specifically, the derivation of word-order can be captured without giving up a restrictive theory of meaningful movement, and without postulating vacuous featural dependencies.

Implementing this line of analysis comes with a requirement: if word-order patterns like those in **Table 1** reflect the application of operations defined in the Lexicalisation Algorithm, it must be the case that lexicalisation-driven operations can be triggered by configurations that involve the merger of two phrasal constituents, e.g. an adjectival and a nominal one. As we detail in the next section, this is not the case under the condition on lexicalisation currently assumed in Nanosyntax (see Caha et al. 2024 for an overview). In what follows, we show how an alternative formulation of this condition unlocks the required derivations.

3 The Lexicalisation Algorithm and its reconceptualisation

3.1 Nanosyntax, the core concepts

At the heart of Nanosyntax lie three ontological categories: a universal sequence of functional features (or functional sequence, in line with the Cartographic approach; Rizzi 1997; Cinque 1999; Cinque & Rizzi 2010), the language-specific Lexical Items (LIs) by which the functional sequence

can be lexicalised, and the Lexicalisation Algorithm dictating how the functional sequence and the language-specific LIs interact.

Since the functional sequence (*fseq*) is assumed to be universal, no crosslinguistic variation is encoded within this domain. Concrete syntactic derivations employ features drawn from the universal *fseq*, depending on the functional semantics the speaker wants to express. Each derivation must be however compliant with the universal ordering of the *fseq*, in that the features it combines must be merged to the structure in the order dictated by the hierarchical relations stated in the *fseq*.

The set of language-specific LIs form the language-specific lexicon, the locus of crosslinguistic variation. Each LI is a device that can store externalisation information (whether phonological, as often in verbal languages, or in a different format), syntactic information (a lexically stored syntactic constituent or L-tree), and semantic-conceptual information falling outside the set of functional grammatical features. This is schematized in (6).

(6) LI_n: ⟨ /fəˈnɑlədʒi/, XP, [CONCEPT] ⟩

△

The Lexicalisation Algorithm is a universal set of operations guiding the lexicalisation of any given structure resulting from binary MERGE of syntactic-semantic features drawn from the *fseq*. It is cyclically activated at each application of MERGE. As customary in Nanosyntax, we assume MERGE to be a binary operation that assembles categories drawn from the *fseq* into syntactic trees. Specifically, it combines an already existing syntactic object with a new object that projects the next functional category in the active derivation.[3] The new object can be a feature (MERGE F), which we discuss here, or a phrase, which we address in detail later in this section and in Section 3.2. The goal of the Lexicalisation Algorithm is to provide the new syntactic object with a lexicalisation *via* the language-specific lexicon. The next object is not merged until a lexicalisation is found.

The currently most updated version of the algorithm is in (7) (Caha 2019; Caha et al. 2024).[4]

(7) MERGE F

a. LEXICALISE [FP].

b. LEXICALISATION-DRIVEN MOVEMENT I: If fail, evacuate the closest labelled non-remnant constituent, re-try a.

[3] See fn. 8 for a discussion of the first application of merge.

[4] (7) merges the most recent version of the lexicalisation-driven movements, reconceptualised in terms of (sub)extraction (Cortiula 2023; Caha et al. 2024), with the most updated version containing the creation of a Complex Left Branch (Caha 2019).

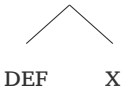
c. LEXICALISATION-DRIVEN MOVEMENT II: If fail, evacuate the immediately dominating constituent, re-try a. (recursive)
d. BACKTRACKING: If fail, go back to the previous cycle and try the next option for that cycle, re-try a. (recursive)
e. COMPLEX LEFT BRANCH: if fail, build a new derivation in an auxiliary workspace providing F and merge it with the main derivation from the original workspace, projecting [FP].

Each step is activated only if the previous one fails to provide a lexicalisation, and starts from the initial syntactic object resulting from MERGE F (i.e. it does not build on the output of the previous steps). Let us illustrate this by deriving a plural NP, assuming the minimal lexicon in (8).

(8) a. LI_{27}: ⟨ /ˈkæt/, NP, [CONCEPT] ⟩

b. LI_{12}: ⟨ /z/, PLP ⟩

c. LI_{10}: ⟨ ðə/, DEFP ⟩



We start from the derivational step in (9).

(9) NP

When (9) is derived, step a. of the Lexicalisation Algorithm is activated, and an LI matching it is searched in the lexicon. The matching operation looks at the syntactic information stored in the LIs in (8) – the lexically stored constituents, or L-trees – and checks if it fulfils the MATCHING CONDITION in (10) with respect to the syntactic phrase subject to lexicalisation (S-tree), here (9).[5]

(10) (Caha et al. 2024: 18)
A lexically stored constituent *L* matches a syntactic phrase *S* iff *S* is identical to *L*.

[5] The MATCHING CONDITION is a simplified version of the SUPERSET PRINCIPLE (Starke 2009: 3) maintaining all its properties *via* a more explicit formulation (Caha et al. 2024: 18–19).

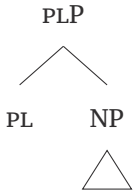
As any subconstituent of a lexically stored constituent is itself lexically stored (Caha et al. 2024), matching is satisfied also when an L-tree contains the S-tree as one of its subconstituents. Concretely, the matching process top-down evaluates the nodes of the S-tree for identity with an L-tree. When many L-trees match an S-tree, the L-tree with fewer unused features wins (an instance of Kiparsky's 1973 Elsewhere Condition, also known in Nanosyntax as MINIMISE JUNK, Starke 2009: 4). In the case of (9), the lexicon in (8) is searched for an L-tree matching NP. Since (8a) contains it, (9) is matched and lexicalised. This is shown in (11), where the circle indicates that the S-tree is paired with the LI reported below.

(11)

LI_{27}

Merging the next functional feature PL yields (12).

(12)

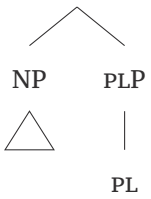


This again activates the Lexicalisation Algorithm, which searches for (12) in the lexicon. Only (8b) has an L-tree that contains the S-tree's root node (PLP), but the L-tree's node does not directly dominate an NP. Hence, (12) is not matched and step b. of the algorithm is activated (LEXICALISATION-DRIVEN MOVEMENT I), targeting the closest labelled non-remnant constituent. Following Caha et al. (2024), a non-remnant constituent is defined as a constituent out of which nothing has moved.[6] By this definition, the closest labelled non-remnant constituent within (12) is NP, which is therefore moved to the left of PLP. As this operation is lexicalisation-driven and does not add any functional semantics, PLP and NP are joined by an unlabelled root node, yielding (13). Since lexicalisation-driven movements are assumed not to leave traces (Caha 2009), PLP is a unary-branching node that only dominates the feature PL.[7]

[6] Naturally, this leaves open the question of how to technically identify a node out of which nothing has moved. Maintaining the idea that lexicalisation-driven movements leave no traces (see fn. 7), one could define a node as remnant if such node is unary or directly dominates a unary node. In Section 3.2, we put forward a different implementation that does not refer to non-remnant constituents. By this novel definition, only syntactic configurations that are the output of previous lexicalisation cycles can be moved.

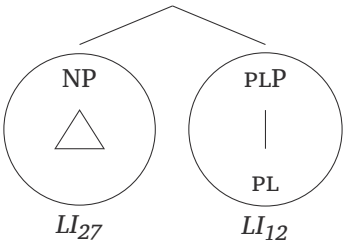
[7] See also Starke (2018) and Caha (2019) for a discussion on why such movements do not leave traces. The crucial point is that such movements are not triggered by a syntactic dependency, they are a response to a lexicalisation problem. As such, there is no chain between two different interpretive positions. This in turn entails that they have no semantic import, do not alter scope relations, and show no reconstruction effects.

(13)



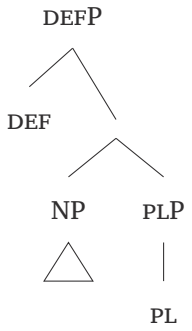
After movement applies, the lexicon is searched again for a match (i.e. step a. of the algorithm is attempted again). This time, the search is successful: the L-tree in (8a) matches the structure under the left node, while the L-tree in (8b) matches the structure under the right node (14). The root node is not matched, but this is not problematic: lexicalisation is successful as soon as all features are contained in a matched node, which is the case in (14).

(14)



As a next step in our example derivation, the merger of the new feature – DEF – yields DEFP (15).

(15)



As at each syntactic cycle, the lexicon in (8) is searched to lexicalise the new S-tree (step a.). Since no match is found for (15), the subsequent steps of the Lexicalisation Algorithm are activated. In this case, however, lexicalisation-driven movements cannot yield a structure where DEF is lexicalised. This is because the only L-tree containing DEF is in (8c), and it has two bottom features, DEF and X (binary "foot").[8] To see why this is problematic, consider the mutual

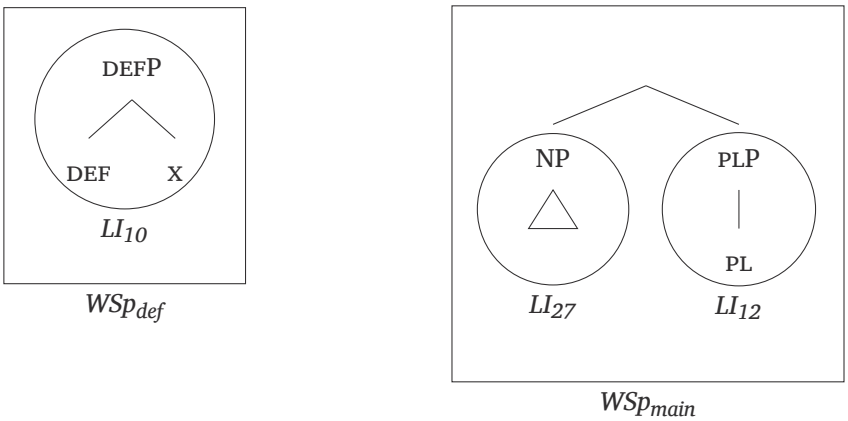
[8] In line with Caha (2019: 157–158), we use X as a placeholder to abstract away from the debate on the first merge operation and what it combines, which is beyond the scope of the paper (on this, see also De Belder & van Craenenbroeck 2015). As for fully functional items (e.g. complementisers, aspectual prefixes, determiners), it has been proposed that their bottom is composed merging two features of the main *fseq* (Starke 2018; Caha 2019), or that first merge always combines the new feature with an inert one, which never projects. On the other hand, items carrying additional conceptual information (e.g. adjectives, adverbs) plausibly represent a different case, as they involve a separate *fseq* (see fn. 10).

implication between lexicalisation-driven movements and unary nodes. Since lexicalisation-driven movements are assumed not to leave traces, they always leave a unary node on their right, (see (13)). On the other hand, unary nodes can only be created by lexicalisation-driven movements, since the only other operation that can alter the structure is internal/external merge, which is binary. This entails that only L-trees with a single bottom feature (unary "foot"; e.g. (8b)) can match a structure resulting from the application of lexicalisation-driven movements after MERGE F, unlike LIs that only contain binary nodes, like (8c). As a consequence, each step of the algorithm necessarily fails to lexicalise DEF, until step e.[9]

Step e. (COMPLEX LEFT BRANCH) implicates two operations, (i) building a new "auxiliary" XP containing the relevant feature (here, DEF), (ii) merging the XP with the main derivation, projecting DEFP. Operation (i) requires a parallel workspace, additional to the one of the main derivation, where the new "auxiliary" XP is derived. We take each workspace to minimally contain the syntactic structure that has been derived and lexicalised in previous cycles (if present), as well as the LIs lexicalising it. This structural information is only updated once a successful lexicalisation is found for the S-tree derived within the workspace.

With this in place, let us go back to our example derivation. Prior to operation (ii) we have two distinct lexicalised trees in two separate workspaces, encased in squares in (16). One is the output of the previous lexicalisation cycle in the main workspace, call it WSp_{main} (cf. (14)). The other is the DEFP built in the auxiliary workspace, call it WSp_{def}.

(16)

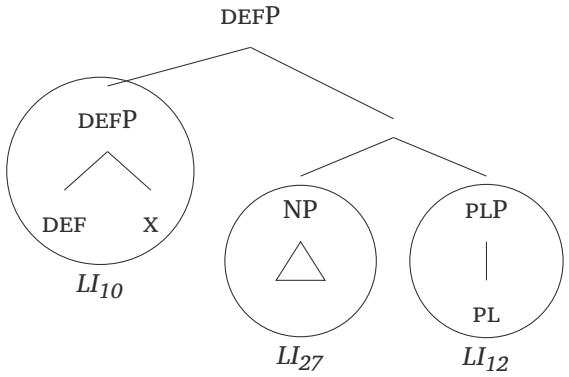


Merging the two S-trees from the workspaces in (16) yields (17).[10]

[9] This includes BACKTRACKING to previous cycles of the derivation (step (7)d.), which we do not discuss in this contribution. For an overview on its properties, issues, and alternatives see Blix (2021).

[10] In its current formulation, step e. of the algorithm models covers "auxiliary" XPs of (descriptively) two types: fully functional ones like the definite article in (17), whose insertion is only driven by the functional features that are in need of a lexicalisation in the main spine, and XPs carrying additional conceptual information, as in the case of e.g. adjectives. This raises the question how this distinction should be captured (on this, see also Starke 2024b). As further

(17)



The root node of (17) is not matched, parallel to (14). This is again not problematic, as all features are nonetheless contained in a matched node. Auxiliary XPs are also known in the Nanosyntactic literature as Complex Left Branches (CLBs), indicating a left constituent terminating with a binary "foot". In the system described so far, the complexity of the "foot" of the L-tree predicts the position of the LI with respect to the lexical core of the derivation: LIs whose L-tree has a unary "foot" are necessarily to the right ("post-elements"), LIs whose L-tree has a binary "foot" – auxiliary XPs/CLBs – are necessarily to the left ("pre-elements"). This has been proposed in Starke (2018) and adopted in Caha (2019); Pinzin (2024); Gök & Demirok (In press), among others. To the best of our knowledge, this represents the state of the art of the Nanosyntactic computation.

3.2 A new lexicalisation condition and its consequences

This brings us to the focal point of our discussion. In the current formalisation, CLBs are the last step of the Lexicalisation Algorithm (cf. (7)), and are always successfully merged to the left of the lexical core of the derivation. Since both *black* and *nero* are complex modifiers and hence merged as CLBs (both are not strictly adjacent to the N and can be modified, e.g. *very black/molto nero*), it is impossible to model word-order alternations as in (18) (repeated from (1)) in terms of lexicalisation-driven movements. Such movements can only apply (and fail) before deriving and merging a CLB like *black/nero*. After merger of the adjectival CLB on top of the noun, the algorithm defines no further operation. As such, the alternation in (18) must rather be modelled in terms of either a featural dependency driving movement of the noun across the adjective in (18b) or linearisation statements encoding their relative order.

(18) a. black dog
b. *Italian*
cane nero
dog black
'black dog'

clarified later, this is not directly relevant to our argument: to the extent that both scenarios involve merger of an XP to the main structure, our proposal uniformly applies to both.

In other words, any attempt to capture word-order alternations in the current system must be in line with the models already available in the literature and inherit their tensions (see Section 2). As already claimed in Section 1 and 2, we contend that a single revision allows an alternative way to model word-order patterns in terms of lexicalisation-driven movements. Specifically, the key to unlock this line of analysis amounts to a modification of the lexicalisation condition. Current Nanosyntax considers a syntactic node to be lexicalised as soon as all its features are contained in a node matched by an LI. This is expressed in (19).

(19) A syntactic node is lexicalised iff all the features contained in it are dominated by a node matched by an LI.

Following (19), the root nodes in (14) and (17) are lexicalised. We propose to replace (19) with (20).

(20) A syntactic node is lexicalised iff all the labelled nodes contained in it are matched by an LI.

Compared to (19), (20) shifts the target of lexicalisation from features to labelled nodes. The intuition behind this is the following: the addition of new functional material to the structure results in the projection of a new label on the main spine, and each such new object must be licensed by being matched by an LI.

Note that the two formulations are equivalent in each configuration derivable via MERGE F only. This is because matching only targets phrasal constituents, and not single features (see the MATCHING CONDITION in (10)). This means that the lexicalisation of each feature implies matching of its labelled mother node, or of a constituent containing it. Since new labelled nodes in these derivations are only created by merging in a new individual feature, a labelled node cannot remain unmatched (violating (20)) without also leaving a feature without lexicalisation (violating (19)).[11]

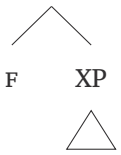
However, the equivalence does not hold when we consider the operation of adding F to the S-tree of the main workspace *via* a CLB derived in a parallel workspace, as in (17). In such cases, (20) requires both the labelled root node of the CLB and the labelled root node of the full structure to be matched by an LI, while (19) only requires the node dominating F to be matched by an LI, i.e. the root node of the CLB. Concretely, (17) is fully lexicalised under (19), but not under (20). By the latter, the unmatched labelled root-node DEFP blocks any further merge operation.

From a different perspective, adopting (20) results in parallel lexicalisation requirements for both the case in which a new feature is added to the S-tree of the main workspace directly

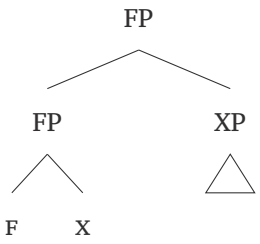
[11] From a theory-internal perspective, this means that adopting (20) has no bearing on previous Nanosyntactic analyses adopting just the MERGE F operation.

(MERGE F) and as a part of an XP derived in a parallel workspace. Both in (21a) and (21b), the root node of the structure must be matched by an LI.

(21) a.



b.



This parallelism further implies a reconceptualisation of the Lexicalisation Algorithm, by which the operation that adds a new branch to the structure, call it MERGE FP, is parallel to MERGE F. In this new version, spelled out in (22), MERGE FP substitutes the CLB step in (7). Like the CLB step, it only applies if the whole set of lexicalisation-driven movements after MERGE F apply and fail. Unlike in the previous version, however, MERGE FP is followed by the same lexicalisation steps that follow MERGE F: lexicalisation of the root FP node (a.), the two types of lexicalisation-driven movement (b. and c.), backtracking (d.).[12]

(22) Lexicalisation Algorithm:

1. MERGE F	a. LEXICALISE [FP].
2. MERGE FP	b. LEX-DRIVEN MOV I: If fail, evacuate the closest labelled non-remnant constituent, re-try a.
	c. LEX-DRIVEN MOV II: If fail, evacuate the immediately dominating constituent, re-try a. (recursive)
	d. BACKTRACKING: If fail, go back to the previous cycle and try the next option for that cycle, re-try a. (recursive)

Note that our reasoning applies to any configuration in which a new phrase is added to the structure and projects a feature, regardless of the mechanism deriving this phrase (see fn. 10). In what follows, we abstract away from this issue.

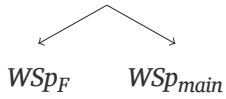
[12] A similar reconceptualisation of the Lexicalisation Algorithm is proposed in Starke (2024a), where the relative order of MERGE FP and movement based on syntactic dependencies is considered too (MOVE/RE-MERGE FP). As this is not relevant for our proposal, we leave this aspect aside.

What is crucial is that following this reconceptualisation, lexicalisation-driven movements can be triggered after MERGE FP, allowing the LIs present in the language-specific lexicon to influence the position of a new branch with respect to the rest of the derivation via lexicalisation-driven movements.

Let us take a further step. In a structure resulting from MERGE FP as (21b), which kind of LIs match the root FP node? In line with Nanosyntax, we assume that any new externalisation/conceptual information carried by the LI matching a node *N* necessarily overwrites any previous information provided by the LIs matching the nodes contained within *N*. If the LI does not contain externalisation/conceptual information, we take the previous information to be inherited. If the LI matching the root FP node in (21b) were to carry externalisation/conceptual information, such information would overwrite the information already present in the two branches dominated by FP, i.e. the main branch requiring F (XP) and the branch providing F (the lower instance of FP). Since on the contrary MERGE FP *adds* a new FP branch to the main derivation and does not overwrite (in standard cases) any previous externalisation/conceptual information, we conclude that the relevant LIs matching the root FP node lack any externalisation/conceptual information and only contain syntactic information.[13] In a sense, these LIs are "ordering instructions" storing – and therefore licensing – a syntactic arrangement of the two branches involved in a MERGE FP operation. As these LIs order the lexicalisation output of previous derivational steps, we refer to them as second-order LIs.

In concrete terms, a second-order LI storing and licensing the order FP – main branch can be schematised as in (23).

(23) $LI_{FP\text{-}main}$: ⟨ FP ⟩



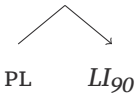
No externalisation or conceptual information is present, as discussed above. The syntactic information requires instead further clarifications. Both branches of the L-tree in (23) are *pointers*, as signalled by their arrow-shaped edge. Pointers are standardly used in Nanosyntax to allow reference to specific LIs within an L-tree (Starke 2014; Caha et al. 2019; Caha 2019). An L-tree containing a pointer to an LI can only match a structure in which the node that is pointed to is lexicalised by the specified LI (see Caha 2019: §4.6). This models cases where the insertion of a given LI is contingent on the presence of other LIs in the tree, as with root suppletion (where the suppletive form can only target structures containing a given root LI, cf. Caha et al. 2019) or

[13] Starke (2014) already proposes LIs only containing syntactic and conceptual information but without externalisation to capture idiomatic conceptual readings of complex phrases, as *kick the bucket* (= TO DIE), or *hold your horses* (= BE PATIENT). See Baunaz & Lander (2018) for an overview. Our LIs are therefore a natural extension of the system.

idioms (where idiomatic conceptual content is inserted only in structures containing the relevant LIs, cf. Starke 2014). To exemplify, we can take the singular-plural couple *mouse-mice*, where the LI for *mice* (25) points to the LI for *mouse* (24) in its L-tree.

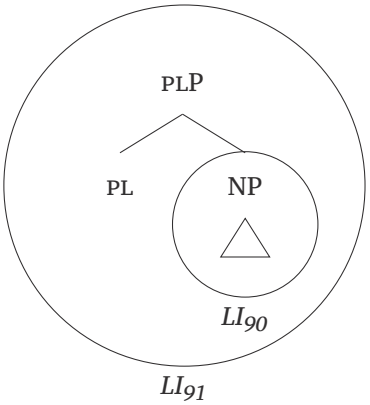
(24) LI_{90}: ⟨ /maʊs/, NP, [CONTENT] ⟩

(25) LI_{91}: ⟨ /maɪs/, PLP ⟩



The L-tree in (25) matches a PLP node only if its right node is lexicalised by (24), as in (26).

(26)



L-trees containing a pointer to an LI abstract over the different syntactic configurations that can be matched by that LI. To see why, let us consider how matching works in such cases. With respect to the MATCHING CONDITION in (10), pointers act as "breakpoints" in the matching procedure. Namely, once the top-down comparison process between an S-tree and an L-tree finds a pointer to some node *N*, a secondary comparison procedure starts.[14] This secondary comparison evaluates whether node *N* in the S-tree has a successful lexicalisation based on the information stored in the corresponding node *N* that is pointed to in the L-tree. For example, in (26) it evaluates whether the node NP in the S-tree has a successful lexicalisation based on the information in the corresponding node that is pointed to in the L-tree, i.e. LI_{90}. As the syntactic information in LI_{90} – its L-tree – contains an identical NP, a match is found. Note that LI_{91} in (25) is devoid of conceptual content. As previously remarked, this entails that the already available conceptual information is not overwritten, but carried over from the previous cycle.[15]

[14] We thank Pavel Caha for a crucial discussion on this point.

[15] We thank an anonymous reviewer for pointing out that the "carrying-over" mechanism we assume can be extended to the analysis of root suppletion.

From a more general perspective, the node that is pointed to is a variable over possible syntactic trees. In the case of pointers to an LI, it ranges over any tree contained in its L-tree. If the S-tree is identical to one of the possible values of this variable, a match is found. We propose to extend recourse to this device by allowing reference to other entities visible to the derivation, i.e. not only features and LIs, but also workspaces.

The underlying reasoning is the following: the two branches involved in a MERGE FP operation are built in two independent derivations. Each derivation runs in a separate workspace, as defined in Section 3.1. From the global perspective of the derivation building e.g. a nominal phrase modified by an adjective, syntax has access to two workspaces, one containing the derivation of the adjectival phrase, one containing the derivation of the nominal phrase below the merge site of the adjective. The two workspaces involved in a MERGE FP operation are identified by their role: the main workspace is the one with which the initial feature has been merged (*via* MERGE F), while the other one provides that feature. We refer to the former as WSp_{main} and to the latter as WSp_F. We take workspaces to be objects visible to syntax, so that L-trees can point to them as much as they can point to LIs.

With this in place, let us look again at (23). Its left branch points to the workspace in which the new branch providing F has been built and lexicalised (WSp_F), while its right branch points to the workspace of the main derivation (WSp_{main}). The MATCHING CONDITION works as specified above for the case of pointers to LIs, and checks if the node in the S-tree has a successful lexicalisation based on the information contained in the pointed node of the L-tree, i.e. the syntactic information present in WSp_F for the left node and in WSp_{main} for the right node. Thus, (23) matches an FP whose left node matches the content of WSp_F (i.e., the last tree lexicalised in WSp_F) and whose right node matches the content of WSp_{main} (i.e., the last tree lexicalised in WSp_{main}). Since the content of each workspace necessarily varies depending on the specific derivation, a pointer to a workspace abstracts over the infinite number of possible derivations that can be performed in it. This level of abstraction is necessary: MERGE FP combines recursive structures that cannot be referred to via a static index.[16] If (23) is stored in the lexicon of a language, the order in which the new FP branch precedes the rest of the derivation is licensed in that language, irrespective of their content.

To exemplify the role of LIs like (23) in a derivation, we show how this system models the variation in (18). Following the Cartographic literature, we assume that an adjective like *black* satisfies a functional feature within the nominal functional spine (Cinque 2010, a.o.). We label

[16] Other options for the left branch are possible, as referring to single LIs of lists of them; we discuss them in Section 5. Note that referring to workspaces appears to be incompatible with the presence of segmental externalisation information. If (23) were to contain such information, it would necessarily overwrite the externalisation information from any previous node. Each structure labelled as FP would therefore be externalised identically, severing any link between form and conceptual information.

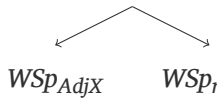
this feature ADJ_X and abstract away from the issue of its exact identification (see Svenonius 2008 and Cinque 2010 for an overview and Manzini Submitted for a recent critical discussion). In this context, we only need this feature to exist, irrespective of its precise definition.

We take the English word-order in (18a) to follow from the lexicon in (27).

(27) a. LI_{45}: ⟨ /ˈblæk/, ADJ_XP, [CONCEPT] ⟩

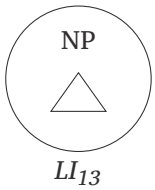
b. LI_{13}: ⟨ /ˈdɔg/, NP, [CONCEPT] ⟩

c. LI_{58}: ⟨ ADJ_XP ⟩



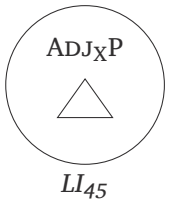
Let us start from the derivational step where the NP has already been derived and lexicalised via (27b), following the Nanosyntactic mechanisms seen above. This yields (28).

(28)



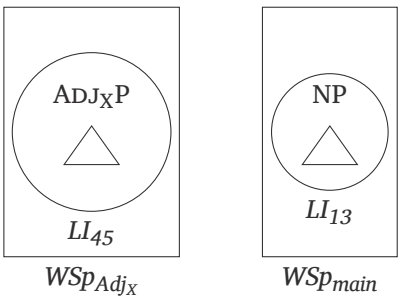
Now, the functional layer related to ADJ_X is added. The algorithm in (22) dictates that MERGE ADJ_X (= MERGE F) is attempted first, and that the merger of an independent branch projecting ADJ_X (MERGE ADJ_XP) is only attempted if all the steps of the algorithm following MERGE ADJ_X fail. Since the lexicon in (27) contains no L-tree with a unary bottom, this is necessarily the case. Now, the MERGE FP operation implies that a new branch providing ADJ_X – whose lexicalisation is LI_{45} (*black)* – has been independently derived and lexicalised in a separate workspace, yielding (29).

(29)



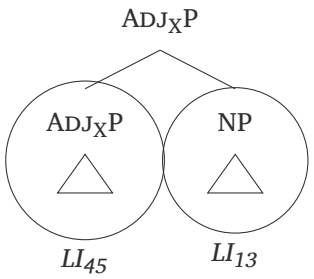
At this point of the derivation, two workspaces are active: WSp_{Adj_X} and WSp_{main}. This is represented in (30).

(30)



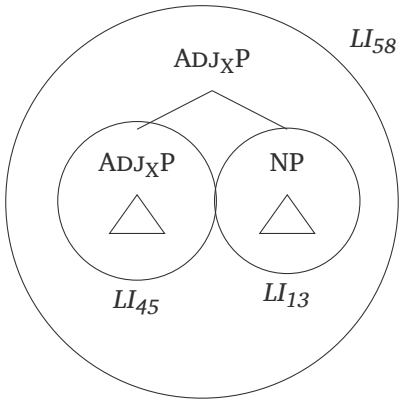
The two S-trees in (28) and (29) are now merged, projecting ADJXP.

(31)



This is the crucial point where our proposal differentiates itself from previous Nanosyntactic accounts. (31) contains a labelled node that is not matched by any LI: the root node ADJXP. Adopting the new lexicalisation condition in (20), (31) is not lexicalised. Hence, the Lexicalisation Algorithm is activated and looks for a match for the root node ADJXP. Given the presence of (27c) in the lexicon above, the search is successful. The root node of the S-tree in (31) is contained in the L-tree in (27c). Proceeding top-down, the comparison procedure finds pointers in the L-tree for both branches, triggering two secondary comparison procedures. For each of the two branches, such procedure checks if the content of that branch of the S-tree is identical to a tree contained in the workspace pointed to in the corresponding node, i.e. the current instance of WSp_{Adj_X} and WSp_{main} shown in (30). As this is the case, match is successful, yielding (32).

(32)



This derives the English order, where *black* precedes *dog*.

Let us now look at Italian, where the opposite is true (18b). As customary in Nanosyntax, crosslinguistic variation reflects a difference in the lexicon of the two languages. More precisely, we derive the opposite order from the lexicon in (33).

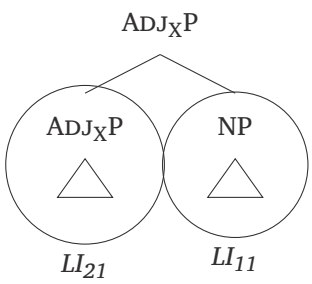
(33) a. LI_{21}: ⟨ /ˈnero/, ADJ_XP, [CONCEPT] ⟩

b. LI_{11}: ⟨ /ˈkane/, NP, [CONCEPT] ⟩

The difference between (33) and (27) is the absence in the Italian lexicon of the LI licensing the linearisation of adjectives before the rest of the structure. Let us see how this derives (18b).

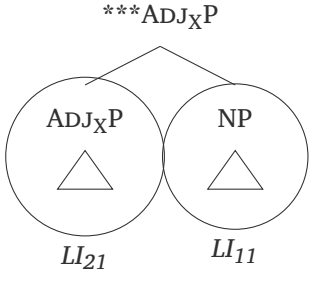
All steps are identical until ADJ_XP is merged with the main derivation, yielding (34) (cf. (31)).

(34)



By (20), (34) is not lexicalised, like (31). This activates the Lexicalisation Algorithm, which attempts to match (34). However, no LI in (33) matches it, contrary to the English derivation. The root node of the S-tree in (34) is only matched by the L-tree in (33a). However, the L-tree in (33a) only contains the adjectival branch, and not the NP. As no other LI is available, matching fails, as signalled in (35) by the three stars.

(35)



The movement steps of the Lexicalisation Algorithm are then activated, targeting the closest labelled non-remnant constituent (step b.). Following the definition of non-remnant as *a constituent out of which nothing has moved* (cf. Section 3.1), both ADJ_XP and NP are possible

candidates. Now, movement of ADJ_XP would be in violation of the empirically necessary restriction identified by Cinque (cf. Section 2), namely that only constituents containing the lexical head can move. Under this formulation, something else must exclude movement of modifiers alone. This follows if *closest* is defined as the non-reflexive relation *closest to the item whose merger has triggered the opening of the lexicalisation procedure* (in this case, ADJ_XP). The same result is obtained by assuming that lexicalisation-driven movements cannot disrupt labelling dependencies: movement of ADJ_XP would leave a unary phrasal node with the same label dominating only NP.

Note that the issue stems from the negative definition of the licit targets of lexicalisation-driven movements provided in the Lexicalisation Algorithm. Following the positive definition in Cinque (2005; 2023), whereby only constituents containing the head of the relevant (sub)hierarchy can move, no additional specification is needed. In (35), only the NP contains the head of the relevant (sub)hierarchy, the nominal one. In Nanosyntactic terms, this can be rephrased as constraint whereby only constituents that contain the lowest feature of the current *fseq* can move. Contrary to the negative definition we adopted until now, this excludes movement of any constituent merged *via* MERGE FP.

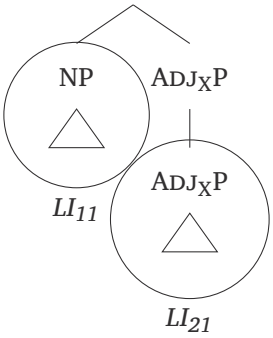
While this formulation correctly captures the required constraints, it remains stipulative. Reasoning in terms of workspaces, however, enables an alternative perspective on the issue of why only XPs containing the lowest feature of the current *fseq* are proper targets of lexicalisation-driven movements. As discussed in Section 3.1, we take a workspace to be the object that (minimally) stores the output of a complete derivational cycle, i.e. the S-tree built via the application of a MERGE operation together with information about how it is lexicalised. Following the logic of the Lexicalisation Algorithm, a new derivational cycle can only start if the previous one is complete, i.e. if the S-tree built up to that point has found a successful lexicalisation. The algorithmic nature of the lexicalisation procedure is such that different syntactic operations may be attempted before reaching the one that enables a successful lexicalisation. Once this is reached, the corresponding S-tree and the information about how it is lexicalised must be stored for them to be the input of further derivational cycles or an externalisation procedure. After each complete derivational cycle, the content of the current workspace must therefore be updated, where *current* refers to the workspace where the last MERGE operation has taken place. Consider now that each update of a workspace necessarily contains the whole S-tree derived up to that point in that workspace, up to its lowest feature. Following this logic, step b. of the algorithm could be rephrased as *evacuate the closest labelled constituent that is the output of a previous derivational cycle in the current workspace*, as in (36). Intuitively, you can only move trees that have already been successfully lexicalised in the current workspace, starting from the closest labelled one.

(36) Lexicalisation Algorithm:

1. MERGE F	a. LEXICALISE [FP].
2. MERGE FP	b. LEX-DRIVEN MOV I: If fail, evacuate the closest labelled constituent that is the output of a previous derivational cycle in the current workspace, re-try a.
	c. LEX-DRIVEN MOV II: If fail, evacuate the immediately dominating constituent, re-try a. (recursive)
	d. BACKTRACKING: If fail, go back to the previous cycle and try the next option for that cycle, re-try a. (recursive)

Under this formulation, the only proper target of movement in (35) is NP, which is then displaced to the left of ADJ_XP and merged with it *via* an unlabelled root node (37).

(37)



The unlabelled root node does not signal addition of new functional material and is therefore exempt from lexicalisation requirements. The NP on the left already has a lexicalisation: LI_{11}. The same is true for the lower instance of ADJ_XP, which is already lexicalised by LI_{21}. The only labelled node without a match is the higher ADJ_XP node, whose label has been projected by the lower ADJ_XP *via* MERGE FP. Since Nanosyntax standardly assumes lexicalisation-driven movements to leave no traces (see Section 3.1 and fn. 7), the higher occurrence of ADJ_XP in (37) is a unary-branching node only dominating a lower instance of itself, i.e. a case of "vacuous" recursion. Since by (20) all labelled constituents must be matched, this leaves us with two possible options. The first is to postulate the existence of an LI matching the larger ADJ_XP constituent under the right node of the tree. This would establish a symmetry between direct lexicalisation (the case of English; see (32)) and lexicalisation after movement of the whole complement, since both cases would require dedicated LIs to match the resulting structure. While this remains a viable option, we opt for a different implementation, and assume that in such cases of "vacuous" recursion the higher ADJ_XP inherits the match of the lower occurrence of the same label. This results in (37) being licensed as it is.[17]

[17] Note that match by "inheritance" after roll-up movement of the whole complement cannot apply in the context of word-level lexicalisation (MERGE F). The reason is that such process requires the presence of an already matched

A first consequence of this choice is that word-orders derived via systematic roll-up need no second-order LIs to license any of the relevant syntactic outputs. In Section 6, we elaborate on how this choice makes further testable predictions. A second consequence is that, if roll-up after MERGE FP produces a configuration that is licensed in the absence of any dedicated second-order LIs, it will always produce a licit output. This further entails that the BACKTRACKING step of the Lexicalisation Algorithm cannot be activated after MERGE FP, making it irrelevant for the present contribution.

In this section, we have shown that syntactic movements deriving basic word-order can be formalised as a reaction to a lexicalisation problem. These movements are triggered when a syntactically derived constituent is not lexicalised, i.e. when it contains a labelled node that is not matched by any LI of the language-specific lexicon. As these syntactic operations are triggered by an interface condition with the lexicon and not by syntactic dependencies (differently from Cinque 2005; 2023), they are necessarily meaningless. This in turn explains why they follow a different set of constraints than meaningful syntactic operations, which are instead triggered by syntactic dependencies (Abels & Neeleman 2009; 2012). Following this proposal, a universal linearisation procedure can be maintained (in line with Kayne (1994)'s *Linear Correspondence Axiom* adopted in Cinque 2005; 2023), and crosslinguistic variation in basic word-order as (18) can be modelled in terms of an independently necessary source of variation: the content of the language-specific lexicon. These crucial theoretical advantages are however only valid once we demonstrate that the current proposal can model the empirical constraints on crosslinguistic variation in basic word-order highlighted in the literature (see Section 1, **Table 1**). In the next section, we see how this is the case.

4 Deriving U20 generalisations

In the previous sections, we argued that differences in the base word-order can be reduced to variation in the shape and content of the language-specific LIs, which interact with the syntactic derivation as specified by the Lexicalisation Algorithm. Based on the foregoing discussion, two options are open: i) direct lexicalisation (step a. of the algorithm), or ii) movement (step b./c.).

XP dominating the relevant feature, i.e. it necessitates MERGE FP. To see why, consider the starting configuration of MERGE F: [FP F [XP]]. In such cases, roll-up movement yields [[XP] [FP F]]. Match by "inheritance" requires a unary node directly dominating an already matched constituent with the same label. However, this is impossible in this configuration, since matching only targets phrases and not atomic features. This necessarily triggers a search in the lexicon, which can fail and activate the next step in the Lexicalisation Algorithm. MERGE FP starts instead from [FP [FP F] [XP]], where the left [FP F] is already necessarily matched by an LI. Here, roll-up movement yields [[XP] [FP [FP F]], which is the configuration we defined for the application of match by "inheritance": a unary branching node dominating a lower instance of the same node, where the lower instance is already matched by an LI.

The aim of this section is to show that this set of expressive possibilities rules in all and only the operations that capture the absolute constraints in **Table 1**.[18]

The upshot is that this lexicalisation-based model allows to replicate Cinque's (2005) results, deriving all attested orders and none of the unattested ones. From the perspective of the syntactic operations involved, this is trivial, since the system effectively mimics the different derivational options proposed by Cinque. This is already shown in Section 3, where we illustrate how lexicalisation-driven movements only target constituents containing the lexical head (the lowest feature of the *fseq* in Nanosyntactic terms). This is captured by constraining these movements to target trees that have already been lexicalised in the current workspace. The crucial difference is that LIs of the format introduced in Section 3.2 and the standard Lexicalisation Algorithm provide a rationale for the relevant operations, which are connected to an independent theory of externalisation.

For the sake of readability, we do not individually discuss all 24 cases (i.e. the 14 attested ones and 10 unattested ones, cf. **Table 1**). In Appendix 1, we explicitly spell out the list of LIs that need to be postulated for each of the 14 attested ordering possibilities. Here, we first illustrate direct lexicalisation based on the derivation of the linear order Dem-Num-Adj-N. Then, we discuss different movement options, clarifying how the content of the lexicon uniquely determines which option is successful. Finally, we show that unattested orders fall outside of the generative possibilities of the system, since they would require the application of operations that cannot be formulated within the present set of assumptions.

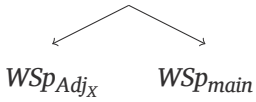
4.1 Dem-Num-Adj-N

We start from the derivation of the order that reflects the order of merge of the four relevant items, namely Dem-Num-Adj-N. In this and the following derivations, we abstract over preliminary steps independently needed in all cases. What is relevant for the discussion is what happens after a new branch is merged to the structure, e.g. when NumP is merged to a tree already containing an AdjP and an NP. In light of the Lexicalisation Algorithm discussed in the previous section, this presupposes that a) all steps in the algorithm preceding Merge FP failed, and b) some LI can lexicalise the new branch in the separate workspace in which it was built. Ultimately, any difference in the output order reduces to which second-order LIs (if any) the lexicon contains in addition to those that lexicalise each branch.

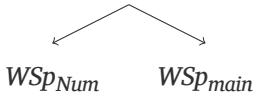
In the case of the (many) languages that display Dem-Num-Adj-N, we postulate the lexicon for such a language to also include the following second-order LIs:

[18] As a consequence, we do not discuss other possible sequences that some languages allow as alternative orders alongside one of the fourteen expected ones (cf. Cinque 2023 for a discussion of relevant cases). In line with Cinque's analysis, we assume the unexpected orders to result from the application of meaningful (i.e. not lexicalisation-driven) movement operations, which are independent of our proposal.

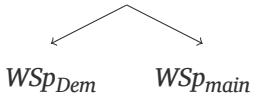
(38) a. LI_{71}: ⟨ ADJ$_X$P ⟩



b. LI_{72}: ⟨ NUMP ⟩

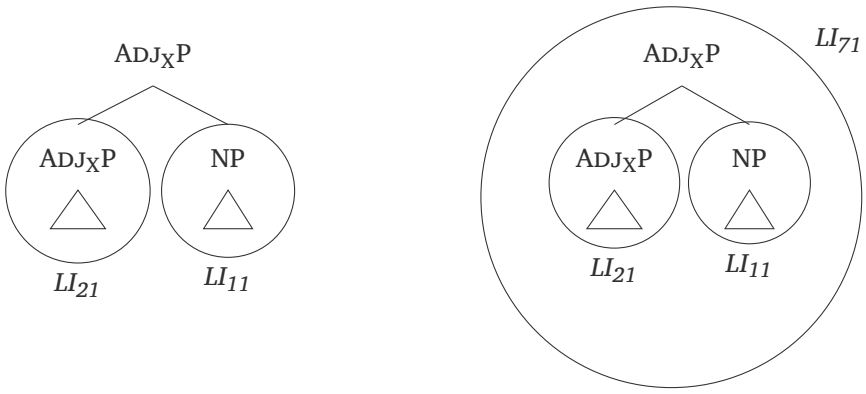


c. LI_{73}: ⟨ DEMP ⟩



Let us see how the derivation proceeds given the LIs above. We start from a situation in which an NP has been derived and the structure is enriched by the merger of an ADJ$_X$P, resulting in the configuration represented on the left in (39). As noted above, this presupposes an intermediate stage (not represented here) in which the adjectival branch is independently built and lexicalised, say *via* LI_{21}. What is crucial is that after the application of MERGE FP a lexicalisation procedure is activated. Step a. of the algorithm looks for a direct match for the entire structure as is. This is found in the case at hand: LI_{71} in (38a) matches precisely a structure in which a projecting adjectival ADJ$_X$P takes the rest of the structure as a complement. Thus, the entire tree is licensed, yielding the order Adj-N.[19]

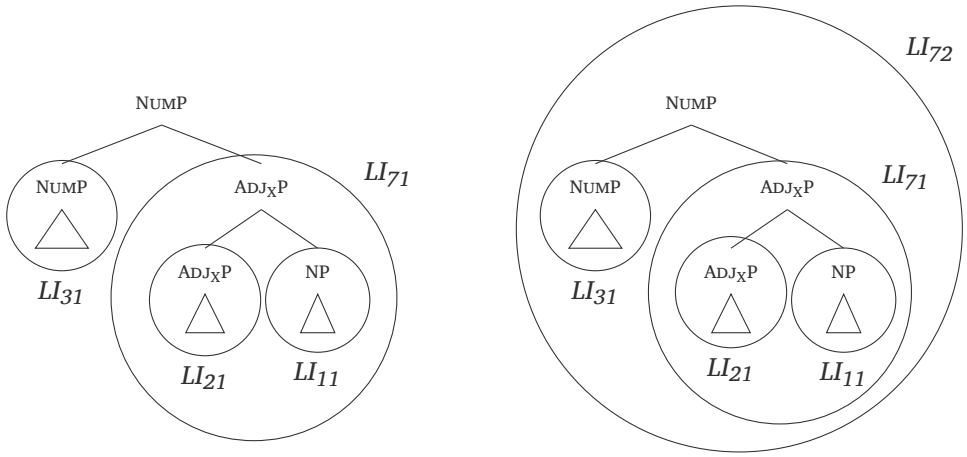
(39)



Later stages of the derivation involve exactly the same steps. Thus, merger of NUMP produces the configuration on the left in (40), which is directly matched by LI_{72} (38b).

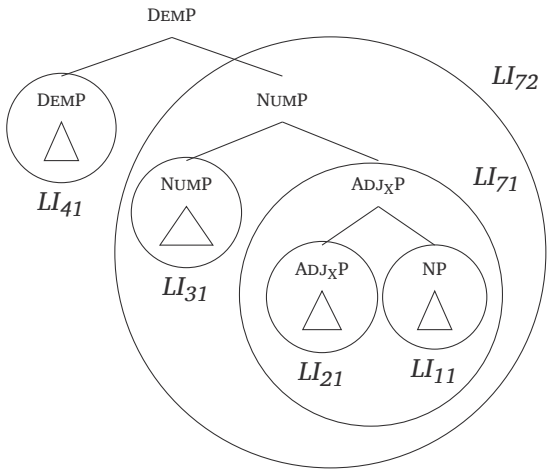
[19] Recall that the mechanism of pointers and the lack of externalisation information in the second-order LI have the effect that the content of the LIs that individually lexicalise the adjective and the noun is not overridden, but rather carried over (see Section 3.2).

(40)

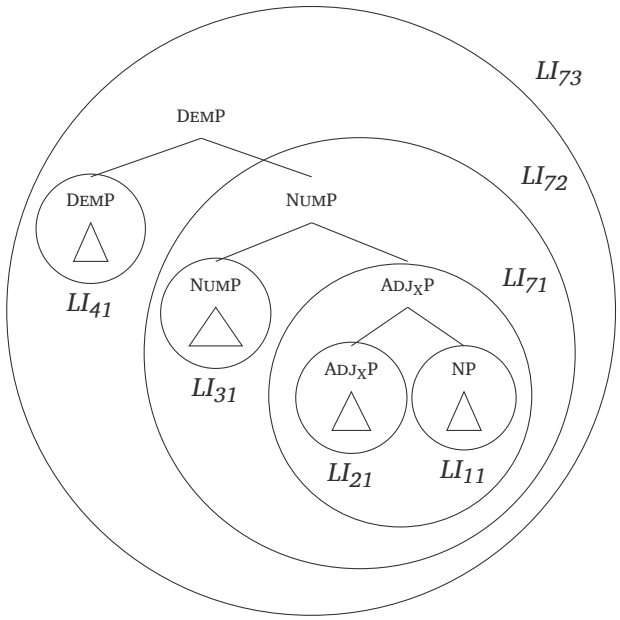


Finally, merger of DemP with the previously derived NumP produces the configuration on the left in (41), which LI_{73} (38c) matches directly (42). This licenses the order Dem-Num-Adj-N without the need for lexicalisation-driven movements rearranging the structure.

(41)



(42)



4.2 N-Adj-Num-Dem

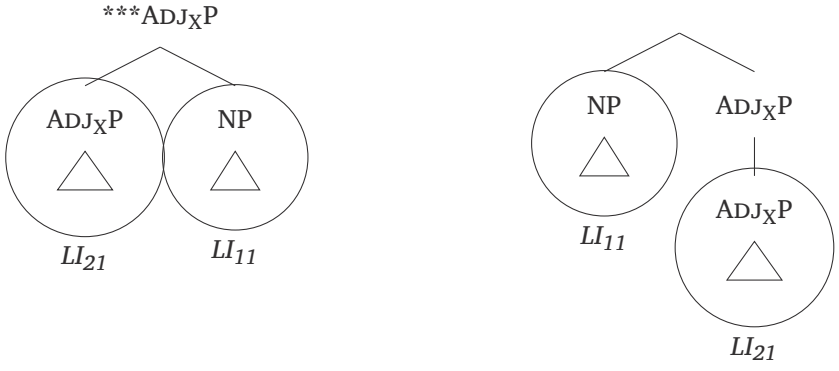
Let us turn to the exact mirror order, namely N-Adj-Num-Dem. As discussed in Section 3.2, the logic of the proposal has the following consequence: after merger of a new branch, application of roll-up derives a configuration in which the lexicalisation requirements are satisfied without the need for any second-order LI.

To repeat the general reasoning, lexicalisation is triggered whenever an application of merge (MERGE F or MERGE FP) creates a new labelled constituent, with the aim to find an LI matching this object. Lexicalisation-driven movements, on the other hand, do not contribute anything new, and merely rearrange existing branches. This property can be captured by postulating that they create unlabelled nodes, as is standard in Nanosyntax. It follows that any node created by a lexicalisation-driven movement will not need to find a match in the lexicon.

In the present context, the consequence is that a language that lacks any second-order LIs dedicated for the lexicalisation of DEMP, NUMP, or (any kind of) ADJP will eventually resort to roll-up to derive a licit configuration whenever any such branch is merged. Consider again the same starting point as for the previous derivation, namely the configuration resulting from the merger of ADJ_XP to NP, represented on the left in (43). Since the lexicon does not contain a matching LI, lexicalisation-driven movements are triggered. By step b. of the algorithm, the first (and only) available operation is movement of NP across ADJ_XP, as shown on the right.

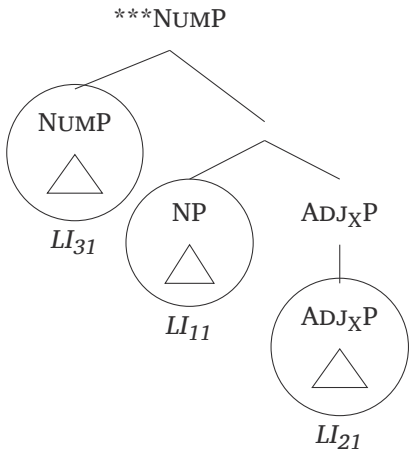
Roll-up of NP creates a configuration in which the right branch of the tree involves what we termed "vacuous" recursion, i.e. a unary-branching node dominating a node with the same label. In Section 3.2, we proposed that in such cases the higher instance of the node inherits the lexical match of its daughter. This has the effect that the order N-Adj is licensed without a dedicated second-order LI, as shown on the right in (43).

(43)



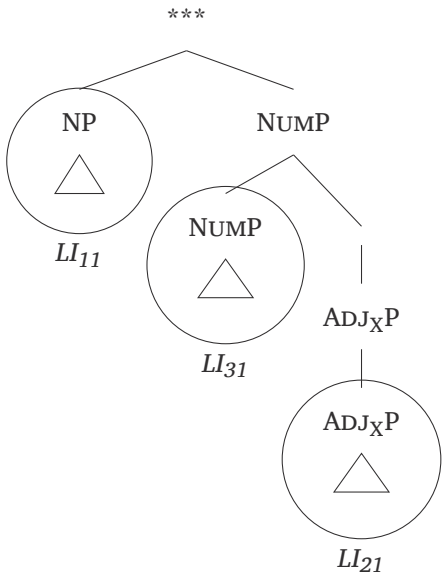
The next step in the derivation involves merger of NUMP. As in the previous cycle, no direct match is found for this configuration (44).

(44)



This triggers lexicalisation-driven movements, which (again by step b. of the algorithm) first attempt movement of the NP alone. This movement leaves a NUMP that has to be matched under the left node. As by hypothesis no second-order LI is present in the lexicon of this language, no match is found (45).[20]

(45)

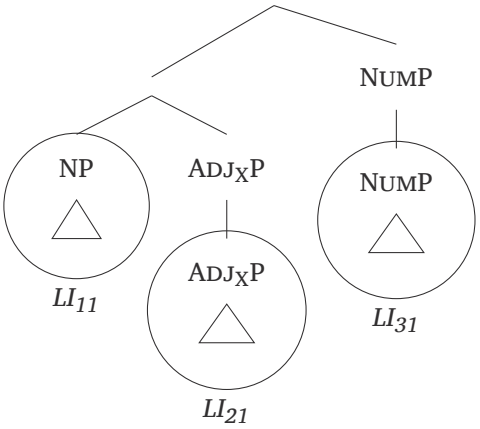


Now, let us recall that the steps of the Lexicalisation Algorithm do not build on top of each other but always start from the initial configuration that triggered the lexicalisation procedure. In this context, this means that step c. of the Lexicalisation Algorithm takes as an input the original structure derived *via* MERGE FP (44) and tries to move the node immediately dominating the one

[20] Analogous cases of movement and the corresponding LIs that license the configurations they produce are discussed in the next section.

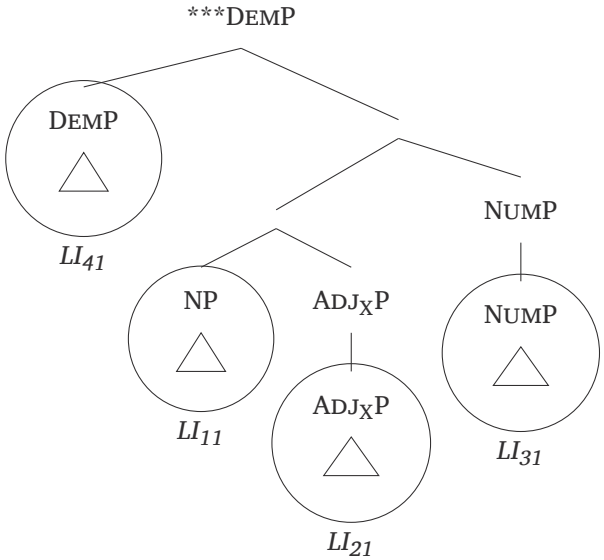
whose movement was attempted at the previous step, i.e. the node immediately dominating NP. The result is shown in (46). In this derived configuration, the reasoning illustrated above about lexicalisation in case of "vacuous" recursive nodes can be replicated. The order N-Adj-Num is thus licensed.

(46)

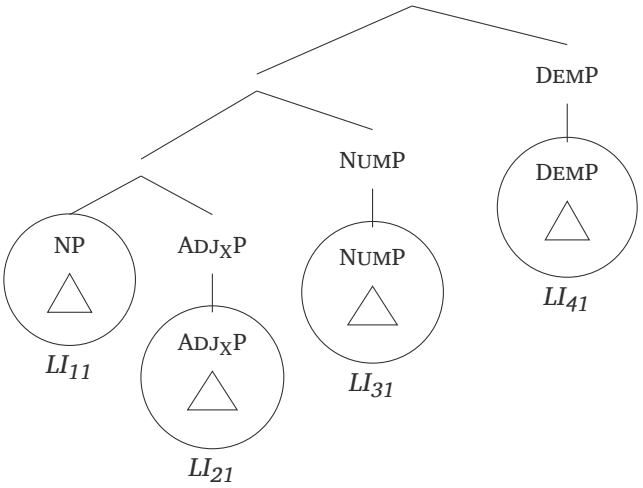


Finally, the next cycle in the derivation is triggered by merger of DEMP (47) and follows the same steps. In lack of any dedicated second-order LI, no successful match can be found until the algorithm triggers roll up of the entire complement of DEMP, as shown in (48). This again produces a configuration in which the independent lexicalisation previously found for each branch is capitalized on to lexicalise the full structure, yielding N-Adj-Num-Dem.

(47)



(48)



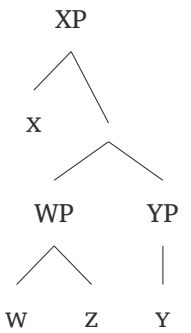
4.3 Cases of (sub)extraction

As encoded by the Lexicalisation Algorithm, the derivational option of roll-up is not the only available one. In this subsection, we discuss two different linear orders that involve the only other type of scenario: (sub)extraction out of a complement.

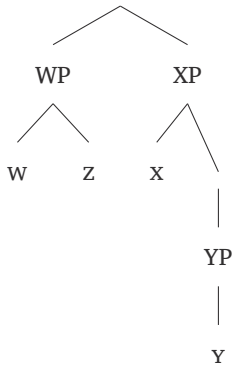
Following the step b. and c. of the algorithm, what constituents can move is determined by the geometry of the tree, and lexicalisation-driven movements are attempted starting from the closest labelled movable branch. (Sub)extraction is a descriptive label for situations in which the moved branch is smaller than the complement of the newly merged F/FP. In such cases, lexicalisation-driven movement leaves a remnant in the complement of the newly merged F/FP, by either moving the left branch of the complement (extraction) or a subpart of an already moved constituent (subextraction).

To visualise the relevant type of configuration, consider the abstract structure in (49a). If direct lexicalisation fails, step b. of the algorithm dictates lexicalisation-driven movement of WP. The result of such movement is the structure in (49b), where the complement of the new feature X contains a remnant of the original complement, i.e. YP.

(49) a.



b.



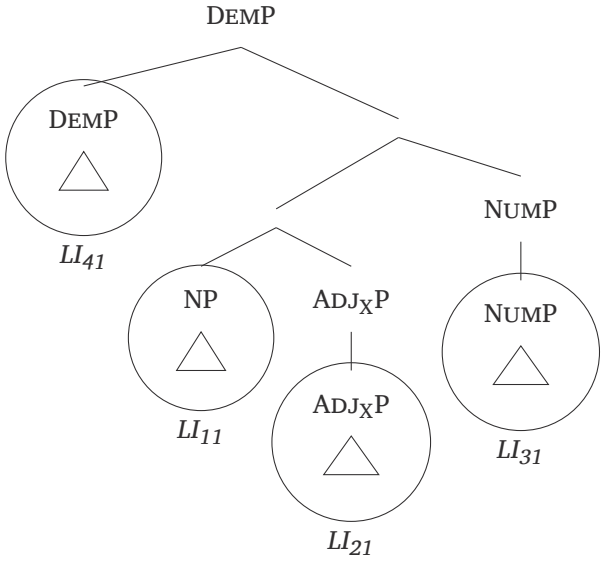
Before this lexicalisation-driven movement, the constituent to lexicalise – XP – contained all other branches. After extraction of WP, XP can be lexicalised by an LI that contains X and YP in the corresponding configuration, like (50).

(50) LI_n: ⟨ /fəˈnɑləʤi/, XP, [ ] ⟩

In the present context, the question is what kind of LIs can match one such (sub)extraction configuration derived after the application of MERGE FP. We argued in Section 3.2 that LIs containing an L-tree with one concrete syntactic syntactic configuration as (50) are not adequate. Defining the order of complex XPs standardly requires to abstract over all their possible syntactic configurations. LIs containing a pointer to WSp_{main} of the type we introduced in Section 3.2 are not suitable either. The current state of WSp_{main} contains the syntactic structure with which the new branch has been merged. This only allows direct lexicalisation after MERGE FP, regardless of the internal articulation of the complement of the new branch. What needs to be encoded to capture cases of (sub)extraction is that a successful lexical match is conditional on the application of lexicalisation-driven movements which evacuate a subpart of the complement of the new branch (i.e. of the syntactic structure stored in WSp_{main}), leaving a remnant. Following the logic of lexicalisation, we need a second-order LI whose right branch matches the remnant WSp_{main}: WSp_{main} *minus* the evacuated XP.

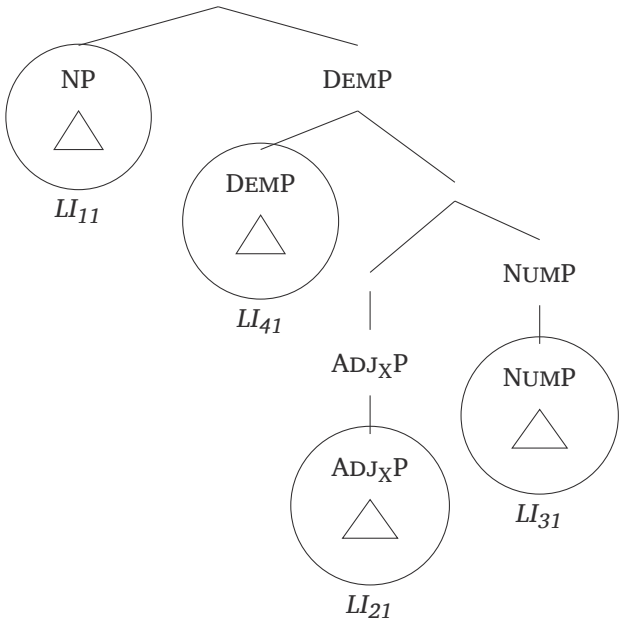
To better understand the issue, let us consider the S-tree in (51). Parallel to (49a) above, this tree is the result of the application of roll-up, in this case involving movement of NP across ADJ_XP, and of [[NP] [ADJ_XP]] across NUMP, as outlined in Section 4.2. The output of this successive application of roll-up is then merged with DEMP, which projects its label at the root node.

(51)

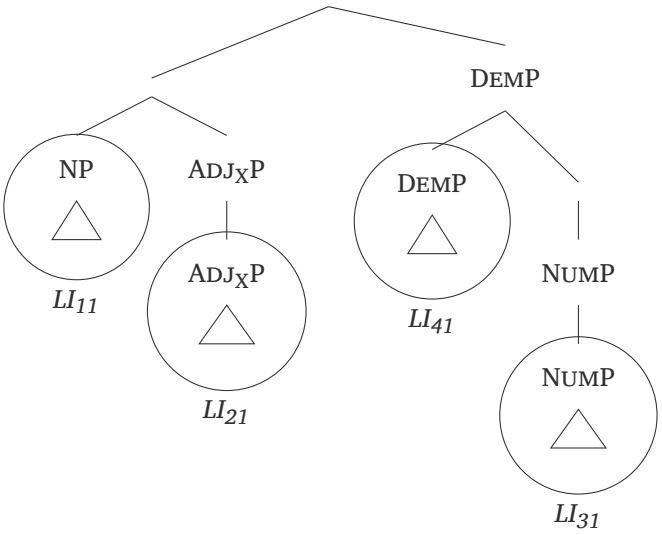


(51) is compatible with multiple derivational possibilities, including two (sub)extraction cases. By step b. of the Lexicalisation Algorithm, the first attempt involves movement of the closest labelled constituent containing the lowest feature of the current *fseq*, i.e. NP (52); in case of failure, step c. will target [[NP] [AdjXP]] (53), and, as a last resort, the entire complement of DemP (i.e. roll-up) (54). Each option yields a different order: N-Dem-Adj-Num, (line p. in **Table 1**), N-Adj-Dem-Num, (line l. in **Table 1**), and N-Adj-Num-Dem (line y. in **Table 1**). While (54) is derived in case no second-order LI is present (cf. Section 4.2), both (52) and (53) require the remnant DemP under the right node to be matched.

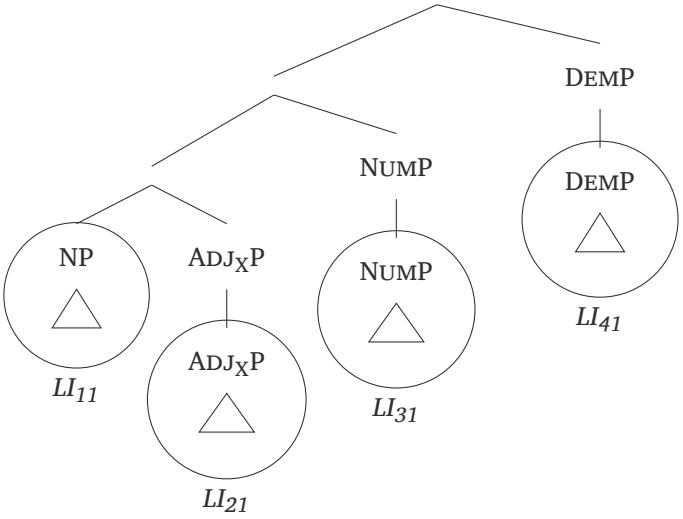
(52)



(53)



(54)



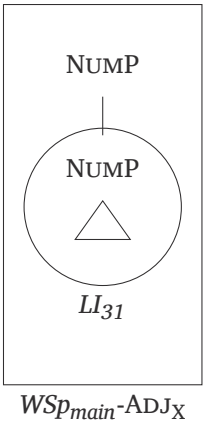
The implementation we propose for these cases starts from the reasoning we applied in Section 3.2 to allow for direct matches after MERGE FP. To abstract away from the content of the complement of the new branch, we resorted to pointing to WSp_{main}, which contains the syntactic structure lexicalised in the main workspace at the previous cycle. In proposing that L-trees can point to WSp_{main}, we leverage the fact that such objects must be independently assumed to be available in the derivation and, therefore, visible to syntax (cf. Section 3.1). Similarly, the *fseq* – the universal sequence of features/categories – is not itself a syntactic object, but interacts with syntax in specifying the licit orders of merge of the relevant features/categories (cf. Section 3.1). We hence take the *fseq* and its categories to be visible to syntax, so that they can be referred to within specific L-trees. To account for cases of (sub)extraction, we then enrich the expressive possibilities of WSp_{main} with the possibility to specify "cut-off points" in terms of features/categories of the

fseq. Given a (simplified) *fseq* <N, Adj_X, Num, Dem>, we propose second-order LIs like those in (55), which specify different "cut-off points".

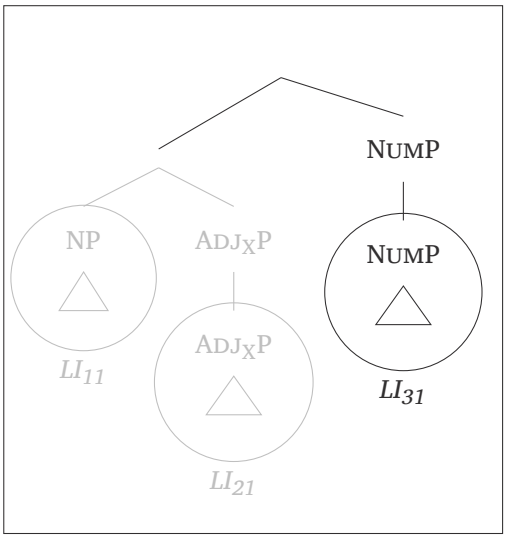
(55) LI_{78}: ⟨ DemP ⟩ LI_{79}: ⟨ DemP ⟩

Specifically, we propose that the relevant diacritics specify a category in the *fseq* such that a match is found only if the right branch does not contain any element of that category or of a lower one in the *fseq*. To better define how this works, let us repeat how the Matching Condition applies with L-trees containing a pointer. When a pointer is found in the L-tree, the matching process checks if the information present in the pointed node matches the corresponding node in the S-tree. As discussed above, if the pointed node is an LI, identity must be established with the L-tree of that LI. If the pointer instead refers to a workspace, the S-tree under the corresponding node must be identical to a tree that has a lexicalisation in that workspace. Informally, diacritics like WSp_{main}-ADJ_X mean that the procedure compares the relevant portion of the S-tree with the tree contained in WSp_{main} after the "removal" of any element of the same category as the specified "cut-off point" or of a lower category in the *fseq*. This "removal" operation can be understood in two possible ways. The first is to interpret it in terms of the creation of a copy of WSp_{main} out of which the closest phrase containing all and only the relevant categories is removed. The second is to interpret it as the instruction to ignore the lexicalisation of the relevant categories in WSp_{main}. Considering (51), the result of interpreting the diacritic WSp_{main}-ADJ_X in LI_{79} under the first option is represented on the left in (56), while the result of the second option is on the right.

(56) a.



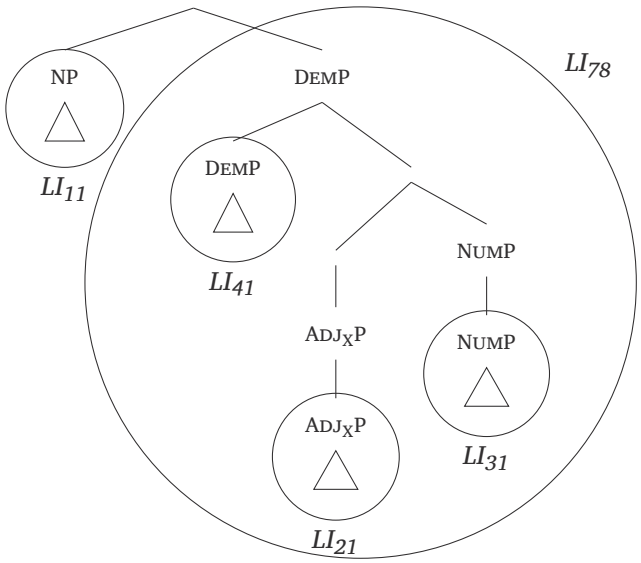
b.



Concretely, LI_{79} matches (55) any configuration in which the complement of DEMP *does not* contain ADJ_X or any constituent with a lower label in the *fseq*. As we detail in Appendix 2, this correctly captures all possible orders in cases of gaps, namely in cases in which one or more modifiers are not part of the structure.

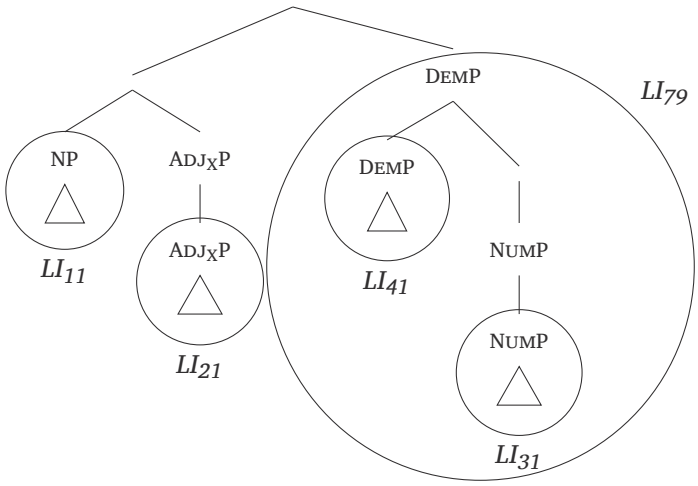
To show these two different second-order LIs at play, consider the different results they yield after merger of DEMP in a configuration like (51) above. A language with LI_{78} finds a match as soon as the first lexicalisation-driven movement – movement of NP – is attempted, yielding the order N-Dem-Adj-Num (57) (= (52)).

(57)



On the other hand, a language with an LI like LI_{79} requires a further movement attempt to find a match, moving a larger constituent including the noun and the adjective across DEMP. This yields N-Adj-Dem-Num (58).

(58)



4.4 Unattested orders and impossible operations

This section outlined how the present systems replicates the different derivational scenarios employed by Cinque (2005) and ff., reducing them to an interaction between second-order LIs and the Lexicalisation Algorithm. The possibility of having LIs that license a direct match after the merger of a new branch corresponds to a situation where no movement occurs, or in more recent versions of Cinque's approach, to cases of "pictures-of-whom pied-piping" (movement of a head-final phrase in Cinque's (2023) terminology). The lexicalisation-driven movements made available by the algorithm cover instead the other two cases in Cinque's system, i.e. movement without "pied-piping" (e.g. movement of N alone) or with "whose-pictures pied-piping" (e.g. roll-up movement of N-Adj across Num).

In this system, other derivational scenarios are not available. The consequence is that the ten unattested orders in **Table 1** are not derivable. To repeat, this follows from the restrictions on licit movements encoded in the Lexicalisation Algorithm, repeated in (59).

(59) a. LEXICALISE FP
b. LEX-DRIVEN MOV I: If fail, evacuate the closest labelled constituent that is the output of a previous derivational cycle in the current workspace, re-try a.
c. LEX-DRIVEN MOV II: If fail, evacuate the immediately dominating constituent, re-try a. (recursive)

Crucially, step b. and c. restrict the targets of movement to constituents containing the lowest feature in the current *fseq* (see the discussion in Section 3.2). This has the same effect achieved in Cinque (2005; 2023) by restricting licit movements to constituents containing the lexical head.

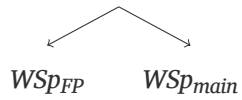
Like proposals by Cinque (2005; 2023) and Abels & Neeleman (2009; 2012), this system incorporates the restriction axiomatically. In that respect, an attempt to go beyond an axiomatic approach is put forth by Steddy & Samek-Lodovici (2011). Their analysis focusses on ruling out movements of constituents from which the lexical head of the extended projection has been extracted (i.e. remnant movements). This ban depends on a set of Optimality Theory constraints imposing that every category of the DP must be left-aligned with the DP's left edge. Under this approach, the strings generated by remnant movement are always sub-optimal with respect to at least one string obtained without remnant movement under any possible ranking of the constraints. Crucially, however, this does not cover the necessary ban on movement of modifiers alone, for which additional assumptions are required. As discussed in Section 3.2, the alternative we propose identifies the proper targets of lexicalisation-driven movement with structures that represent a successful output of previous lexicalisation cycles in the current workspace. This correctly excludes both remnant movement and movement of modifiers alone. Moreover, defining the possible targets of movements in terms of derivational stages could also to capture the overall restriction in a non-axiomatic way. In this sense, one should investigate if there is a

principled reason why lexicalisation-driven movements and the configurations that have already been lexicalised are linked, something we necessarily leave to future research.

5 What is under the left node

Having presented the role of second-order LIs in the lexicalisation-driven derivation of U20, in this section we elaborate on further details of their internal structure. Our formalisation employs second-order LIs like (60):

(60) LI_{58}: ⟨ ADJ_XP ⟩



Under this implementation, any structure produced by merging to the current tree a phrasal constituent that projects the category ADJ_XP will be matched. This approach captures the observation that the complexity of the new branch generally does not affect the final word-order.[21] For instance, further modifying an adjective with a degree modifier like *very* does not change the word-order (En. *black dog* vs. *very black dog*).

If this were the only available format in our grammar for second-order LIs, a given functional category would always yield the same word-order. Concretely, for languages where the order of (direct modification) Adj and N depends on the sub-category of the Adj (as Italian, see Cinque 2010), this would imply the existence of a functional head for each relevant sub-category. If so, a different second-order LI for each functional head would be stored in the lexicon, each potentially instantiating a different word-order.

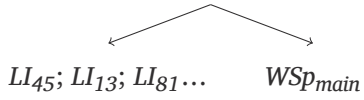
While this is a plausible and familiar solution within the Cartographic framework, the core of our proposal does not hinge on this. In fact, our introduction of second-order LIs pointing to workspaces like (60) in Section 3.2 is logically independent from the possibility of second-order LIs pointing to LIs. On the contrary, these would have to be ruled out by stipulation. Since pointing to LIs remains a possibility next to pointing to workspaces, this essentially defines "rules" of different specificity. Under the assumption that more specific rules are only posited if more general ones fail, the system can therefore capture exceptions/more specific word-order rules via second-order LIs pointing to LIs, without losing its generality. Here, we briefly speculate on the format and the possible empirical correlates of these more specific second-order LIs.

Concerning the format, note that L-trees store configurations of objects visible to the derivation. Necessarily, these must include features, workspaces, LIs, and the Lexicon. Under the assumption that the Lexicon is but a list of LIs, the more abstract format of L-trees proposed

[21] On cases of complements of adjectives leading to reversing the "regular" word-order (e.g. *a proud man* vs. *a man proud of his daughter*), see Williams (1982), Emonds (1976).

here allows reference to lists of LIs of different sizes (including lists with only one member). As a tentative exploration, this reasoning allows LIs that refer to a list of LIs in their left branch, as in (61).

(61) LI_{58}: ⟨ ADJ_XP ⟩



The LI in (61) states that if the branch satisfying the feature ADJ_X is lexicalised by any of the LIs specified in the list under the left node, the order new branch – main derivation is licensed.

A crucial property of this format is that it refers to specific LIs, making it possible to embed a degree of lexical specificity in word-order rules. In other words, two adjectives satisfying the same functional feature ADJ_X could in principle be linearised before or after the rest of the nominal derivation, depending on their presence or absence from the list under the left node. Following this logic, sub-regularities can be captured without necessarily stipulating new, more specific categorical distinctions. Namely, LIs like (61) allow to capture cases in which single LIs or groups of LIs might deviate in their word-order properties from the general class of items of the same grammatical category. From a more general perspective, it becomes possible to elaborate on the properties that can lead the speakers to 'group together' different LIs. Such properties can also be extra-grammatical (e.g.the adjectives belonging to the same conceptual area), or based on grammatical but not syntactic features (e.g. phonological similarity, metric structure).

Despite the potential positive outcomes of the format in (61) in terms of modelling lexical exceptions and non-syntactic grouping factors, an issue remains open: suffixation. Within Nanosyntax, a suffix is inserted to lexicalise the right branch of the structure after the application of lexicalisation-driven movements, which create unlabelled nodes. The consequence is that a form like *color-ful* is dominated by an unlabelled node, and does not require a single matching LI. But this in turn means that there is no single LI that can be referred to within a LI like (61). A way around this issue is to assume that root-suffix complexes are stored as a second-order LI themselves, making it possible to refer to them as a unit. We leave the issue open as a *caveat* on a possible implementation in terms of lists of LIs.

6 What second-order LIs might be good for

A general consequence of our proposal is that different word-orders can be understood as reflecting the properties of objects stored in the lexicon. This makes it possible to address questions concerning how these objects are acquired by learners and how they interact with the syntactic derivation and with each other. In this section, we sketch how such questions can be further pursued in two distinct domains, namely the typological distribution of word-orders and syntactic priming.

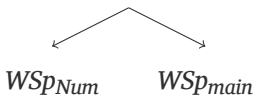
6.1 The lexical information for homomorphic orders is easier to acquire

As highlighted in the literature on the Universal 20, not all attested linear orders in nominal phrases are equally frequent across languages. The results of a series of artificial language learning experiments (Culbertson & Adger 2014; Martin et al. 2020; 2024) suggest that among the determinants of the typological frequency is a general cognitive bias towards transparent mappings between the hierarchical arrangement of items and their linear placement. What these studies observed is that participants learning nominal phrases in an artificial language tend to generalise *homomorphic* orders, in which items are linearly adjacent to items/phrases they dominate in the hierarchy. Crucially, this was found to be the case also when testing speakers who do not have superficial evidence for homomorphism in their language, as in the case of Kîîtharaka (Bantu) (Martin et al. 2024). Since precisely homomorphic orders are among the crosslinguistically most common ones, such experimental evidence can be invoked to support the idea that the same cognitive bias observed in the performance of participants is also responsible for the typological tendencies in the distribution of word-orders.

This argument leaves an open question: why would homomorphism in itself play a role? As it turns out, modelling the different orders in terms of LIs and lexicalisation-driven operations helps highlight a possible rationale for why homomorphic orders tend to be preferred, namely the fact that they require less input to be acquired.

To see how, consider again the question of how the parsing of simple N + modifier pairs relates to the acquisition of the target word-order in more complex nominal structures. Assuming a conservative lexical acquisition process, parsing a Num-N sequence will lead to positing an LI with the format of (62), while N-Num will support the deduction that no LI is needed, as the order is derived via roll-up (cf. Sections 3.2 and 4).

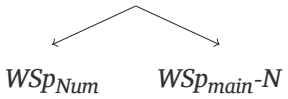
(62) LI_{72}: ⟨ NumP ⟩



Now, all homomorphic orders can be fully characterised in terms of either of these options for each modifier. The consequence is that generalisations based on simple pairs of items are enough to correctly capture the behaviour of the same items in more complex structures. LI_{72} in (62) will always license configurations in which Num comes before the lower parts of the structure, regardless of whether the noun combines with lower modifiers, and roll-up across Num will target a constituent that contains any item below it in the hierarchy. Thus, for instance, the information that can be deduced upon encountering N-Adj and N-Num is already enough to derive N-Adj-Num, without the need of additional, more specific input. Non-homomorphic orders, on the other hand, do not share this property. The reason is that their derivation always involves a

(sub)extraction step, as outlined in the previous section, and pairwise input is not enough to posit the LIs that license it. In the example at hand, the information required for a target non-homomorphic sequence like N-Num-Adj can only be fully obtained after directly encountering a nominal phrase where the noun combines with both a numeral and an adjective, which will lead to positing a second-order LI like (63).

(63) LI_{82}: ⟨ NUMP ⟩



More abstractly, the proposal allows to formulate the following hypothesis: systems where the target LIs can be acquired based on shorter sequences of elements tend to be more frequent. While the empirical consequences of such logic remain to be explored, the crucial point is that it enables a concrete perspective on the typological frequency of homomorphic orders, based on an intuitively plausible claim: systems that can be learned reliably on the basis of fewer information tend to be preferred.

Note that the possibility of stating this in terms of 'ease' of lexical acquisition is directly connected to the proposal that word-order information is stored in terms of LIs. At the same time, an analogous reasoning about the role of homomorphism can be developed under the post-syntactic approaches discussed in Section 2 (Abels & Neeleman 2009; 2012; Manzini Submitted).[22] There too, a difference is encoded between homomorphic and non-homomorphic orders: the first can be fully captured in terms of simple linearisation statements, provided that different modifiers like e.g. adjectives and numerals are adjoined to hierarchically different categories. Non-homomorphic orders, instead, additionally require movement of a category containing the nominal head.[23] Under the assumption that linearisation statements are generalised as the default option to capture the order of two items, and meaningless movement is only posited as a last resort, the successful acquisition of a non-homomorphic order like N-Num-Adj requires input providing evidence for N-movement, i.e. minimally the sequence N-Num-Adj itself.

6.2 Second-order LIs and syntactic priming

The previous section focused on testing our theoretical proposal considering empirical data from the typological tradition. In a similar vein, our proposal can also be connected to experimental measures such as syntactic priming. Priming refers to the facilitative effect that the processing of a linguistic expression has on the production or comprehension of the same or a related linguistic

[22] We thank an anonymous reviewer for pointing this out.

[23] To be precise, Manzini (Submitted)'s proposal rather involves a distinct type of post-syntactic statement concerning the position where the noun is pronounced. To the extent that two independent types of devices can derive the same surface string, the argument equally applies.

expression (Tulving et al. 1982). This phenomenon has been shown to be consistently attested in the production and comprehension of syntactic structures independently of other linguistic layers (see the seminal works in Bock 1986; Bock & Loebell 1990; Pickering & Branigan 1998; for a review see Branigan & Pickering 2017; see also the meta-analysis in Mahowald et al. 2016 showing how the syntactic priming effect in production is robust across constructions and languages).

Crucially for our discussion, priming processes have been claimed to tap into our linguistic representations and their inter-relations. On this basis, syntactic priming has been proposed as a methodological tool for probing the representational units shared between syntactic representations *via* their activation (see Branigan & Pickering 2017[24]). From a generative perspective, however, it is unclear what the objects of such activation might be. Now, our approach provides a way to model syntactic activation along the more familiar lines of lexical activation: since any syntactic structure is licensed via lexicalisation, parsing a given input string results in the activation of the LIs required to license it, including second-order LIs. If syntactic priming involves the activation of second-order LIs, we expect their formal properties to be compatible with the general findings from syntactic priming experiments. This is what we show in the next paragraphs, before looking at some further predictions testable via the same methodology.

The most consistent and replicated finding from syntactic priming (Bock 1986; 1989; Bock & Loebell 1990, among others) is that syntactic information can be primed autonomously; syntactic priming effects do not depend on the repetition between the prime and the target of LIs with a specific externalisation (irrespective of these LIs being more "functional" or "lexical"). In our model, second-order LIs only encode syntactic information; no externalisation (or conceptual) information is present (see Section 3.2). If syntactic priming involves the activation of such LIs, independence from externalisation is expected. Besides externalisation, syntactic priming has also been proved to be independent from the complexity of the internal structure of the constituents involved (Pickering & Branigan 1998) and the general syntactic context in which the given structure is present (Branigan et al. 2006). This too is expected in our system. Second-order LIs encode local relationships between the syntactic constituent projecting a given functional category and the rest of the tree. Crucially, both the new branch and the rest of the derivation are pointed at as unanalysed units (workspaces), so that their internal syntactic structure is not decisive for the possibility of priming the whole second-order LI.

At the same time, the magnitude of the effect has been shown to significantly increase with the repetition of lexical material from the prime to the target sentence (lexical-boost effect; Pickering & Branigan 1998; Branigan et al. 2000; Cleland & Pickering 2003). The lexical-boost effect is

[24] See the peers' comments for criticism, especially on the fact that priming might not be sensible enough to detect *all* differences between syntactic representations. However, there is crucially agreement on the fact that positive evidence of priming can be used for investigating syntactic representations.

however short-lived compared to the overall priming effect, to the extent that it entirely vanishes when material is found in between the prime and the target (Hartsuiker et al. 2008; Mahowald et al. 2016). As argued above, our second-order LIs do not directly contain overt lexical material and can therefore be activated independently of it. Nonetheless, the S-trees they match always necessarily contain overt lexical material in each workspace. Such information is inherited at the root node. We might then speculate that the lexical-boost effect stems from the co-activation of a second-order LI with the inherited lexical material. Repetition of the same lexical material in the next target enhances retrieval of the second-order LI (explicit memory as a retrieval cue, see Hartsuiker et al. 2008). While this enhancing-by-repetition effect is constrained to the immediately adjacent production, the implicit activation of the abstract second-order LI outlives it and persists over time.

Overall, the model we propose is balanced enough between abstraction and overt externalisation to account for these results. On the one hand, the lexicon contains a set of abstract "ordering instructions" represented as second-order LIs just encoding the relative position of the two branches they dominate (no externalisation or conceptual information). These branches are in turn represented as unanalysed units (*WSp*). On the other hand, such second-order LIs are necessarily always matched *via* the Nanosyntactic lexicalisation mechanism with actual syntactic derivations containing overt lexical material.

On top of being generally compatible with previous experimental evidence on syntactic priming, our system lends itself to further predictions in this domain. Here we briefly sketch three potential lines of inquiry.

Let us take the basic order between adjective (Adj) and noun (N). Our system makes a crucial difference between the Adj-N and the N-Adj order: the first requires a second-order LI stored in the lexicon licensing it (cfr. (38a)), the second is the result of the absence of any second-order LI ordering N and Adj (cfr. (43)). If true, this implies an asymmetry also in the priming effect between these two conditions. We moreover predict that the asymmetry should favour the Adj-N order, as activating an LI via priming should be less demanding than inhibiting one. To properly test this, one should investigate a bilingual population with language A in which only the Adj-N order is grammatical and language B where only the N-Adj order is grammatical.[25] Finding a difference in the magnitude of the priming effect in favour of the Adj-N order would be consistent with our proposal.

Besides the asymmetry between presence and absence of a second-order LI, it is also possible to investigate the difference between the activation of one vs. two (or more) second-order LIs. Let us assume Language A, where the unmarked order is N-Adj-Num-Dem, Language B, where the

[25] Priming studies already investigated the order Adj-N and N-Adj, also in crosslinguistic settings (Hsin et al. 2013; Van Dijk & Unsworth 2023). All of them, however, involved a Romance and a Germanic variety, where the Romance variety has both orders.

unmarked order is Dem-Num-N-Adj, and Language C, where the unmarked order is Dem-N-Adj-Num. Language A requires 0 second-order LIs, Language B requires 2 second-order LIs, Language C requires 1 second-order LI (see Appendix 1). Following our proposal, it should be easier to prime the order of Language C to Language A than the order of Language B to Language A. Along the same lines, priming the order of Language B to the order of Language C (and vice versa) should give comparable results to priming the order of Language C to Language A, since in these cases the difference between the regular and the primed order amounts to one LI.

Finally, it is interesting to question the level of abstraction at which syntactic priming can take place. Is it possible to activate the abstract form of a second-order LI and transfer it to a separate second-order LI which refers to a different functional feature? To give a concrete example, would it be possible to transfer the format of the second-order LI ordering Adj before N to the functional level of the demonstrative, activating the order Dem-N? On a more speculative level, a positive answer to this question would partially explain the level of homogeneity or harmony that is found in natural languages (see Section 6.1): second-order LI tend to attract each other, and maybe more so if they are often found together, as when they all belong to the same functional domain (e.g. the nominal or the verbal domain).

7 Conclusions

In this contribution, we contended that crosslinguistic differences in basic word-order depend on variations in the application of movements of syntactic constituents, in line with "syntax-based" approaches relying on a universal linearisation procedure (Cinque 2005; 2023). Unlike these approaches, we model these movements as a reaction to an interface condition on lexicalisation, whereby a syntactic constituent is licit iff each labelled node it contains is matched by a Lexical Item in the language-specific lexicon. When this is not the case, the fixed series of movements codified in the Nanosyntactic Lexicalisation Algorithm (Starke 2009; 2018; Caha et al. 2024) applies, to give lexicalisation a second chance (Section 3). These movements can only target constituents that contain the lowest feature of the *fseq*. In Section 3.2, we argued that this amounts to restrict movement to trees that were the output of a derivational cycle in the current workspace. As a result, the system mirrors Cinque's restriction on meaningless movement and stands the test of the U20 generalisation (Greenberg 1963; Cinque 2005; 2010; Abels & Neeleman 2009; 2012), in that it correctly rules in all the attested orders and rules out all the unattested ones. (Section 4).

From a theoretical perspective, this lexicalisation-based approach preserves the advantages of "syntax-based" approaches while avoiding the problematic conflation of meaningless and meaningful movements. The first find a place in the theory of externalisation as lexicalisation-driven operations that only affect the structural arrangement of one and the same functional content, and therefore lack any semantic correlate. The second are instead systematically

associated to structural dependencies that have a semantic/functional reflex. Since the two operations have distinct types of triggers, it also comes as no surprise that they obey a different set of constraints (as noted in Abels & Neeleman 2009; 2012, among others). Furthermore, the model encodes the information determining which orders are licensed in a given language in the form of Lexical Items. This allows to reduce crosslinguistic word-order variation to an independently needed source of variation: the content of the language-specific lexicon.

On an empirical level, the proposal allows to model observational findings in the domain of typological markedness (Section 6.1) and syntactic priming (Section 6.2) based on the properties of the syntactic representational units implied by each word-order, i.e. the LIs.

Supplementary files

Supplementary file 1: Appendix 1. Full list of LIs for each attested base word-order for Dem, Num, Adj, and N. DOI: https://doi.org/10.16995/glossa.18521.s1

Supplementary file 2: Appendix 2. Cases of 'gaps'. DOI: https://doi.org/10.16995/glossa.18521.s2

Acknowledgements

Many friends and colleagues have been instrumental in shaping the final version of this work. In particular, we would like to thank those who read and provided thorough comments on earlier versions, including Michal Starke, Cecilia Poletto, Guglielmo Cinque, Pierre Larrivée, Giacomo Presotto, and Tommaso Balsemin. Further crucial input came from conversations with Maria Rita Manzini, Pavel Caha, Karen de Clercq, Emanuela Sanfelici, Nicolas Lamoure, Edoardo Cavirani, Jeroen van Craenenbroeck. We would also like to thank the audience of the *Incontro di Grammatica Generativa* (IGG50, Padova) and of the *CRISSP Seminars* for their valuable feedback. Finally, we warmly thank the Editor, Guido Vanden Wyngaerd, for his advice and support, as well as our three anonymous reviewers, who provided extremely valuable comments that helped make our proposal substantially clearer and more precise. All remaining errors are, of course, our own.

Competing interests

The authors have no competing interests to declare.

References

Abels, Klaus. 2003. *Successive cyclicity, anti-locality, and adposition stranding*. University of Connecticut dissertation.

Abels, Klaus. 2016. *Neutral word order and stack sorting*. University College London: Workshop on 'Movement'.

Abels, Klaus & Neeleman, Ad. 2009. Universal 20 without the LCA. In Brucart, José M. & Gavarró, Anna & Solà, Jaume (eds.), *Merging Features*, 60–79. Oxford: Oxford University Press. DOI: https://doi.org/10.1093/acprof:oso/9780199553266.003.0004

Abels, Klaus & Neeleman, Ad. 2012. Linear asymmetries and the LCA. *Syntax* 15(1). 25–74. DOI: https://doi.org/10.1111/j.1467-9612.2011.00163.x

Baunaz, Lena & Lander, Eric. 2018. Nanosyntax: The basics. In Baunaz, Lena & Haegeman, Liliane & De Clercq, Karen & Lander, Eric (eds.), *Exploring nanosyntax*, 3–56. Oxford: Oxford University Press. DOI: https://doi.org/10.1093/oso/9780190876746.001.0001

Blix, Hagen. 2021. Phrasal spellout and partial overwrite: On an alternative to backtracking. *Glossa: a journal of general linguistics* 6(1). 1–17. DOI: https://doi.org/10.5334/gjgl.1614

Bock, J. Kathryn. 1986. Syntactic persistence in language production. *Cognitive Psychology* 18(3). 355–387. DOI: https://doi.org/10.1016/0010-0285(86)90004-6

Bock, J. Kathryn. 1989. Closed-class immanence in sentence production. *Cognition* 31(2). 163–186. DOI: https://doi.org/10.1016/0010-0277(89)90022-X

Bock, J. Kathryn & Loebell, Helga. 1990. Framing sentences. *Cognition* 35(1). 1–39. DOI: https://doi.org/10.1016/0010-0277(90)90035-I

Branigan, Holly P. & Pickering, Martin J. 2017. An experimental approach to linguistic representation. *Behavioral and Brain Sciences* 40. e282. DOI: https://doi.org/10.1017/S0140525X16002028

Branigan, Holly P. & Pickering, Martin J. & Cleland, Alexandra A. 2000. Syntactic co-ordination in dialogue. *Cognition* 75(2). B13–B25. DOI: https://doi.org/10.1016/S0010-0277(99)00081-5

Branigan, Holly P. & Pickering, Martin J. & McLean, Janet F. & Stewart, Andrew J. 2006. The role of local and global syntactic structure in language production: Evidence from syntactic priming. *Language and Cognitive Processes* 21(7–8). 974–1010. DOI: https://doi.org/10.1080/016909600824609

Caha, Pavel. 2009. *The nanosyntax of case*. Universitetet i Tromsø dissertation.

Caha, Pavel. 2019. Case competition in Nanosyntax. A study of numeral phrases in Ossetic and Russian. *LingBuzz* lingbuzz/004875.

Caha, Pavel & De Clercq, Karen & Starke, Michal & Wyngaerd, Guido Vanden. 2024. Nanosyntax: State of the art and recent developments. *LingBuzz* lingbuzz/007744.

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

Chomsky, Noam. 2001. Derivation by phase. In Kenstowicz, Michael (ed.), *Ken Hale: A life in language*, 1–52. Cambridge, MA: MIT Press. DOI: https://doi.org/10.7551/mitpress/4056.003.0004

Chomsky, Noam. 2013. Problems of projection. *Lingua* 130. 33–49. DOI: https://doi.org/10.1016/j.lingua.2012.12.003

Chomsky, Noam. 2021. Minimalism: Where are we now, and where can we hope to go. *Gengo Kenkyu* 160. 1–41. DOI: https://doi.org/10.11435/gengo.160.0_1

Cinque, Guglielmo. 1999. *Adverbs and functional heads: A cross-linguistic perspective*. Oxford: Oxford University Press. DOI: https://doi.org/10.1093/oso/9780195115260.001.0001

Cinque, Guglielmo. 2005. Deriving Greenberg's Universal 20 and its exceptions. *Linguistic inquiry* 36(3). 315–332. DOI: https://doi.org/10.1162/0024389054396917

Cinque, Guglielmo. 2010. *The syntax of adjectives: A comparative study*, vol. 57. Cambridge, Mass: MIT press. DOI: https://doi.org/10.7551/mitpress/9780262014168.003.0005

Cinque, Guglielmo. 2023. *On linearization: Toward a restrictive theory*. Cambridge, Mass: The MIT Press. DOI: https://doi.org/10.7551/mitpress/14681.001.0001

Cinque, Guglielmo & Rizzi, Luigi. 2010. *The Cartography of syntactic structures*. Oxford: Oxford University Press. DOI: https://doi.org/10.1093/oxfordhb/9780199544004.013.0003

Cleland, Alexandra & Pickering, Martin J. 2003. The use of lexical and syntactic information in language production: Evidence from the priming of noun-phrase structure. *Journal of Memory and Language* 49(2). 214–230. DOI: https://doi.org/10.1016/S0749-596X(03)00060-3

Cortiula, Maria. 2023. *The Nanosyntax of Friulian verbs: An analysis of the present and past in Tualis Friulian*: Masaryk University Brno Doctoral Thesis.

Culbertson, Jennifer & Adger, David. 2014. Language learners privilege structured meaning over surface frequency. *Proceedings of the National Academy of Sciences* 111(16). 5842–5847. DOI: https://doi.org/10.1073/pnas.1320525111

De Belder, Marijke & van Craenenbroeck, Jeroen. 2015. How to merge a root. *Linguistic Inquiry* 46(4). 625–655. DOI: https://doi.org/10.1162/LING_a_00196

Emonds, Joseph. 1976. *A transformational approach to English syntax: Root, structure-preserving, and local transformations*. New York: Academic Press.

Gök, Serra & Demirok, Ömer. In press. On the non-uniform nature of auxiliaries: A case study on Turkish. In Caha, Pavel & De Clercq, Karen & Vanden Wyngaerd, Guido (eds.), *Nanosyntax and the Lexicalisation Algorithm*, 313–338. Oxford: Oxford University Press.

Greenberg, Joseph H. 1963. Some universals of grammar with particular reference to the order of meaningful elements. In Greenberg, Joseph H. (ed.), *Universals of language*, 40–70. Cambridge, Mass: MIT Press.

Grohmann, Kleanthes K. 2011. Anti-Locality: Too-close relations in grammar. In Boeckx, Cedric (ed.), *The Oxford Handbook of Linguistic Minimalism*, 260–290. Oxford: Oxford University Press. DOI: https://doi.org/10.1093/oxfordhb/9780199549368.013.0012

Hartsuiker, Robert J. & Bernolet, Sarah & Schoonbaert, Sofie & Speybroeck, Sara & Vanderelst, Dieter. 2008. Syntactic priming persists while the lexical boost decays: Evidence from written and spoken dialogue. *Journal of Memory and Language* 58(2). 214–238. DOI: https://doi.org/10.1016/j.jml.2007.07.003

Hsin, Lisa & Legendre, Geraldine & Omaki, Akira. 2013. Priming cross-linguistic interference in Spanish-English bilingual children. In Baiz, Sarah & Goldman, Nora & Hawkes, Rachel (eds.), *BUCLD 37 Proceedings*. Somerville, Mass: Cascadilla Press.

Kayne, Richard S. 1994. *The antisymmetry of syntax* (Linguistic Inquiry Monographs 25). Cambridge, Mass: MIT Press.

Kiparsky, Paul. 1973. Elsewhere in phonology. In Anderson, Stephen & Kiparsky, Paul (eds.), *A Festschrift for Morris Halle*, 93–106. New York: Holt, Rinehart & Winston.

Mahowald, Kyle & James, Ariel & Futrell, Richard & Gibson, Edward. 2016. A meta-analysis of syntactic priming in language production. *Journal of Memory and Language* 91. 5–27. DOI: https://doi.org/10.1016/j.jml.2016.03.009

Manzini, M. Rita. Submitted. Left-right asymmetries in Romance (Italian) adjectives: Partial order, phases, gradability.

Martin, Alexander & Adger, David & Abels, Klaus & Kanampiu, Patrick & Culbertson, Jennifer. 2024. A universal cognitive bias in word order: Evidence from speakers whose language goes against it. *Psychological Science* 35(3). 304–311. DOI: https://doi.org/10.1177/09567976231222836

Martin, Alexander & Holtz, Annie & Abels, Klaus & Adger, David & Culbertson, Jennifer. 2020. Experimental evidence for the influence of structure and meaning on linear order in the noun phrase. *Glossa: A Journal of General Linguistics* 5(1). 1–21. DOI: https://doi.org/10.5334/gjgl.1085/

Pickering, Martin J. & Branigan, Holly P. 1998. The representation of verbs: Evidence from syntactic priming in language production. *Journal of Memory and Language* 39(4). 633–651. DOI: https://doi.org/10.1006/jmla.1998.2592

Pinzin, Francesco. 2024. What's hidden below definiteness and genitive: On indefinite partitive articles in Romance. *Linguistics* 62(5). 1251–1300. DOI: https://doi.org/10.1515/ling-2022-0059

Rizzi, Luigi. 1997. The fine structure of the left periphery. In Haegemann, Liliane (ed.), *Elements of grammar*, 281–337. Dordrecht: Springer. DOI: https://doi.org/10.1007/978-94-011-5420-8_7

Starke, Michal. 2009. Nanosyntax: A short primer to a new approach to language. *Nordlyd* 36(1). 1–6.

Starke, Michal. 2014. Towards elegant parameters: Variation reduces to the size of lexically stored trees. In Picallo, M. Carme (ed.), *Linguistic variation in the minimalist framework*, 140–152. Oxford: Oxford University Press. DOI: https://doi.org/10.1093/acprof:oso/9780198702894.003.0007

Starke, Michal. 2018. Complex left branches, spellout, and prefixes. In Baunaz, Lena & Haegeman, Liliane & De Clercq, Karen & Lander, Eric (eds.), *Exploring nanosyntax*, 239–249. Oxford: Oxford University Press. DOI: https://doi.org/10.1093/oso/9780190876746.003.0009

Starke, Michal. 2024a. *The day morphology ate syntax, presentation at Nanodays*. Masaryk University Brno. https://michal.starke.ch/2024/nanodays/#mv-2nd-slot.

Starke, Michal. 2024b. *Nanoseminar. Online lecture series at Masaryk University*. Brno: Masaryk University Brno.

Steddy, Sam & Samek-Lodovici, Vieri. 2011. On the ungrammaticality of remnant movement in the derivation of Greenberg's Universal 20. *Linguistic Inquiry* 42(3). 445–469. DOI: https://doi.org/10.1162/LING_a_00053

Svenonius, Peter. 2008. The position of adjectives and other phrasal modifiers in the decomposition of DP. In McNally, Louise & Kennedy, Christopher (eds.), *Adjectives and Adverbs*, 16–42. Oxford: Oxford University Press. DOI: https://doi.org/10.1093/oso/9780199211616.003.0002

Tulving, Endel & Schacter, Daniel L. & Stark, Heather A. 1982. Priming effects in word-fragment completion are independent of recognition memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 8(4). 336–342. DOI: https://doi.org/10.1037/0278-7393.8.4.336

Van Dijk, Chantal & Unsworth, Sharon. 2023. On the relation between cross linguistic influence, between-language priming and language proficiency: Priming of ungrammatical adjective placement in bilingual Spanish-Dutch and French-Dutch children. *Open Mind* 7. 732–756. DOI: https://doi.org/10.1162/opmi_a_00105

Williams, Edwin. 1982. Another argument that passive is transformational. *Linguistic Inquiry* 13(1). 160–163.