Minimax Feature Merge: The Featural Linguistic Turing Machine

Louis Van Steene
University of Cambridge

Abstract. In Minimalist syntax, linguistic expressions are typically modelled as being projected from a set of lexical items, themselves composed of three independent kinds of features (phonological, syntactic and semantic/pragmatic). The nature of syntactic features has perpetually been confused, and yet they remain the foundation of much of syntactic theory. I contest that an alternative architecture may be preferable in terms of explanatory power within the purview of mathematical biolinguistics, as described by Watumull (2012; 2013; 2015). Namely, I contest that, rather than being the driving force behind syntax, the lexicon is instead distributed amongst the interfaces in the form of non-generative lookup tables, taking Scheer's (2020) view to the logical conclusion, in parallel to DM. Syntax combines syntactic primitives freely except as constrained by the interfaces; these features are atomic, arbitrary (substance-free) computational symbols comprising the set F with cardinality at least one. Following Watumull (2015), language is considered as a mathematical structure, abstracted from its neurological substrate. This structure is isomorphic to a Turing machine, in turn isomorphic to the simplest group-theoretical object, the free magma. The central motivation is the concept of optimality captured in the minimax principle, in turn minimising the burden of the innate first factor and maximising the role of the mathematical laws and heuristics that comprise Chomsky's (2005) third factor. The ultimate aim is to begin to meet the prerequisites of explanation as defined in biolinguistics—learnability and evolvability—by formalising a theory of syntax and its place in the linguistic architecture from the ground up.

Keywords: syntax; minimalism; biolinguistics; features; interfaces; group theory

---

1 Introduction

Formalising the primitive symbols and procedures of linguistic computation is a central goal of linguistic theory, and a prerequisite for understanding the nature of language and its place within the human mind and brain. It is important theory-internally, in order to make it possible to evaluate a theory objectively. It is also important theory-externally, as one evaluates how language fits into a wider theory of cognitive science. In the spirit of the biolinguistic programme, interweaving a number of disciplines is essential to revealing fundamental facts about the nature of the human capacity for language.

In pursuit of this goal, this dissertation constitutes an investigation into the fundamental building blocks of syntactic structures. In Chomsky's (2000; 2001) typical formalisation, an I-language (internal, intensional, individual) L is taken as a device that generates a set of expressions *Exp = hP F, LFi*, i.e., mappings between phonological and logical form. Chomsky denotes the fundamental building blocks of this system as features, which are the properties of language which enter into PF, LF, and the computational system L that generates them. L is a state of the faculty of language FL, 'a component of the human mind/brain dedicated to language' (Chomsky, 2000, p. 89).

(1) An expression *Exp* is an ordered pair *hP F, LFi* where PF and LF are legible inputs to the interface between the syntactic and sensorimotor SM or conceptual-intentional C-I modules, respectively. The syntactic module L generates an infinite set of expressions *{Exp}*.
(2) The Strong Minimalist Thesis (SMT): Syntax is an optimal solution to legibility conditions imposed by the phonological and semantic modules.
(3) UG ('universal grammar') is the initial state of FL.

Significantly, constraining the output to objects of the form *Exp*, in line with the Strong Minimalist Thesis (2), entails a 'Spellout' procedure to be within the syntactic component; namely, a procedure which converts syntactic structures into representations legible by sensorimotor (SM) and/or conceptual-intentional (C-I) systems, respectively. To be explicit about the exact nature of the architecture of grammar requires a clear elaboration of the nature of the interfaces to 'narrow syntax' (NS). This represents a terminological divergence from Chomsky (2001), for whom '[n]arrow syntax maps a selection of choices from Lex to LF' (Chomsky, 2001, p. 11), and one more in line with Boeckx's (2014) conception, which devolves more power to phenomena arising from the interfaces. As such, I adopt more traditional labels for the components of grammar: we have the syntactic component, which is the primary focus of this dissertation; the phonological component, connected to the syntactic component via the phonological interface; and, the semantic-pragmatic component, connected to the syntactic component via the semantic pragmatic interface. The complexities of the latter two components lie well beyond the scope of this work; where reference is made to their inner workings, it is cursory in nature—there is a certain independence afforded to respective theories of each. The syntactic component and the nature of the interfaces in the technical, modular sense are the focus here. Thus, the 'standard working assumption [] that performance systems are external to FL' (Chomsky, 2000, p. 90) is maintained. Section 3 focuses on the aforementioned architectural considerations and proposes a modular architecture, adopting insights from various theoretical approaches.

Another, related issue with the theory presented by Chomsky (2000, i.a.) is in the definition of the lexicon Lex. Chomsky (2001, p. 10) adopts only a vague notion, in which 'Lex [is] in principle 'Bloomfieldian', a 'list of exceptions' that provides just the information required to yield the interface outputs and does so in the best way, with least redundancy and complication'. In a system which maps features to expressions, as above, the organisation of features needs to be established in a clearer way. This is the focus of Section 4, which looks more deeply at the operation at the heart of L, Merge, and how it must operate on individual features. The second operation Chomsky (2001, p. 10) assumes—which 'assembles [features] to lexical items LI of [Lex], the LIs then entering into computations as units'—is thus incorporated into Merge, making the traditional structure of Lex obsolete.

To open, however, a full framing of the research programme to which this dissertation belongs, 'mathematical biolinguistics', is provided in Section 2, which also sets the philosophical stage for the discussion. This is followed by a description of an alternative to the lexicocentric 'Y-model' architecture of grammar in Section 3. Section 4 then focuses on the internal machinery of the syntactic component of this architecture. The ultimate goal is to take a step towards overcoming the granularity mismatch problem, as described by Poeppel and Embick (2005), by proposing a sufficiently fine-grained approach to syntactic structure building and placing this component within a coherent architecture of the language faculty and its place in the mind/brain.

2 Mathematical biolinguistics

2.1 An overview

'Biolinguistics' refers to the study of 'language as a natural phenomenon—an aspect of [man's] biological nature, to be studied in the same manner as, for instance, his anatomy' (Lenneberg, 1967, p. vii). The Minimalist approach, detailed in Section 1, naturally meshes with this paradigm. Furthermore, 'issues of computation [are] central to the biolinguistic enterprise' (Di Sciullo & Boeckx, 2011, p. 6), since it is this computation which is biologically implemented. Issues of computation, and the mathematical theory surrounding it, have been a central part of the history of generative grammar[15]. This dissertation belongs to a subset of the biolinguistic enterprise, explicitly demarcated by Jeffrey Watumull (2012; 2013; 2015): 'mathematical' biolinguistics.

In mathematical biolinguistics, the focus does not lie on the precise neurobiological system – rather, the object of study is the mathematical object which is somehow physically encoded in biological wetware. The formalisation of language into its mathematical fundamentals involves, as Watumull (2015, p. 6) states, the formalising of the objects, functions, and relations of language. The mathematical formalisation is isomorphic to the mental representation, which in turn is isomorphic to its neural implementation, as detailed by Gallistel (2001) in his elaboration of the psychology of mental representations. In other words, '[w]hat matters in representations is form, not substance' (Gallistel, 2001, p. 9692). This dissertation proposes a formalisation of the objects that enter into syntactic computation (features) and the function that combines them (Merge). Note that mathematical biolinguistics is not entirely commensurable with the field of biolinguistics generally (the latter as defined by Boeckx (2013) and Martins & Boeckx (2016)), as indicated directly by Martins and Boeckx (2016, f.n. 4) where 'another author' in this footnote is referring to Jeffrey Watumull[16]. The distinction as relevant within the context of this dissertation, however, is mainly one of focus. Mathematical biolinguistics focuses almost entirely on 'software' concerns, representing a departure from biolinguistics proper which focuses on precise questions regarding the form and evolution of the biological substrate. This effectively follows from Deutsch's (2011) logic: 'if you can't program it, you haven't understood it' (Deutsch, 2011, p. 146, emphasis original). Importantly, in this context, the 'program' must have the quality of 'strong generativity', in other words in must generate the hierarchical structure of language in the same way as the mind/brain, rather than merely equivalent strings—this is what gives the 'program' explanatory adequacy. Indeed, '[t]he understanding of mental software [] precedes and conditions the understanding of neurophysiological hardware' (Watumull, 2012, p. 232), a point which lies at the heart of this Section.

Nevertheless, mathematical biolinguistics is certainly still worthy of the 'biolinguistics' moniker, not least because of the shared central concern of the two enterprises: the 'austere' requirements of learnability and evolvability (Chomsky, 2019b). Both of these constraints are given focused attention as part of Section 4; the latter is expanded upon in the following SubSection.

2.2 Evolvability, economy and the Minimax Theorem

[15] Compare Tomalin (2006) for a historiographical study of the origins and mathematical and philosophical bases of generative grammar.

[16] 'It is worth noticing that Behme (2015) is a continuation of an ongoing discussion with another author, which revolves around what is called 'Chomsky's biolinguistic ontology'—despite the far more ambitious and generic title—and as such qualifies for what we have identified as 'biolinguistics as a Chomskyan enterprise'' (Martins & Boeckx, 2016, f.n. 4). Watumull (2013, p. 302) further confuses the matter as 'my [Watumull's] work and the ontology it assumes are not representative of all biolinguistic research'. Compare Section 2.5 for further analysis of these papers.

The requirement of evolvability deserves some further attention here. It at the very least intersects with computational (and mathematical) optimality, as elaborated by Watumull and Chomsky (2020) and Roberts et al. (in press).

The assumed theoretical basis of the discussion of evolution is grounded in Balari and Lorenzo's (2012) approach, what they refer to as the 'dual nature of language' (Balari & Lorenzo, 2012, p. 5). A 'grammar' consists of an integral part of the human organism combined with a cultural invention. Phenomena like language change belong to the latter source. The former is equivalent to what Balari and Lorenzo call the Central Computational Complex (CCC): a computational system coupled with a collection of interfaces to the peripheral or external systems that it subserves. This dissertation constitutes a preliminary theory of the CCC— although I instead predominantly use the more traditional term 'faculty of language' (FL) to refer to the object of study, in a somewhat looser sense than Chomsky's (2000; 2001) since I include architectural considerations as in Section 3.

In sum, the general idea of 'optimality' is interpreted as equivalent to the Leibnizian sense of economy—a minimax optimisation (see Roberts & Watumull, 2015). In short, the Minimax Theorem, originally formulated by von Neumann (1928), is as such: 'minimise the maximum loss and maximise the minimum gain' (Watumull, 2015, p. 48). Precisely how this applies to language is detailed in Section 4. From the perspective of evolvability, this property is presumed, going 'beyond explanatory adequacy' (Chomsky, 2004), to have resulted from the evolutionary phenomenon of convergence and its effects on the CCC. An intersecting sense is provided by the theory of computational complexity which naturally arises in the context of computational cognitive science and which is the topic of the following SubSection.

2.3 Computability theory and cognitive science

The research programme pursued here is subsumed within a Gallistelian approach to cognitive science, predominantly as summarised by Gallistel and King (2010). The key ideas as relevant in this work are the concept of 'computability' as subsumed by the Turing machine, and the related notion of an 'effective procedure'.

The Turing machine is a definition of computation — one could see it (non-technically) as the 'Platonic form' of computation. Formulated by Turing (1936) as a means of solving the 'Entscheidungsproblem' (Decision Problem), it was proven that the Turing machine can compute everything that is computable, and only that which is computable — the Turing machine is equivalent to the computable functions. As such, it is a foundational component of computability theory (also known as recursion theory). To be clear, '[a] Turing machine is not a physical device—nor an intended model/simulation thereof — but rather a mathematical abstraction representing the functional conditions necessary and sufficient for any system — including the brain — to be computational' (Watumull, 2012, p. 233). 'The Turing machine is a mathematical abstraction rooted in a physical conception' (Gallistel & King, 2010, p. 125); it has three essential components (see Turing, 1936): a symbolic memory (the 'tape'), an ability to read from and write to this memory (the 'read/write head'), and a finite-state processor whose actions are determined by the current state of the processor coupled with the symbol currently being read. The Turing machine renders precise the notion of effective procedure — an algorithm which acts on symbols and produces other symbols. A simplistic way of implementing a procedure is with a lookup table, a one-to-one mapping between inputs and outputs, which quickly becomes impossibly large (and thus time-consuming to traverse) with complex functions. Much investigation into computation is the search of

procedures which are compact, or generative, meaning 'the number of bits required to communicate them (the bits required to encode the algorithm) is many orders of magnitude smaller than the number of bits required to communicate the look-up table for the function realized by the procedure' (Gallistel & King, 2010, p. 301). Generativity 'makes it possible for finite symbolic resources to pick out any referent from an infinitude of possible referents' (Gallistel & King, 2010, p. 81), allowing for d-infinity: '[a] finite system that in principle can [strongly] generate an infinite set of hierarchically structured expressions by recursively combining discrete elements' (Watumull, 2012, p. 224). The study of the computational mechanisms of language is thus the search for an effective procedure which strongly generates language[17]. This requires a consideration of the 'space' and 'time complexity' of the algorithms involved (see Mobbs, 2015).

2.4 The mathematical hypotheses and the linguistic Turing machine

There are a number of hypotheses central to Watumull (2015), the work which this dissertation extends, and consequently it is worth recapitulating them:

(1) The mathematical hypotheses
(i) The External Reality Hypothesis (ERH)
'There exists an external physical reality completely independent of us humans.' (Tegmark, 2014, p. 254, emphasis original)

(ii) The Mathematical Universe Hypothesis (MUH)
'Our external physical reality is a mathematical structure.' (Tegmark, 2014, p. 254, emphasis original)

(iii) The Mathematical Language Hypothesis (MLH)
'FL is a mathematical structure' (Watumull, 2015, p. 2).

This strong form of the MLH is in some sense not essential for much of the discussion to hold — Tegmark's structural realism (see Cohen (2008) for a philosophical approach) constitutes a somewhat extreme position; however, it can be used to justify a plausible biolinguistic ontology (Watumull, 2013; Watumull, 2015) contra Postal (2009). Furthermore, as Tegmark (2014) details, his MUH is built upon an empirical as well as a philosophical basis and is as a result entirely falsifiable. Even if this were to occur, it would not invalidate the MLH or findings contingent on the MLH, thanks to the implicational logic of the situation. Just like the MUH, the MLH has an empirical foundation — namely that of the findings of generative grammar (see Chomsky et al., 2019) — and if nothing else, it is a useful model. In other words, the mathematical baby does not need to be thrown out with the bathwater — rather, the bathwater could even be kept and sold on for profit. Nevertheless, the hypotheses in Example 4 will be assumed in their full force in the present context, in line with Watumull (2015).

[17] Contrasting with 'weak generativity', which only generates strings, rather than 'hierarchically structured expressions' (see Watumull, 2012).

The MLH needs to be elaborated further — namely by defining the mathematical structure in question. Watumull (2012; 2015) defines this as the 'linguistic Turing machine' (LTM). The LTM is defined as such[18]:

(1) The linguistic Turing machine LTM is the quintuple (Q, Γ, δ, $\#S$, $\#H$) Q: states
Γ: symbols (not including the blank symbol, see Sipser, 2012, p. 168)
δ: transition function, from one state/symbol pair to another
$\#_S$: start symbol
$\#_H$: halt symbol[19]

The LTM encodes the computable function f_{MERGE}, defined as in Example 6[20]: (6) $f_{MERGE}(X, Y) = X, Y$ for $X, Y \in \Gamma$.

The set of symbols Γ is the set of syntactic objects (SOs), the union of the urelements (LIs) with the result of the recursive application of f_{MERGE} on the LIs. The idea of 'recursive application' is crucial and is facilitated in the LTM by the 'tape', the persistent memory of the Turing machine. The output of each step in the computation is stored on the tape, which serves as the input to the next step. Further consequences of this are discussed in Section 4.

The Turing Program for Linguistic Theory (TPLT) (Watumull, 2012) is a label for the investigation of LTM and thus constitutes a subset of mathematical biolinguistics. In particular, the goal is to demonstrate that '[t]he mathematical universality of the Turing machine — the abstract system of computation — implies that it is not only relevant to biolinguistic research, but intrinsic to linguistic — indeed any cognitive-neurobiological — computation' (Watumull, 2012, p. 222, emphasis original). This is self-evident under a Gallistelian approach, thus understanding exactly what kind of computation is needed for language is critical to the programme. Section 4 is thoroughly grounded in TPLT, as a reconceptualization of Watumull's (2015) LTM using features, rather than LIs, as primitives. The majority of Watumull's proofs hold despite this adjustment, as they are not contingent on the *urelements*, the indivisible constituents of sets. There is not room for a thorough mathematical breakdown of this; however, some theoretical consequences of this proposal are discussed in Section 4. For now, the point is that this objective clearly aligns with the central challenge proposed by the TPLT: '[t]o precisify (formalise) the definitions of linguistic primitives in order that 'linking hypotheses' (not mere correlations) to as yet undiscovered neurobiological primitives can be formed' (Watumull, 2012, p. 227). These 'as yet undiscovered neurobiological primitives' align with the Gallistelian search for the neural mechanisms of computation, which, as Gallistel and King (2010) speculate, may be sub-neuronal. Crucially, under the MLH and within the TPLT, the precise formulation of the LTM is not 'play[ing] mathematical games', but rather 'describ[ing] reality' (Chomsky, 1975, p. 81) 'the theory needs to be mathematical because the phenomenon is mathematical' (Watumull, 2012, p. 229).

Finally, note that the treatment of FL as a mathematical structure is implicit in, or at least compatible with, much of the work that falls under the umbrella of biolinguistics. The search is for a biologically implement(ed/able) generative grammar, an invertible function which maps between PHON and SEM. The

[18] Equivalent formulations are possible. This particular formulation has been adapted from Watumull (2012) for expository purposes.
[19] See Watumull (2015) for proofs and derivations using his formulation of the LTM.
[20] A terminological note is in order: I use f_{MERGE} specifically to refer to the mathematical function as defined in Example 6, or as defined in Section 4 below. The term 'Merge' will be used to describe the binary set-forming operation more generally, for instance as considered in the wider Minimalist literature.

function must be biologically implementable because it is evidently biologically implemented, as decades of empirical evidence provided in support of generative grammar have more or less proven. This provides us with necessary constraints on what this system can be like—which can be derived from the essential properties of a biological system: learnability and evolvability. These two parallel constraints find a special case in language but are considered essential throughout the evo-devo programme. Central to mathematical biolinguistics and the TPLT is the Gallistelian notion that the brain is a 'biological computer', subject to the same universals of computation uncovered by Turing. While the specifics of language must be constrained biologically, notably at the interfaces, its core inherits the power and constraints of the computational system underlying it — the LTM. The architecture that enables this is the subject of Section 3.

2.5 Ontology

In exposition of the ideas detailed in the preceding SubSections, I turn now briefly to a criticism of mathematical biolinguistics levelled by Behme (2015), which does not yet seem to have been directly addressed in the public domain[21]. Behme's article constitutes a response to Watumull (2013), which in turn is a response to Postal's (2009) critique of Chomsky's biolinguistic ontology — I concentrate here on the objection to Watumull, leaving the broader attack on Chomsky aside for lack of space (cf. Martins & Boeckx, 2016)[22]. Fundamentally, Behme's criticism is based upon a misunderstanding of the essence of Watumull's argument: Behme repeatedly makes reference to the idea that physical objects cannot have abstract properties, but this is incoherent under Watumull's assumption of mathematical realism and the MUH. Behme and Postal are unable to reconcile the abstract and the physical because they overlook the significance of computability theory (Turing, 1936): the infinite can be compressed into the finite, in the form of an effective procedure. To recite the quintessential example given by Turing (1936), one can design, and physically implement, a procedure that can (given infinite time and memory) compute the irrational (infinite) number pi. Pi has been compressed into a finitary procedure, one which can be physically implemented in a finite machine. In the exact same way, the LTM compresses the infinity of language into a finite, physically implementable machine. The infinite can be represented physically. To force the point: one can imagine a very simple program which writes the number '1' indefinitely[23]. Voila, infinity captured in a finite procedure, nullifying 'Behme's repeated denial of d-infinity, asserting that '[t]he knowledge [of language — LVS] of any speaker is finite because their brain and their lifetime are finite' (Behme, 2015, p. 39). Behme and Postal are implicitly adhering themselves to the connectionist dogma, that functions must be implemented using a lookup table architecture — the architecture adopted by neural network models. The assumption that neural networks accurately model what is happening in the brain leads to the conclusion that effective procedures, i.e., Turing machines, cannot be physically realised in the brain. What Behme and Postal do not appear to realise is that this approach is inherently anti-representational, as

[21] Martins and Boeckx (2016) briefly mention the article as part of a wider point.

[22] The criticism of Behme here is not to be taken as a glorification of Watumull's article, which is not infallible. For instance, Watumull's persistent use of short, even single-word, inline quotes taken out of context is not, as Behme notes, conducive to his argument. Further, Behme and Postal's overall critique of Chomsky's biolinguistics is provocative even in spite of their repeated misapprehensions of the programme, but a full discussion thereof lies beyond the scope of this dissertation (cf. Martins & Boeckx, 2016). For the record, I believe that there is considerably more scope for agreement between the two sides than is apparent from the rhetoric.

[23] i.e., with the pseudocode 'print '1' while true'. Another simple function which could be implemented in an effective (finite) procedure is the successor function *succ*(*x*) (cf. Roberts et al., in press).

Gallistel and King (2010) elaborate, since there is no way for the network to read its current state, without at the same time changing said state. In other words, only a god outside the machine can read off values from a neural network, making these representations meaningless within the network, and by extension within the brain[24]. In sum, '[c]onnectionists draw their computational conclusions from architectural commitments, whereas computationalists draw their architectural conclusions from their computational commitments' (Gallistel & King, 2010, p. ix). To be clear, Behme and Postal do not explicitly represent a connectionist position; however, Behme's implicit denial of the neurobiological relevance computational principles lying behind Watumull's (2013) mathematical biolinguistic ontology on the basis of their 'lack[ing] any biological content' (Behme, 2015, p. 37) is reminiscent of 'draw[ing] computational conclusions from [] architectural commitments'[25].

Further to this, Behme appears to mistake Watumull's claim of isomorphism between the physically encoded symbols in the brain and the symbols of mathematics to mean that the brain must interact with abstract Platonic forms in a kind of mangled Cartesian dualism. The rebuttal to this again follows from a point which is almost obvious from the perspective of computational cognitive science, assumed in the Gallistelian paradigm in which Watumull is working: isomorphism between representations does not mean that they are identical (see Gallistel, 2001). Rather, it is a defining property of representations that they are isomorphic to what they represent — otherwise they would be of no use at all. To analogise to an example of Gallistel and King (2010), if direction vectors were represented in the mind of an ant in a form such that the only operation available to them (say, +) merged the two vectors into a set (say, the form given to syntactic objects), the representation would be completely useless since it has no functional basis in the ant's representation of the world. In the context of language, Watumull's claim that the representations in his LTM are isomorphic to the representations encoded by the brain is an empirical hypothesis. Indeed, Behme (2015, p. 37) seems to overlook this important point, dismissing Watumull's perfectly acceptable claim (in another article, though repeatedly quoted by Behme) that 'as yet undiscovered neurobiological primitives' (Watumull, 2012, p. 227) are the basis for the kind of linguistic computation Watumull espouses. This is certainly in the spirit of Poeppel and Embick (2005) in that it sets the stage for potential neurobiological study, which may reveal the kind of sub-neural computation suggested by Gallistel and King (2010), to which Watumull is clearly implicitly alluding. As such, this contributes to a quintessential aspect of the biolinguistic enterprise, that of taking Poeppel's concerns regarding the potential for interdisciplinary work between linguistics and neurobiology seriously (Gallistel & King, 2010; cf. Boeckx, 2013; Martins & Boeckx, 2016).

Behme also appears to have the causal link between the investigation of biology and linguistics within this research programme backwards. The plausibility of the 'abstract' approach to language as a mathematical object derives from the theory surrounding the Minimalist Program. In short, in order for language to be evolvable and acquirable, it seems to be a reasonable starting point for a hypothesis to say that language is an 'optimal' solution[26]. What this hypothesis comes to mean under the Watumullian program is that language is isomorphic to the LTM. Correspondingly, as in the Minimalist Program generally, a large amount of effort goes towards understanding the 'interface systems', where 'imperfections' are hypothesised to arise. Biological constraints naturally arise—arguably nowhere is this more prominent than in the phonological and phonetic domains, where physical constraints like the need to

[24] The most commonly used mechanism to update the weightings of neurons in a neural network, back-propagation, relies on a god outside the machine in order to read and adjust these values, a matter reluctantly accepted by neuroscientists but one which makes their models ontologically incoherent (Gallistel & King, 2010).
[25] Note that 'architecture' in this quote is being used in a sense entirely separate from the topic of Section 3.
[26] Precisely what 'optimality' means here is a major theme of Section 4; cf. also Section 2.2 and Mobbs (2015).

externalise language in a chrono-linear fashion cause the output of the LTM to be manipulated in increasingly well-understood ways (see Section 3; as well as Kayne, 1994; Uriagereka, 1999; Chomsky, 2008). Pace Behme's interpretation, then, the goal is not to fine-tune the theory to be as bio logically appropriate as possible, but rather to assume that language is 'perfect', and then to discover the biological constraints that impose limits on the output of a physical implementation of such a system, and to determine whether these constraints can reliably be considered part of the 'interface' or whether they require high-level adjustment to the LTM. To this end, I submit that a proper hypothesis regarding the architecture of the linguistic system is needed, in order to clarify the relations between the LTM and other linguistic and non-linguistic cognitive domains. This is the topic of the following Section. Such a hypothesis allows inferences to the structure of the high-level structure of the LTM to be made, namely in the elaboration of what exactly 'optimal' means. Section 4 takes this a step further, suggesting that inferences can be made towards a refinement of the structure of the LTM via application of the minimax theorem upon this architectural basis[27].

3 Architecture

3.1 The lexicon and projection

The traditional generative view of the architecture of the grammar is captured by the 'feed-forward Y-model', as depicted in Figure 1.

As described in Section 1, the lexicon in this model is a list of exceptions composed of bundles of 'features', which are circularly defined as the primitives which make up lexical items. I shall argue that this identification of features proves to be wholly vague and reductionist, and as a result unfalsifiable and thus untenable as a theory. More acutely, it is difficult to find a coherent theory of features throughout the Chomskyan programme — often they are taken as axiomatic, leaving their ontology in the dark, a problem which is widely acknowledged in the literature (Adger & Svenonius, 2011). Furthermore, the definition of features given by Chomsky (1995b) can be argued to violate strict modularity, particularly in the case of 'interpretability' of features, as argued by Zeijlstra (2014). This approach implicitly assumes that computational symbols (interpretable features) must be shared between computational modules, when in fact the exact opposite is the case, as made explicit in the condition of domain-specificity. Even if phonological, grammatical and semantic features are clearly demarcated sets, strict modularity entails that they are completely illegible to the modules they do not belong to, unless they are translated at an interface. In the architecture adopted here, features which appear to be 'interpretable' are merely those features which are translated in a certain special way at the interface, but whose meaning is not dictated by the syntactic feature itself. This also gives room for manipulation of how features are interpreted to take part in the acquisition process — since substance-free features are underspecified in UG. Evidently, the issue of finding an appropriate definition of 'features' is inherently intertwined with architectural considerations — an issue already clear from the phonological literature (see Hale & Reiss, 2008).

[27] Indeed, Section 4 goes some way to addressing Braine's (1992, p. 79) related point of there not being a clear theory of the syntactic primitives which need to be somehow biologically (genetically) encoded, and subsequently employed to 'bootstrap' acquisition (cf. Pinker, 1984)—a citation which Behme (2015) also invokes in her criticism of Watumull (2013).

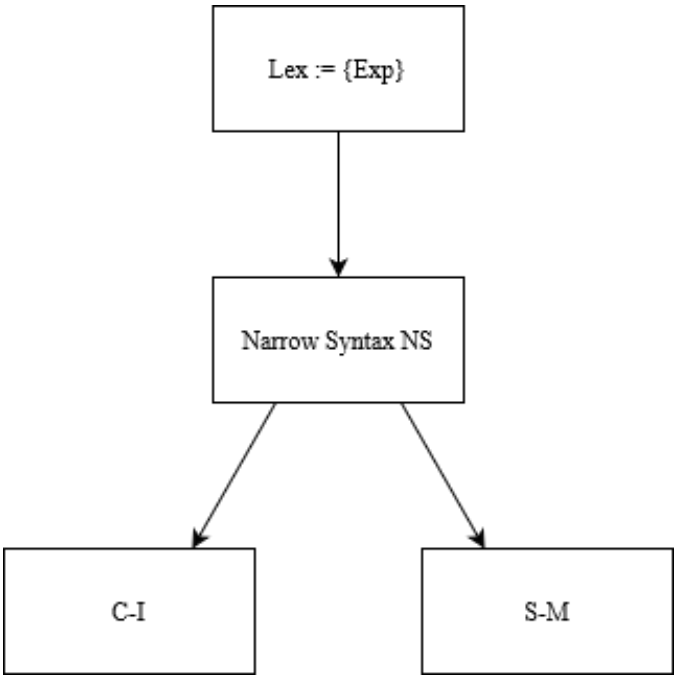

Figure 1: *The Y-model.*

In the Y-model, therefore, the so-called 'projection principle' is foundational. The principle was explicitly formulated in the Government & Binding era, predating modern Minimalism, as 'the idea that all syntactic structure is projected from the lexicon' (Haegeman, 1994, p. xx). In Bare Phrase Structure (Chomsky, 1995a) the principle in its original formulation is incompatible, due to the elimination of X-bar structure; however, Boeckx (2014, p. 6) argues that it persists so long as the lexicon remains the base of the derivation, as in Figure 1 'minimalism has so far failed to distinguish itself from previous transformational accounts, which relied on [the projection principle]'. This is clear from the way Chomsky (1995b) describes the process of lexical 'selection' into the 'numeration', the first step of the derivation. As such, lexical items and their constituent features drive the derivation: the syntax begins with a subset of the lexicon (call this the 'numeration' or 'lexical array' or whatever) and the derivation is constructed from this. Furthermore, the principle persists in conjunction with the Borer-Chomsky Conjecture (BCC) (Baker, 2008, p. 354), such that 'derivations [are] driven by morphological properties [i.e., features — LVS] to which syntactic variation of languages is restricted' (Chomsky, 1993, p. 44). This is the root of what Boeckx (2014) calls 'lexicocentrism'. In the following Sub-Sections, I make the case for eliminating lexicocentrism by eliminating the lexicon as the root of the derivation, in a manner similar to Distributed Morphology (DM) (Marantz, 1997). I integrate this architecture with the representations used in the cartographic, one-feature-per-head approach adopted by Nanosyntax and their interpretation at the interfaces (Baunaz & Lander, 2018; Caha, 2020).

3.2 The semantic spine

When defining an architecture, it is vital to decide on the division of labour between the different modules. Since each module is afforded its own encapsulated computational symbols and procedures, this decision can have empirical consequences. It is interesting, therefore, that in a framework like Nanosyntax, as

Baunaz and Lander (2018, p. 5) openly admit, 'syntax is assumed to be the vehicle for expressing grammatical semantics, and it does so by means of abstract syntactico-semantic features that are arranged by syntax'.

In discussion of 'syntactico-semantic features', it may prove a useful analogy to explore the perpetual focus on features in phonological theory, where defining the inventory and properties of features has been part and parcel of any theory dating back to Jakobson et al. (1928; 2002). It is likely that this is the result of the more apparent 'surfacy' effects of (traditional) phonological features, which have predominantly been linked directly to acoustic and/or articulatory properties, e.g., as in Chomsky and Halle (1968). The same cannot be said of many proposed syntactic features, such as at the extreme the elusive 'EPP-feature' (Chomsky, 2001, p. 4) and its variants, which do not have clear physical correlates. Scheer (2020) refers to this as the degree of 'faithfulness', which makes sense only at the phonology-phonetics interface, but not the syntax phonology one. Nevertheless, the 'substance-free' approach to phonological features entails that this faithfulness is illusory (Hale & Reiss, 2008). The central thesis of this research programme is that phonological features are doubly arbitrary computational symbols with no universal correlates to syntactic or phonetic features beyond language-specific realisation rules. Phonological features are in fact abstract symbols of computation, just as 'formal' syntactic features are[28].

Returning to syntax, cartographic approaches make no attempt to hide the radically sub stance*ful* nature of the functional hierarchy and its features. Blending cartographic insights with a substance-free theory is thus a fairly radical step; however, it is not at all paradoxical in the search for a truly explanatory theory. The directive that syntax and semantics ought to be separated in a modular sense holds, leading to the incentive to separate what is syntactic from what is conceivably some kind of semantic 'spine', corresponding to that of e.g., Cinque (1999), Wiltschko (2014), Ramchand and Svenonius (2014), or some combination.

This leads to the question now of where this 'spine' is stored — is it integrated into the syntactic inventory, as in cartography? Or is it somehow a phenomenon merging from the syntax semantics interface? It is of note here, as noted by Chomsky (2019a) and others, that much of what is typically called 'semantics' in linguistics is actually just a form of syntax. Grammars augmented with rules that are imbued with lambda calculus representations of compositional meaning, are inherently 'syntactic' — technically speaking, a 'semantics' only arises when this representation is linked to the world. This, then, makes sense of the cartographic fusion of 'syntactico-semantic' features which lead to the equation of the syntactic module in Nanosyntax to 'SMS [, which] stands for syntax, morphology, and semantics, which in nanosyntax are seen as one and the same module' (Baunaz & Lander, 2018, p. 11). This is further equivalent to a 'language of thought' as conceived of by Chomsky (2017, p. 200) and the similar idea expressed by Sigurdsson (2020, p. 7). The 'linking to the world', or more precisely to extralinguistic components in the mind/brain which may then transduce their contents into physical phenomena, is performed by C-I, which I have also been referring to as the semantic module.

My inclination is to say that these restrictions on syntactico-semantic structure are indeed interface phenomena. What is deemed a 'feature' in the cartographic sense is (a part of) an entry in the syntactic/semantic-pragmatic interface lookup table. The 'spine' is similarly a 'lexical' concept. Whilst I have thus far assumed for simplicity that all interface translation rules are generated in the process of language acquisition, it is plausible for some to be innate, or further for them to emerge in minimax fashion

[28] Terminologically, I avoid the conventional use of 'formal' to distinguish syntactic/grammatical features, since as symbols they are equally as formal as any other computational symbol. The question remains of what, if anything, substantively distinguishes the different sets of features.

out of constraints imposed within the syntactic or semantic-pragmatic modules themselves. The formalisation of acquisition lies beyond the scope of this work, but it is clearly at least plausible.

3.3 A modular architecture for FL

Each stage in the computation has access to a particular inventory of symbols and an inventory of computations. The module takes a representation formulated in said symbols and from this generates an output determined by the input and the computational inventory, formulated in the same set of symbols.

In between each module must be a translator which converts representations legible in one component to representations legible in another[29]. This is effectively a function between two mutually exclusive sets — compare the function(s) within a module, which are defined by closure over the set of symbols. There are two hypotheses for the nature of this transducer function, known as 'lexical translation and computational translation', as elaborated by Scheer (2020). Lexical translation takes the form of a lookup table — a weak computational architecture, representing the bare minimum needed to represent a function, with maximum specification and no compression. On the other hand, computational translation, as the name implies, applies a computational procedure to the input, transforming it into an entirely different symbolic space. In other words, computational translation is an unhappy hybrid of lexical translation with the fully-fledged computational properties of a module. A computational translator is thus a module with a symbol inventory $U := M_1 \cup M_2$, the union of the inventories of two other modules. This contradicts one of the key motivations for modules in the first place: domain-specificity. If a module can 'understand' two symbolic inventories, there's no need for a 'translation' to take place at all — all computation can take place within the single module, so 'interface devices [are] pointless: if modules can parse the vocabulary of their neighbours, no translation is needed in the first place' (Scheer, 2020, p. 183). Therefore, lexical translation is adopted in this architecture as the model for the interfaces. Furthermore, this corresponds with Boeckx's position (directly following from Marantz, 1997) that the lexicon should not be generative. In line with DM, in which the object that is 'distributed' is the lexicon itself (Marantz, 1997), it seems to be a natural conclusion that the lexicon is in fact an interface phenomenon—in the literal sense that the interfaces constitute the lexicon. Indeed, this is compatible with Scheer's lexical translation, unifying the two approaches. The distributed approach is incompatible with Nanosyntax, however, which avoids lexicocentrism in the sense that it reserves the syntax as the driving force of derivations, rather than projection from the lexicon, but which treats the lexicon as a unified object. Lexical entries in the nanosyntactic lexicon consist of three representations (SO, PF, LF), which can be modelled with the lookup architecture.

In sum, we have modules, and we have interfaces between modules, as depicted in Figure 2[30].

[29] This is in fact distinct from Pylyshyn's (1984) transducer (as adopted by Hale & Reiss, 2008), which converts between symbolic and physical representations, e.g., from phones to sound waves. The phonological system has its own computational module intervening between the output of syntax and SM, and thus its own rules and representations, which may be in the form of a 'phonological Turing machine' (see Vaux & Watumull, 2012).

[30] The increasingly technical specification of modules in the mind should not set off neurological alarm bells— the idea is not phrenological, literally dictating that there is a direct physical correlate of the 'interface-as-list' somewhere in the brain. Rather it constitutes a model for the operations of the mind, with plausible but undefined for each interface (corresponding to DM's Lists 2 & 3).

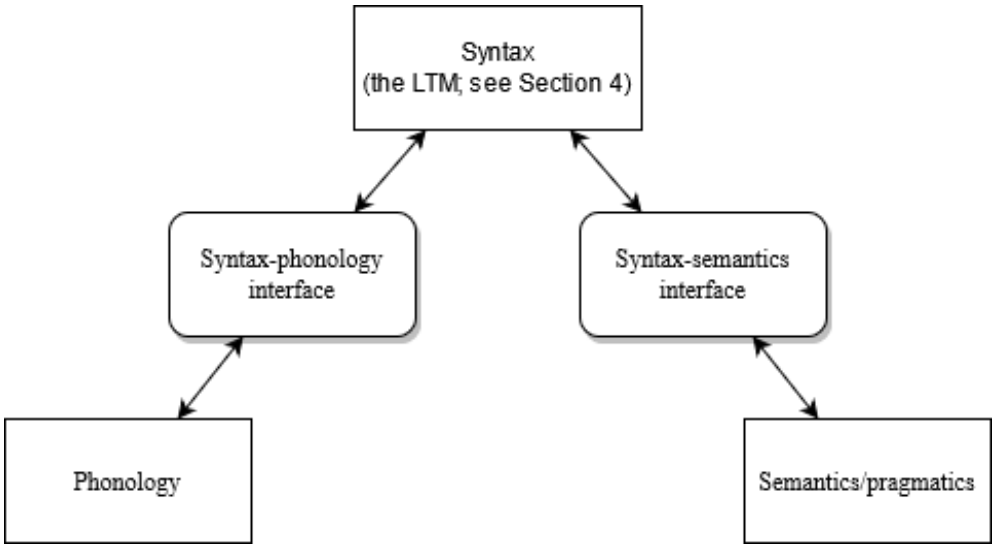

Figure 2: *The architecture.*

The LTM takes the role of Nanosyntax's SMS, incorporating a feature inventory with a Merge operation (the topic of Section 4). A computational module is equivalent to the function $f : X \rightarrow X$; an interface is equivalent to the function $g : X \rightarrow Y$; $X \cup Y = \emptyset$. Different interfaces use different sets of symbols corresponding to X and Y ; however, the general form of the representation is isomorphic between all instantiations (cf. Section 2). In line with the desideratum of mathematical biolinguistics, apparently complex concepts have been reduced to mathematical structures.

This architecture of grammar yields both methodological and conceptual advantages. Considering methodology: due to the flexibility of the interfaces and relative independence of the components, there arises little conflict between individual theories of each of the components, even when radically different philosophical standpoints are adopted. The phonological and semantic components are completely independent of the syntactic component and are permitted to use any form of input as appropriate, assuming that a computable function can be defined to transduce syntactic output into legible phonological/semantic representations — be they the linear representations of Government Phonology (Scheer, 2004) or the arboreal structures of Feature Geometry (McCarthy, 1988; Halle et al., 2000), or similar alternatives on the semantics side. The function g is equivalent to what is usually denoted the 'lexicon', although there is a lexicon neural implementation. The plausibility derives from my adoption of Gallistel and King's (2010) well-argued hypothesis that the brain effectively implements a computational model equivalent to the Turing machine (see Section 2).

With this architecture established, the next task is to establish the inner workings of the LTM, which is the subject of the next Section.

4 Minimax Feature Merge

4.1 Motivations

There are two key motivations for the reformulation of the LTM which is the focus of this Section. The first is deduced from the architecture presented in Section 3, which renders untenable Watumull's (2015) original conception of the LTM, with LIs as the irreducible (from the perspective of the LTM) urelements. In particular, since there is no pre-syntactic lexicon, but rather only a pre-syntactic set of features, the lexical item cannot be a primitive of computation. The procedure encoded by the LTM, f_{MERGE}, must directly manipulate features, or rather constituent structures composed of features — syntactic objects (SOs). Note that even complex SOs are not LIs as defined in Section 1, neither are the translation rules constituting the interface equivalent to LIs; the latter I denote 'lexical entries' to emphasise the distinction.

The second motivation derives from the 'minimax' idea introduced in Section 2.3, 'minimise the maximum loss and maximise the minimum gain'. Relying on LIs as primitives requires considerably more overhead, since LIs take up more space than individual features, and are thus less economical in terms of space complexity (cf. Mobbs, 2015, p. 141). Additionally, LIs are opaque: they are, for the most part, treated by Watumull's (2015) LTM as atomic — the urelements of Merge, which do not need to be 'seen into' for the LTM to function. Indeed, this atomism fails in practice: in the sample derivation provided by Watumull (2015, p. 97) complex featural computation is included in the LTM by way of another (undefined) function, f_{SELECT}, which putatively performs minimal search to detect the LI with an appropriate feature fingerprint, consequently providing it to his f_{MERGE}. The problem with this is immediate from the strictly modular approach: separate operations within the same module appear to be dealing with different computational primitives (symbols) — there is confusion between *syntactic* objects and features, the latter of which are not valid SOs without a parent LI[31]. The 'configuration' is not uniform — rather, two configurations are cascaded: the configuration of LIs and the configuration of features. This problem is entirely eliminated if features are inherently valid SOs, as in the present model, where the problematic intermediary of the LI is eliminated. This is achieved by redefining the LTM to use features, and syntactic objects composed of constituent features by Minimax Feature Merge, as the objects of computation, as is the subject of Section 4.2.

'Minimax Feature Merge' is inductively defined in Section 4.4; however, some preliminary notes are in order. Watumull (2015, p. 27) denotes f_{MERGE} as encoded in the LTM as 'Minimax Merge'. The function is 'minimax' for a conceptually simple, yet vital reason: 'the optimal function for generating syntactic objects is minimax: an 'effectively unary binary function'' Watumull (2015, p. 27, emphasis original). How can a binary function be 'effectively unary'? The answer is natural in the context of the Turing machine, since it corresponds directly to the concept of the 'tape': one of the inputs to the next operation of Minimax Merge is the output of the previous stage in the computation. The tape — the Turing machine's memory — persists between stages in the computation (this is one of the key properties which distinguishes the Turing machine from less powerful automata, see Gallistel & King, 2010). Thus, the first argument in Minimax Merge is always satiated by the current contents of the tape. Minimax Feature Merge inherits this property and is hence a minimax-optimised function, creating the maximal possible output from the minimal possible input. The idea of minimax can be further extended in the present case by looking at features in particular, and how they themselves may be optimised to provide the most information with minimal overhead. This is the topic of Section 4.3.

[31] This is not to say that an LI cannot *contain* a single feature. But if Merge manipulates individual features by incorporating e.g., Chomsky's (2015, p. 240) 'Move F', then, if Merge is also minimax (effectively unary), you arrive at the nanosyntactic view (and the one adopted in the present proposal) of the structure of syntactic representations for free.

It is of further interest to compare this notion of memory to the notion of the workspace (WS), as used by Chomsky et al. (2019). In this configuration, the Merge function is actually ternary (cf. the definition given by Chomsky, 2019a), needing three inputs: the two objects to be merged *and* the workspace as a whole, which may not be the same as the other two SOs. This conception of Merge is thus not minimax in the sense of being effectively unary. It requires far greater complexity, since two objects must be selected to be merged, rather than the single object needed by Minimax (Feature) Merge. Effective unarity also directly relates to the Extension Condition, whereby 'Merge always applies in the simplest possible form: at the root' (Chomsky, 2015, p. 227). This is a given, since one argument of Minimax (Feature) Merge is always satiated by memory. A non-minimax Merge requires the Extension Condition to be a stipulation, rather than something which emerges from the mathematical properties of the function itself. In the present proposal, WS is assumed to be isomorphic to the tape[32].

4.2 Reformulating the LTM

With this preliminary discussion in place, it is possible to provide a formal definition of the featural LTM. As compared to the LTM in Example 5, the featural LTM requires an entirely different vocabulary than that represented by Γ, since Γ contains 'lexical items' which have no status in my theory. Instead, the urelements will be denoted by F, and the vocabulary of the featural LTM by S:

(1) F is the set of syntactic primitives such that $|F| \geq 1$.
(2) The members of F comprise the urelements in S.
(3) S is defined as the union of F with the result of the inductive application of f_{MERGE} to F.
(4) $f_{MERGE} : S \times S \rightarrow S$

f_{MERGE} is a function that takes two members of S as input and returns a member of S as output.

$f_{MERGE}(X, Y) = \{X, Y\}$ for $X, Y \in S$

The featural LTM, as with the LTM in Example 5, is isomorphic to a free magma. A magma is the most basic mathematical structure in universal algebra, consisting of a set S and a binary operation closed under S. A magma is labelled 'free' if no other restrictions are applied (cf. Watumull, 2015, p. 7).

(5) The syntactic component L is isomorphic to the free magma (S, f_{MERGE}), which in turn is isomorphic to the featural LTM.

I return to some interesting consequences of this isomorphism in Section 4.5.

4.3 Features

[32] The question of how adjunction should work in Minimalist Syntax, since it appears to be incompatible with the Extension Condition, is still a topic of debate and there is not room for discussion here.

Despite the pervasive use of the term ‘features’, there remains no generally accepted adequate formal definition thereof. Kibort and Corbett (2010) summarise the issue well:

> ‘Despite their ubiquity and centrality in linguistic description, much remains to be discovered about [features]: there is, for example, no readily available inventory showing which features are found in which of the world’s languages; there is no consensus about how they operate across different components of language; and there is no certainty about how they interact.’ (Kibort & Corbett, 2010, online abstract)

The substance of features in this architecture was briefly discussed in Section 3.2 in relation to the notion of a cartographic ‘spine’. As defined in Section 4.2, however, the featural LTM in principle requires only a single feature to satisfy isomorphism with a free magma. As such, it also does not specify any kind of ‘content’ needed within a feature — to f_{MERGE}, all features are internally uniform (unlike LIs were), although f_{MERGE} is able to distinguish different features externally if $F > 1$. Note that increasing the cardinality of F adds a layer of complexity which takes f_{MERGE} beyond what Chomsky et al. (2019) denote Simplest Merge and note further that this would require corresponding adjustment of the transition function δ of the LTM in order to read and write the new symbol(s) directly.

It is worth then discussing what effects varying the contents of F has on the overall syntax, and the characters of the free magmas it generates. If $F = 1$, then we arrive effectively at Boeckx’s (2014) model, which has only a single feature, denoted the ‘edge feature’ EF. As the only feature, the edge feature is necessarily syntactically inert, serving merely as a remnant of the derivation. This leads to highly uniform trees, which seem perhaps overly simplistic for a language model. Boeckx (2014) resolves this uniformity by appealing to phase theory — the idea of cyclic transfer to the interfaces. If every other phrase is a phase, then phrases can be distinguished, and different properties emerge. Boeckx (2014, p. 44) argues that this gives transitivity: the transitive phase head and the intransitive phasal complement. Syntax is thus ‘regulated’ by the cycle, which prevents unconstrained iteration. A more complex approach is taken by Sigurdsson (2020), who allows for multiple kinds of EFs to be distinguished in the syntax.

The featural LTM with $F = 1$ can be denoted ‘interface-driven’ — in other words, Merge here is free, non-linear, and automatic, as in Boeckx’s (2014) ‘Merge-alpha’ and Chomsky et al.’s (2019) ‘Simplest Merge’. It stands in direct opposition to ‘crash-free’ syntax as perhaps best espoused in the work of Samuel David Epstein and colleagues (Epstein et al., 1998). From the lexicalist perspective, I would argue that this approach makes most theoretical sense, in line with Frampton and Gutmann (2002) and ‘pace’ (Brody, 2002). Any representation constructed by f_{MERGE}, i.e., any member of S, can simply be decomposed into the derivational sequences which constructed the representation. With such a barebones formulation of the function, it is difficult to see how ‘opacity effects’ which obscure the steps of the derivation can be introduced into a representation. This even applies at the interfaces, which need not allow for the ability to take complex, holistic representations as input rather, this data can be ‘serialised’, much as multiple Spell-Out entails in phase theory. Nevertheless, in the mathematical biolinguistic framework, Watumull (2015) argues that much of the derivation-representation issue is moot. The intuitive perspective of computational processes is that they proceed sequentially in time, but ‘the passage of time’ is a purely subjective experience. Objectively, spacetime as a mathematical structure just ‘is’ Tegmark (2014). A computation, like that described by the LTM, is just an alternative description of a mathematical structure. Thus Watumull’s (2015, p. 14) conclusion: ‘[] the derivation/representation debate in syntactic theory is meaningless: subjectively, linguistic expressions are derived in time; objectively, they are simply derived

— represented — in the abstract.' A derivation using the LTM is analogous, indeed, isomorphic, to a proof: the steps taken in the proof are the proof, representationally; analogously, the steps taken in the syntactic derivation using Minimax (Feature) Merge are evident in the final representation. It seems to me, however, that this actually lines up more or less entirely with the position taken by Epstein et al. (1998, p. 12), where 'the post-syntactic computational systems must examine not 'output representations' to determine [syntactic] relations, but, rather, the derivation'. In a free magma, there is by definition nothing more to a representation than the derivation – everything is there, nothing is opaque; the LTM is deterministic.

4.4 f_{MERGE}

Minimax Feature Merge is a recursive function in the sense provided by computability theory as detailed by Watumull et al (2014) and as is essential to its qualification as minimax. A recursive function has three properties: the first is that of Turing computability, the second and third those of induction — definition by induction and mathematical induction, respectively. The fact that f_{MERGE} is Turing computable is evident from its encoding in the featural LTM — namely, it is an effective (finite) procedure, as defined in Section 2.3. Its inductive properties have been implicit in the definition given of the free magma L — the set S is defined by the inductive application of f_{MERGE}. This set, consisting of the syntactic objects, is strongly generated, meaning that a structure given as the argument for f_{MERGE} is one that has already been generated by f_{MERGE}.

The second sense of induction, mathematical induction, extends strong generation to be unbounded – the set S is unbounded. Definition by induction is also a property of the successor function, which derives the natural numbers. Indeed, fMERGE further reduces to the successor function as shown in Section 4.5. Returning to a theme of Section 2: 'finite brains running in finite time literally do generate infinite sets' (Watumull et al., 2014, p. 3, emphasis original).

4.5 Mathematical language

In this final SubSection, I return to the more philosophical matters of Section 2, linking the featural LTM to the desideratum of explanatory adequacy.

The fact that language, assuming my argument to hold, can be 'reduced' to mathematics — that language is isomorphic to a mathematical structure — is not only a matter of beauty, but has concrete consequences for linguistic theory. Mathematics is absolute, so the search for a falsifiable theory of language 'as it is in people's mind/brains' is not only plausible but perhaps inevitable, given enough resources. This is only possible if 'strong generativity' is adopted as the goal, in itself a mathematical notion, associated with the theory of computation and the Turing machine, and thus with the TPLT as presented in Section 2.4. The evidence in favour of the MLH comes from many sources as is characteristic of the Minimalist Program, and indeed biolinguistics generally. Some are in the form of third factors (Chomsky, 2005): for instance, computational efficiency — which crucially has ties to the explanatory requirement of evolvability. As detailed above, f_{MERGE} minimises computational cost and maximises generative capacity, exploiting the nature of the free magma, the simplest mathematical structure but one with the potential for infinite (unbounded) generation, as demonstrated by the isomorphic LTM. There is also empirical evidence in favour of this, adapted from a famous experiment by computer scientist Marvin Minsky and one of his students, Daniel Bobrow. The set of all Turing machines is 'countably infinite' — countable, since they can be listed (unlike, say, the real numbers R), but infinite, since this list would go on forever (cf. Sipser,

2012, p. 206 for a proof). Minsky and Bobrow, in an 'unpublishable' 1961 paper (Minsky, 1967, p. 281) simulated the first few thousand of a subset of Turing machines, finding that, whilst the majority of machines did something entirely uninteresting (e.g., halt immediately, erase their input or get stuck in a loop), the small set which were interesting were all 'essentially the same' and performed a counting operation which 'increased by one the length of a string of symbols' (Minsky, 1985, p. 120). Formally, this is essentially the same as Merge, as Chomsky (2008) claims and as is evident from the definition of the natural numbers by induction using the successor function, shown in Example 12 (from Goldrei, 1996, p. 38):

(6) Given a set x, the successor of x, written as x^+, is the set
$x^+ = x \cup \{x\}$
So
$0 = \emptyset$
$1 = \emptyset^+ = \emptyset \cup \{\emptyset\} = \{\emptyset\}$
$2 = (\emptyset^+)^+ = \{\emptyset\} \cup \{\{\emptyset\}\} = \{\emptyset, \{\emptyset\}\}$
$3 = (\emptyset^{++})^+ = \{\emptyset, \{\emptyset\}\} \cup \{\{\emptyset, \{\emptyset\}\}\} = \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\}$
. . .

Replacing $\cup$ with f_{MERGE}, then, allows the natural numbers to be derived by the LTM and, further, shows that language can be arithmetised. This is shown by Watumull (2015, p. 86) for his version of the LTM; however, the proof using his Minimax Merge is not directly comparable to one using Minimax Feature Merge because of the use of a 'dummy' LI to represent 1. Example 13 shows how the natural numbers can be defined as being derived using a single feature coupled with the empty set:

(13) $0 = \emptyset$
$1 = F_1$
$2 = f_{MERGE}(F_1, \emptyset) = \{F_1, \emptyset\}$
$3 = f_{MERGE}(\{F_1, \emptyset\}, \emptyset) = \{\{F_1, \emptyset\}, \emptyset\}$

Indeed, this formulation arguably makes maximal use of *more minimal* means, by only requiring a single feature, whereas Watumull's (2015, p. 86) appears to require multiple LIs (LI_1, LI_2, . . .).

Notably, this example also shows that a free magma can be generated using only a single feature. This implies that any additional features in F, be their origin in UG (evolution) or in acquisition, are there due to external necessity. Such necessity could be in the form of the semantic spine noted in Section 3.

Returning now to the Minsky & Bobrow experiment: the findings show that the 'useful' areas of computational space consist mostly, perhaps entirely, of functions which reduce to the successor function (and thus to f_{MERGE} as in Example 10). From an *evolutionary* perspective, then, the odds that the useful computation chanced upon by random genetic mutation is f_{MERGE} are relatively high. This affords independent support, then, to the SMT, and the devolution of as much as possible to the interfaces, as described in Section 3.

5 Conclusion

Despite the continual advances of generative grammar over the past more than half a century, 'virtually every aspect of (I-)language remains a problem' (Chomsky et al., 2019, p. 253). This dissertation has attempted to reframe the problem in terms of mathematical biolinguistics, tackling architectural concerns alongside the formulation of syntactic primitives and procedures.

Returning to the title, *{*Minimax *{*Feature Merge*}}* / *{{*Minimax Feature*}* Merge*}* captures the two central aspects of the dissertation: the formulation of features and their role in the architecture of FL (minimax features) and the minimax augmentation of the nanosyntactic disSection of the syntactic component (merging features, not lexical items). The focus has been on Merge in particular, leaving aside other operations like Agree and Label, which may 'fall out' from more general, unavoidable considerations regarding the interactions between cognitive modules[33].

There has not been room to discuss empirical consequences of the theory, and as such this dissertation potentially raises more questions than it answers, as is the nature of such a theoretical work. As I have sought to demonstrate, however, it remains well-founded upon the biolinguistic methodology (Di Sciullo & Boeckx, 2011), embracing interdisciplinarity both within and beyond narrowly syntactic theory.

6 References

Adger, D. & Svenonius, P. (2011). Features in minimalist syntax. In C. Boeckx (Ed.), *The Oxford handbook of linguistic minimalism* (pp. 27–51). Oxford University Press.

Baker, M. C. (2008). The macroparameter in a microparametric world. In T. Biberauer (Ed.), *The limits of syntactic variation* (pp. 351–374). John Benjamins.

Balari, S. & Lorenzo, G. (2012). *Computational phenotypes: Towards an evolutionary develop mental biolinguistics*. Oxford University Press.

Baunaz, L., Haegeman, L., De Clercq, K. & Lander, E. (Eds.). (2018). *Exploring nanosyntax*. Oxford University Press.

Baunaz, L. & Lander, E. (2018). Nanosyntax: The basics. In L. Baunaz, L. Haegeman, K. De Clercq & E. Lander (Eds.), *Exploring nanosyntax* (pp. 3–56). Oxford University Press.

Behme, C. (2015). Is the ontology of biolinguistics coherent? *Language Sciences*, *47*, 32–42.

Boeckx, C. (2013). Biolinguistics: Facts, fiction, and forecast. *Biolinguistics*, *7*, 316–328.

Boeckx, C. (2014). *Elementary syntactic structures: Prospects of a feature-free syntax*. Cam bridge University Press.

Braine, M. D. S. (1992). What sort of innate structure is needed to "bootstrap" into syntax? *Cognition*, *45*(1), 77–100.

Brody, M. (2002). On the status of representations and derivations. In S. D. Epstein & T. D. Seely (Eds.), *Derivation and Explanation in the Minimalist Program* (pp. 19–41). Blackwell.

Caha, P. (2020). *Nanosyntax: Some key features*.

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

Chomsky, N. (1993). A minimalist program for linguistic theory. In K. Hale & S. J. Keyser (Eds.), *The view from building 20: Essays in linguistics in honor of Sylvain Bromberger* (pp. 1–52). MIT Press.

[33] per Watumull (2015, f.n. 117): 'I set aside, or rather subsume within the concept of f_{MERGE}, other functions that enter into the computation to define relations. One such function is Agree . . . I need not posit it 'in addition to Merge' [(Chomsky, 2004, p. 555)] if I can assume the possibility that f_{MERGE} somehow subsumes Agree'.

Chomsky, N. (1995a). Bare Phrase Structure. In H. Campos & P. M. Kempchinsky (Eds.), *Evolution and revolution in linguistic theory: Studies in honor of Carlos P. Otero* (pp. 51– 109). Georgetown University Press.

Chomsky, N. (1995b). *The Minimalist Program*. MIT Press.

Chomsky, N. (2000). Minimalist inquiries: The framework. In R. Martin, D. Michaels & J. Uriagereka (Eds.), *Step by step: Essays on minimalist syntax in honor of Howard Lasnik* (pp. 89–156). MIT Press.

Chomsky, N. (2001). Derivation by phase. In M. J. Kenstowicz (Ed.), *Ken Hale: A life in language* (pp. 1–52). MIT Press.

Chomsky, N. (2004). Beyond explanatory adequacy. In A. Belletti (Ed.), *Structures and beyond* (pp. 104–131). Oxford University Press.

Chomsky, N. (2005). Three factors in language design. *Linguistic Inquiry*, *36*(1), 1–22. Chomsky, N. (2008). On phases. In R. Freidin, C. P. Otero & M. L. Zubizarrete (Eds.), *Founda tional issues in linguistic theory: Essays in honor of Jean-Roger Vergnaud*. MIT Press.

Chomsky, N. (2015). *The Minimalist Program* (20th Anniversary Edition). MIT Press.

Chomsky, N. (2017). The language capacity: Architecture and evolution. *Psychonomic Bulletin & Review*, *24*(1), 200–203.

Chomsky, N. (2019a). *Fundamental issues in linguistics* (Lecture series). MIT. Chomsky, N. (2019b). *The UCLA lectures* (Presentation recording).

Chomsky, N., Gallego, A. J. & Ott, D. (2019). Generative grammar and the faculty of language: ´ Insights, questions, and challenges. *Catalan Journal of Linguistics*, 229–261.

Chomsky, N. & Halle, M. (1968). *The sound pattern of English*. MIT Press. Cinque, G. (1999). *Adverbs and functional heads: A cross-linguistic perspective*. Oxford Univer sity Press.

Cohen, M. (2008). *On reality as a mathematical structure* (PhD Dissertation). Ben Gurion University of the Negev. Israel.

Deutsch, D. (2011). *The beginning of infinity: Explanations that transform the world*. Allen Lane.

Di Sciullo, A.-M. & Boeckx, C. (2011). Introduction: Contours of the biolinguistic research agenda. In C. Boeckx & A.-M. Di Sciullo (Eds.), *The biolinguistic enterprise: New perspectives on the evolution and nature of the human language faculty* (pp. 1–16). Oxford University Press.

Epstein, S. D., Groat, E. M., Kawashima, R. & Hisatsugu, K. (1998). *A derivational approach to syntactic relations*. Oxford University Press.

Frampton, J. & Gutmann, S. (2002). Crash-proof syntax. In S. D. Epstein & T. D. Seely (Eds.). D. W. Lightfoot (Ed.), *Derivation and explanation in the Minimalist Program* (pp. 90– 105). Blackwell.

Gallistel, C. R. (2001). Mental representations, Psychology of. In N. J. Smelser & P. B. Baltes (Eds.), *International Encyclopedia of the Social & Behavioral Sciences* (pp. 9691–9695). Pergamon.

Gallistel, C. R. & King, A. P. (2010). *Memory and the computational brain: Why cognitive science will transform neuroscience*. Wiley-Blackwell.

Goldrei, D. C. (1996). *Classic set theory: For guided independent study*. Chapman & Hall. Haegeman, L. (1994). *Introduction to Government and Binding Theory* (2nd ed.). Blackwell.

Hale, M. & Reiss, C. (2008). *The phonological enterprise*. Oxford University Press.

Halle, M., Vaux, B. & Wolfe, A. (2000). On feature spreading and the representation of place of articulation. *Linguistic Inquiry*, *31*(3), 387–444.

Jakobson, R., Karcevsky, S. & Trubetzkoy, N. (2002). Quelles sont les methodes les mieux ´ appropriees ´ a un expos ` e complet et pratique de la phonologie d'une langue quelconque? ´ *Selected writings: Vol. 1. phonological studies* (3rd ed., pp. 3–6). Mouton De Gruyter. (Original work published 1928)

Kayne, R. S. (1994). *The antisymmetry of syntax*. MIT Press.

Kibort, A. & Corbett, G. G. (Eds.). (2010). *Features: Perspectives on a key notion in linguistics*. Oxford University Press.

Lenneberg, E. H. (1967). *Biological foundations of language*. John Wiley & Sons. Marantz, A. (1997). No escape from syntax: Don't try morphological analysis in the privacy of your own lexicon. *University of Pennsylvania working papers in linguistics*, *4*(2), 14.

Martins, P. T. & Boeckx, C. (2016). What we talk about when we talk about biolinguistics. *Linguistics Vanguard*, *2*(1), 1–15.

McCarthy, J. J. (1988). Feature geometry and dependency: A review. *Phonetica*, *45*(2–4), 84–108. Minsky, M. L. (1967). *Computation: Finite and infinite machines*. Prentice-Hall.

Mobbs, I. (2015). *Minimalism and the design of the language faculty* (PhD Dissertation). Univer sity of Cambridge. Cambridge.

Pinker, S. (1984). *Language learnability and language development*. Harvard University Press. 27

Poeppel, D. & Embick, D. (2005). Defining the relation between linguistics and neuroscience. In A. Cutler (Ed.), *Twenty-First century psycholinguistics: Four cornerstones* (pp. 103–118). Taylor & Francis Group.

Postal, P. M. (2009). The incoherence of Chomsky's 'biolinguistic' ontology. *Biolinguistics*, *3*(1), 104–123.

Pylyshyn, Z. W. (1984). *Computation and cognition: Toward a foundation for cognitive science*. MIT Press.

Ramchand, G. & Svenonius, P. (2014). Deriving the functional hierarchy. *Language Sciences*, *46*, 152–174.

Roberts, I. & Watumull, J. (2015). Leibnizian linguistics. In A. J. Gallego & D. Ott (Eds.), ´ *50 years later: Reflections on Chomsky's Aspects*. MIT Working Papers in Linguistics.

Roberts, I., Watumull, J. & Chomsky, N. (in press). Universal Grammar. In D. Vakoch (Ed.), *Xenolinguistics: Toward a science of extraterrestrial language*. Oxford University Press. Scheer, T. (2004). *A lateral theory of phonology: Vol. 1. what is CVCV and why should it be?* De Gruyter Mouton.

Scheer, T. (2020). On the lexical character of intermodular communication. *Radical: A Journal of Phonology*, *1*, 183–239.

Sigurdsson, H. A. (2020). Universality and variation in language: The fundamental issues. ´ *Evolutionary Linguistic Theory*, *2*(1), 5–29.

Sipser, M. (2012). *Introduction to the theory of computation* (3rd ed.). Cengage Learning. Tegmark, M. (2014). *Our mathematical universe: My quest for the ultimate nature of reality*. Penguin Books.

Tomalin, M. (2006). *Linguistics and the formal sciences: The origins of generative grammar*. Cambridge University Press.

Turing, A. M. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, *2nd series 42*(1), 230–265.

Uriagereka, J. (1999). Multiple spellout. In S. D. Epstein & N. Hornstein (Eds.), *Working minimalism* (pp. 251–282). MIT Press.

Vaux, B. & Watumull, J. (2012). *The phonological Turing machine*.

von Neumann, J. (1928). Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, *100*(1), 295–320.

Watumull, J. (2012). A Turing program for linguistic theory. *Biolinguistics*, *6*(2), 222–245. Watumull, J. (2013). Biolinguistics and Platonism: Contradictory or consilient? *Biolinguistics*, *7*, 301–315.

Watumull, J. (2015). *The linguistic Turing Machine* (PhD Dissertation). University of Cambridge. Cambridge.

Watumull, J. & Chomsky, N. (2020). Rethinking universality. In A. Bar´ any, T. Biberauer, J. ´ Douglas & S. Vikner (Eds.), *Syntactic architecture and its consequences II: Between syntax and morphology* (pp. 3–24). Language Science Press.

Watumull, J., Hauser, M. D., Roberts, I. G. & Hornstein, N. (2014). On recursion. *Frontiers in Psychology*, *4*.

Wiltschko, M. (2014). *The universal structure of categories: Towards a formal typology*. Cam bridge University Press.

Zeijlstra, H. (2014). On the uninterpretabilitty of interpretable features. In P. Kosta, S. L. Franks, T. Radeva-Bork & L. Schurcks (Eds.), *Minimalism and beyond: Radicalizing the interfaces* (pp. 109–129). John Benjamins.