COPiL CAMBRIDGE OCCASIONAL PAPERS in LINGUISTICS
Volume **6**, Article 11: 311–329, 2012 | ISSN 2050-5949

THE COMPUTABILITY AND COMPUTATIONAL COMPLEXITY OF GENERATIVITY*

JEFFREY WATUMULL
University of Cambridge & MIT

ABSTRACT Any theory of linguistic cognition is a theory of linguistic computation and thus must posit a generative function. The common assumption is that this function is n-ary. I argue that this assumption cannot be maintained if the theories of computability and computational complexity are accepted: an n-ary form of the function *Merge* is either incomputable or intractable. By contrast, a form of binary Merge is computable, tractable, and computationally optimal. From the assumption of binarity, predictions are generated as to the universality of binary branching structures and ergo the impossibility of flat-structures (and non-configurational languages).

1 INTRODUCTORY REMARKS

Cognition is demonstrably computational: simply and informally stated, procedures are run to determine the outputs of functions given inputs. In the domain of linguistic cognition, procedures are run to generate syntactic structures, inter alia, and thus an explicit theory of this generative process is a central desideratum of linguistic inquiry.

To define the function generative of syntactic structures as binary is generally believed to be stipulative (see, e.g., Yang 1999); that it is simpler to assume the function intrinsically n-ary but restricted to binarity in the majority of its applications by extrinsic factors such as computational resources (e.g., minimal search), conditions at the interfaces with extralinguistic systems (e.g., predicate-argument structure at the conceptual-intentional interface, linearization at the sensory-motor interface), the imperative for "unambiguous paths" (Kayne 1981) in phrase structure, etc. (see Chomsky 2008). In this squib I shall challenge this assumption, arguing that an n-ary function is at worst intractable—or worse still, even incomputable—and at best intolerably inefficient. It will follow from the conditions n-arity fails to satisfy

* For insightful comments, my many thanks go to Scott Aaronson, Bob Berwick, Noam Chomsky, Randy Gallistel, Marc Hauser, and Ian Roberts.

that *some form* of binarity is computable, tractable, and optimally[1] efficient. My theory of binarity—only briefly adumbrated here (see Watumull 2010, 2012b for its elaboration)—denies the possibility of so-called flat syntactic structures, rejects traditional analyses (some already moribund) of so-called non-configurational languages, and predicts the universality of so-called binary branching structures; I can thus contribute to resurgent and successful research on language universals (see, e.g., Biberauer, Holmberg, and Roberts 2011, Roberts 2011). The particular formulation of the generative function I investigate is called *Merge*; this function is a particularly serviceable lab rat, having been explicitly formulated and extensively discussed (see, e.g., Watumull 2010 and the numerous references therein). N.B., mine is *a theory of binarity*, formulated so as to be generalizable from Merge and the principles of the Minimalist Program (Chomsky 1995, et seq.) to any generative process posited in any research program (see sections 5 and 6); and such a process must be posited in some form(s) in any theory of the human competence to compute syntactic structures.

Finally, an apology and an apologia from the armchair: I shall not dilate on data; my intent rather is to *prepare the ground*, as it were, for empirical inquiry.[2] I contend that such a priori formalization can be constructive for approaching data by defining those properties that must obtain of the theory by virtual logical necessity[3] and by exposing inherently (technically) impossi-

1 The concept of optimality assumed here is that implied by the *minimax* theorem (von Neumann 1928) as formulated in Watumull 2010: informally stated, minimize input representations/resources, maximize output expressions.

2 I am advocating an explicitly theory-laden approach to data. Any approach of any interest is (implicitly) theory-laden: a datum is informative only insofar as it relates to (corroborates, confutes, etc.) prior assumptions (as demonstrated formally in Shannon 1948). This theory of information is a philosophy of science: "If a term 'F' is to be a meaningful observation term, then its predication in 'Fa' must have some material *consequences* [...]. The sentence 'Fa' will clearly have this property if it is asserted in a context where general sentences such as '$(x)(Fx \supset Gx)$', '$(x)((Fx \& Hx) \supset \sim Kx)$', [etc., are] assumed. 'Fa' will then imply 'Ga', be incompatible with '$(Ha \& Ka)$', [etc.]. But if 'F' figures in no such background beliefs or assumptions [...], then 'Fa' will be entirely without consequence or significance [...]. It will have no bridges to link its assertion or denial with the assertion or denial of any other sentence. [I]ts assertion will be *computationally inert*. It will be without computational significance for the very cognitive system that asserts it" (Churchland 1988: 183) (emphases original). For syntactic theory, material consequences follow from the positing of a generative procedure if, inter alia, the principles of computability and computational complexity are assumed.

3 Given the evidence for linguistic computation, I argue it is logically necessary to assume the language faculty to be a form of Turing machine with addressable read/write memory (see Watumull 2012a). The general reasoning is as follows (Gallistel and King 2009: 125, 105, i): If "the brain is an organ of computation[,] then to understand the brain one must understand computation [and how it may be physically implemented]," which necessitates

ble or implausible theories. "It is quite incorrect [...] to regard formalization as an activity that occupies the researcher after he has developed an effective theory. On the contrary, formalization can play a very productive role in the process of discovery itself [...]. *We will find that certain familiar conceptions of linguistic structure, when formalized, literally cannot provide [a] grammar, and that others can do so, if at all, only at an intolerably great cost*" (Chomsky 1955: 58) (emphasis added). (Indeed, my argument here is that n-ary Merge is either incomputable (i.e., it "literally cannot provide [a] grammar") or intractable (i.e., if it can, it can "only at an intolerably great cost").) With these boundary conditions established, the domain of admissible—and potentially productive—theories is drawn into relief. Substantively, and most importantly, *the formalization thus restricts the space of possible linguistic structures.*

2 Merge

The type of n-ary function proposed in the literature (see Chomsky 1995, et seq.) is a set-formation function that accepts n arguments (syntactic objects SOs, simple or complex) to generate as a value the set (a new complex SO) containing the n elements, as in (1). Binary Merge thus functions as in (2).[4]

(1) $f_{n\text{-aryMERGE}}(\delta_{\mathrm{i}}, ..., \delta_{\mathrm{n}}) = \{\delta_{\mathrm{i}}, ..., \delta_{\mathrm{n}}\}$

(2) $f_{2\text{-aryMERGE}}(\alpha, \beta) = \{\alpha, \beta\}$

My argument is that for a subset of cases, (2) is the only form of Merge technically possible and plausible—computable and tractable, respectively—and for all cases the only form worth wanting, assuming the thesis that linguistic computation is optimal; a reasonably defined notion of optimality is

formalization; "[Turing] created a formalization that defined a class of machines," with *functional* components and procedures so elementary as to be multiply physically realized. And "By mathematically specifying the nature of these machines, and demonstrating their far-reaching capabilities, [Turing] laid a rigorous foundation for our understanding of what it means to say something is computable." A formalization can thus define conditions of adequacy that any theory in a particular domain of inquiry must satisfy to be true. Thus if it is demonstrated by research on higher levels of analysis that "brains are powerful organs of computation," and that a formally definable "[addressable read/write] memory mechanism is indispensable in powerful computing devices," then it is incumbent upon researchers at lower levels of analysis to demonstrate how such a mechanism is biologically implemented (see Marr 1982 on levels of analysis).

4 The classical rules of Generative Grammar are compressible into binary Merge: phrase structure rules into external Merge EM ($EM(X,Y|X\notin Y \wedge Y\notin X) = \{X,Y\}$); transformation (movement) rules into internal Merge IM ($IM(X,Y)|X\in Y \underline{\vee} Y\in X) = \{X,Y\}$).

consistent with metaphysical and methodological parsimony (and thus a stimulant to research if nothing else).[5]

3 Computability

Merge of arity n is two-ways incomputable.

3.1 Nonfinite Procedure

Implemented in a Turing architecture (Turing 1936),[6] the input to the Merge procedure, implementing the set-formation function, is a potentially infinite tape (the analogue to a set of SOs to be merged) divided into finite discrete symbols (each symbol representing an SO). The arity of the function is defined by the number of SOs necessary to saturate it: 2 for a binary function, n for an n-ary function. Once the function is saturated, the procedure halts and generates an output (a complex SO); this process can be iterated indefinitely given that the tape can be infinite.

For n-ary Merge, the set of possible values of n—equivalent to the set of possible procedures for computing an output—is infinite and the input k can be infinite (e.g., were SOs to be continually selected and merged from the lexicon and/or parallel derivations). If $n = k$, $k = \infty$, then n-ary Merge cannot be finitely defined—assuming as I am, in this instance, that the function is *not* decomposed into a sequence of finitely defined operations—and thus the procedure cannot halt to produce an output. (Hereinafter, in discussing computability, n-ary Merge refers to a function of arity n, n = input k, $k = \infty$.) This is the problem. "Recursive functions [computable functions] have the important property that, for each given set of values of the arguments, the value of the function can be computed by a finite procedure" (Gödel 1934: 348). The finiteness of the procedure (n.b., *not* the input tape[7]) is not only an important property of computability, it is the sine qua non in that any procedure lacking it is ipso facto incomputable: "There must be exact instructions, (i.e., a program), finite in length, for the procedure [to be computable]"

5 See Moro 2008 and Culicover and Jackendoff 2005 for discussions of the numerous notions—methodological and metaphysical—of linguistic "optimality," "simplicity," etc. in the diverse research programs of Generative Grammar.

6 See Watumull 2012a for a formulation of a linguistic Turing machine.

7 "The entire point of computability theory is to be able to talk about inputs of unbounded length. E.g. once you know a multiplication algorithm, you can multiply n-digit numbers for arbitrary n" (Scott Aaronson, personal communication) (emphasis added). The problem for n-ary Merge is that it cannot be known (i.e., defined) for unbounded input if—as I am assuming in this instance—*the length of the definition of Merge is equal to the length of its input*. For multiplication, the function is defined as binary a priori, but even it is incomputable on incomputable numbers (see footnote 8).

(Enderton 1977: 528). The arity of Merge is an instruction as to the number of arguments to merge. These instructions can be specified for n-ary Merge only once the nth element of the input is counted, and thus if the input k is infinite, and $n = k$, then the procedure is uninstructed and cannot compute.[8]

Natural language is demonstrably a system of discrete (digital/denumerable) infinity, so it is a contradiction in terms for it to be incomputable if fed infinite input. By contrast, binary Merge is computable by definition in that it generates an output if and only if fed two SOs (the minimal number of input representations); this finite process can be iterated indefinitely—i.e., over any k—to generate an infinity of discretely structured expressions (the maximal number of *maximally information-bearing* output representations as defined in Watumull 2012b).

N.B., binary Merge is *strongly generative*: it generates hierarchically structured expressions—representing syntactic information—mappable via semantic and phonological-morphological information to the interfaces with conceptual-intentional and sensory-motor systems, respectively. These strongly generated structures correspond to *weakly generated* strings, which are for the most part—if not wholly—immaterial to syntactic and semantic cognition; n-ary Merge can but need not be strongly generative, and thus can but need not generate expressions material to syntactic and semantic cognition.

3.2 Undecidable Arity

Given some input, unbounded or bounded, if the arity of the function is the unspecified n, it cannot ever be decided when, where, or even if the procedure halts (or starts); in other words, given some input k of SOs, if the arity of Merge is undefined, it cannot be decided which if any members of k become members of the SO under construction (which SOs enter the derivation). The arity of n-ary Merge thus needs to be stipulated merger by merger; its formal definition—and, equivalently, its physical representation in the system—needs to be erased and recoded merger by merger, thus complexifying the system grievously. The undecidable problem does not obtain of a function with an

8 For infinite input, it is as if n-ary Merge is processing an incomputable number: "A computable number [is] one for which there is a Turing machine which, given n on its initial tape, terminates with the nth digit of that number [encoded on its tape]" (Minksy 1967: 159). Equivalently, incomputable numbers are numbers that cannot be physically represented. It follows that the majority of the real numbers are incomputable and that the arithmetical functions are computable only for computable numbers. Analogous to an incomputable real number with an infinite decimal expansion is the infinite input k to n-ary Merge, n = k: such a number/input is incomputable, thus "there is no machine that can generate a representation (encoding) of [it]" (Gallistel and King 2009: 52).

arity prespecified for all possible mergers (e.g., binary Merge).

The procedure implementing n-ary Merge is thus not guaranteed to halt, generating an output, and "one cannot make productive use of a function unless one can determine the output for any permissible input" (Gallistel and King 2009: 87).[9]

3.3 Straw Man

One could concede my argument for the incomputability of n-ary Merge, but dismiss it as otiose, the slaying of a straw man: the incomputability of n-ary Merge emerges only in the hypothetical limit—with infinite input k, $n = k$. In the "real world," however, k will always be finite so that even if n is unspecified, n-ary Merge will eventually halt as computable.

To this dismissal I should reply that incomputability, in its abstractness, is a *really* "real" problem if the object of linguistic inquiry is, or can be regarded as, "the thing in itself," i.e., a computational—thus mathematical—system abstracted away from spatiotemporal contingencies, as a Turing machine is with its memory space and operating time unlimited so as to reveal the *functional* components and procedures which exhaust its definition as a Turing machine (n.b., any computational system is a form of Turing machine); in other words, to be a Turing machine is nothing more and nothing less than to be a mathematical object of a particular type.[10] Consider the "superstition" in attaching "Importance [...] to the fact that modern digital computers are electrical, and that the nervous system also is electrical. Since Babbage's machine was not electrical, and since all digital computers are in a sense equivalent, we see that this use of electricity cannot be of theoretical importance [...]. If we wish to find [...] similarities [between digital computers and the nervous system], we should look rather for *mathematical analogies of function*" (Turing 1950: 439) (emphasis added). Consistent with this reasoning, the object of linguistic inquiry can be defined as a functional (mathematical) *competence*: the

9 Evolutionarily, an incomputable system cannot be selected for—as it confers no advantage—or against—as the majority of incomputable functions are probably neither adaptive nor maladaptive, and thus undetectable by selection (see Minksy 1985). (See Rosenberg 2011 for an argument that nature only selects against, which would imply that a neutral incomputable function could evolve. However, the fixation/retention of such a function in the gene pool could be difficult to explain.) It is possible in principle for an incomputable system to evolve as a spandrel, but this would be *selection of* the system, not *selection for/against* it: the incomputable system would constitute a byproduct of some adaptive trait (see Fodor and Piattelli-Palmarini 2010 for an interesting but fallacious analysis of the selection-of/selection-for distinction.)

10 A Turing machine is a mathematical abstraction: the general and necessary and sufficient conditions an object must satisfy for it to be defined as a (type of) Turing machine are purely functional (see Carnap 1955 on such intensional definitions).

"underlying system of rules" (Chomsky 1965: 4) in the mind that "represents the information concerning sentence structure that is available, in principle, to one who has acquired the language" (Chomsky 1963: 326-327). This information is represented as an "idealization [...] leaving out any limitations [...] of memory, time, and access" (Chomsky 1965: 4, 10). Idealization of the linguistic system reveals the components and procedures which exhaust its definition as a subtype of Turing machine; these components and procedures are purely functional—purely and ultimately mathematical. It is thus legitimate to define the linguistic system as nothing more and nothing less than a mathematical object of a particular type.[11] Such idealization is part and parcel of the methodology *and the metaphysics* of normal science, which proceeds by the "making of abstract mathematical models of the to universe to which at least the physicists give *a higher degree of reality* than they accord the ordinary world of sensation" (Weinberg 1976: 28) (emphasis added).[12]

In the "world of sensation," call it "the computable world," things in themselves, often abstract, are confounded by arbitrary constraints, often physical. For computational systems, confounding the abstract with the physical can conflate the obvious yet only lip serviced distinction between software and hardware; thus this important distinction remains unassimilated, thus preventing recognition of the fact that "As our knowledge increases, the abstract mathematical world becomes farther removed from the world of sensation" (Weinberg 1976: 28). For instance:

> You know that if your computer beats you at chess, it is really the program that has beaten you, not the silicon atoms or the computer as such. The abstract program is instantiated physically as a high-level behaviour of vast numbers of atoms, but the *explanation* of why it has beaten you cannot be expressed without also referring to the program in its own right. That program has also been instantiated, unchanged, in a long chain of different physical substrates, including neurons in the brains of the programmers and radio waves when you downloaded the program via wireless networking, and finally as states of long- and short-term memory banks in your computer. The specifics of that chain of instantiations may be relevant to explaining how the program reached you, but it is irrelevant to why it beat you: there, the content of the knowledge (in it, and in you) is the whole story. That story is an explanation that refers ineluctably to abstractions; and

11 It goes without saying that this is but one (objective) way to explain language.

12 This can be construed as a restatement of the Platonic theory of forms.

> therefore those abstractions exist, and really do affect physical objects in the way required by the explanation. (Deutsch 2011: 114-115) (emphases original)[13]

Consistent with this reasoning, it is not unreasonable to "give a higher degree of reality" to an "abstract mathematical model" of linguistic computation—to which my arguments on (in)computability apply—than to the "ordinary world of sensation," governed by computational complexity theory, with its concerns for (in)tractability.

4 Tractability

In mathematical abstraction, n-ary Merge can be incomputable. In concrete (physical) computation, n-ary Merge, $n > 2$, can be intractable because the spatiotemporal resources necessary for a procedure implementing n-arity to generate an output increase exponentially as the number of arguments to merge increases linearly.[14] And obviously space and time are not limitless; n.b., the amount of energy required to recode the function—erase and restipulate the arity n as it varies—merger by merger is so limited by physical law as probably to trivialize the frequency and extent to which n can vary (see Rio et al. 2011 on erasure as the costliest of computational operations). Thus for some (as yet undefined) "large" value of n—and a fortiori infinite n—n-ary Merge is physically infeasible if not impossible.[15]

13 This is to restate the Aristotelean distinction of matter and form.

14 By way of analogy (see Gallistel and King 2009: 7), suppose "we want to construct symbols to represent different durations [...]. The symbols are to be constructed by placing marbles into rows of hemispherical holes on a board. [E]ach row [could] be the symbol for a different duration and increment the number of marbles in a row by one for each additional second of duration symbolized by that row. We call this the *analog principle* [...]." The analog principle describes n-ary Merge: if a row corresponds to the number of argument positions in the function and the marbles correspond to arguments, "We are immediately struck by the discouraging size of the bag of marbles we will need and the length of the board. The problem with this design is that the demand on these physical resources grows in proportion to the number of [symbols] that we want to distinguish."

15 The size of a merger is multiply definable, and on virtually any definition, the size of n-ary Merge ($n > 2$) exceeds that of binary Merge; and inefficiency increases exponentially with increasing n. An analogy is that of the definition of an algorithm for determining a matching in graph theory: "The relative cost [...] of the various applications of a particular algorithm is a fairly clear notion, at least as a natural phenomenon [...]. The domain of applicability for an algorithm often suggests for itself possible measures of size for individual problems—for maximum matching, for example, the number of edges or the number of vertices in the graph [...]. There is an obvious finite algorithm, but that algorithm increases in difficulty exponentially with the size of the graph. It is by no means obvious whether or not there exists an algorithm whose difficulty increases only algebraically with the size

In connection with this absolute limit on physical computation is the exigency for thermodynamic stability. Because n-ary Merge does not halt to generate an output until the nth argument to be merged is retrieved (counted) from the lexicon and/or a parallel derivation, each argument $< n$, as it is retrieved, needs (in some connectionist models) to circulate in a reverberating memory loop, generating heat, waiting until n is reached. The problem is that "A reverberating loop is a highly volatile mechanism; in the absence of the energy required to maintain continuous signal transmission, the signal dissipates in a matter of milliseconds" (Gallistel 2003: 206). Thus for a large number n of arguments to be merged, requiring large amounts of energy, n-ary Merge is condemned by the laws of thermodynamics. Binary Merge, with the smallest number of arguments to be computed, reduces the load on reverberating loops, thus reducing energy and heat to the minimum.

However, Gallistel argues that reverberating loops are (probably) not central to neural computation (precisely because of their volatility).[16] If he is correct, and I am convinced he is, Merge of any arity is predicted to operate with a stable read/write memory.

But even replacing reverberating loops with the non-volatile read/write memory—the tape(s)—of a Turing machine—equivalently, the stack(s) of a pushdown automaton—does not render n-ary Merge computationally tractable. As the input tape bearing SOs is fed into the machine in discrete moves, binary Merge reads an SO off the tape and writes (pushes) it to the stack, thus forming a structured set with the (complex) SO formed on the preceding move; the stack thus represents the SO under construction (the derivation). Operating sequentially, immediately offloading (merging) to the stack each SO it processes, binary Merge does not count, thus the number of input SOs is immaterial to its computation.[17] For n-ary Merge, however, the length of the tape *is* material, as the input SOs must be stored—effectively counted—in the finite state memory until the nth input is read and Merge can write. The state memory is by definition insufficient to compute a lengthy input, let alone an

of the graph [...]. For practical purposes the difference between algebraic and exponential order is often more crucial than the difference between finite and non-finite" (Edmonds 1965: 450-451). Equivalently, a procedure is efficient if it is upper-bounded by a polynomial function of n and inefficient if lower-bounded by an exponential function of n. Binary Merge is (trivially) polynomial; n-ary Merge is exponential. A simpler measure is in terms of Kolmogorov complexity: the length of the shortest description y such that y represents x, x some program(=function) for the (syntactic) computation; by this measure, the complexity the n-ary Merge program/function exceeds that of the binary Merge program/function by definition.

16 Indeed, connectionism generally is probably false (see Watumull 2012b).

17 A formulation of binary Merge in Polish notation could precisify the "immediacy" of its execution (C.R. Gallistel, personal communication).

infinite one (see Chomsky 1955). Thus again is n-ary Merge intractable.

Binary Merge is tractable if any function is: its spatiotemporal resources are held constant at the absolute minimum necessary and sufficient for a function to be nontrivially generative.[18] In this natural sense of minimizing input to maximize output—expressions can be added to expressions indefinitely—binary Merge is the optimal generative function. Unlike n-ary Merge, binary Merge halts as soon as a combinatorial operation can halt.[19] Thus even a tractable n-ary Merge—a function with not "too large" an n, $n > 2$—is suboptimal vis-à-vis binary Merge.

The (potential) intractability of n-ary Merge is related to its (potential) non-compactness. Informally stated, the procedure for some function is compact if the information necessary to encode it is some significant number of orders of magnitude less than the information the procedure can generate. Binary Merge is compact because it codes for two and only two arguments and yet can be iterated to merge n SOs; this is the power of combinatorial syntax.[20] Thus the (irreducibly[21]) finite procedure implementing the binary function can generate an infinite array of information. But n-ary Merge can code for any number of arguments so that in principle the information necessary to encode it can be equal to—and frequently greater than—the information it can generate: i.e., specifying the function requires as many if not more bits than specifying the problem (the set of SOs to be merged); only if n is constrained can the function be compact, and binarity is maximally compact—the form worth wanting. The distinction in compactness between binary Merge and n-ary Merge can be stated (here informally) in terms of the length of the shortest

18 A unary Merge would be trivially generative in that it would not be combinatorial/compositional.

19 Merging/Halting as soon as possible conforms to the logic of the Earliness Principle (Pesetsky 1989). And by halting as early as possible, binary Merge implements a minimal search procedure general to efficient computational systems: "The symbols that carry the two values that serve as the arguments of a two-argument function cannot occupy physically adjacent locations, generally speaking. Thus, the functional architecture of any powerful computing device, including the brain, must make provision for bringing symbols from their different locations to the machinery that effects the primitive two-argument function" (Gallistel and King 2009: x). If for some system two arguments are necessary and sufficient to generate an output, then ipso facto minimal search is not implemented by a function that must search for n arguments, $n > 2$.

20 Binary Merge can effectively and efficiently simulate n-ary Merge. "Computations are the compositions of functions. A truth about functions of far-reaching significance for our understanding of the functional architecture of the brain is that functions of arbitrarily many arguments [e.g., n-ary Merge—Watumull] may be realized by the composition of functions that have only two-arguments [e.g., binary Merge—Watumull]" (Gallistel and King 2009: x).

21 *Generally*, a binary function cannot be *meaningfully* decomposed into a unary function. For Merge, binarity is irreducible, as unarity is noncombinatorial/noncompositional.

description y such that y represents x, x some program(=function); this measure of Kolmogorov complexity defines the binary Merge program/function as less complex than the n-ary Merge program/function (cf., (1) and (2)).

In sum, the assumption that Merge is restricted to binarity from its initial state of n-arity is fallacious: n-ary Merge is incomputable in a discrete infinity, undecidable for its unspecified arity, inexorably intractable with the spatiotemporal resources of the procedure implementing it increasing exponentially in the number of bits required to encode the input compactly, and inefficient because binary Merge (polynomially bounded by design) is necessary and sufficient for strong generativity.

It stands to reason that evolution would converge on the simplest stable function—binary—rather than jump to one that is complex and volatile: "any evolutionary process must first consider relatively simple systems, and thus discover the same, isolated, islands of efficiency [in the] universe of computation" (Minsky 1985: 122).[22]

5 Optimality

Binary Merge is computable, tractable, and optimal in minimizing abstract representations and spatiotemporal resources in the process of maximizing the strong generation of syntactic structures. But this holds only of *some form(s)* of binary Merge.[23] One form of which it is not true is union Merge (or "Unification"), a function that generates the union of its arguments (see Jackendoff 2011, Zwart 2011, Fortuny 2008, Langendoen 2003).[24]

(3) $f_{\text{unionMERGE}}(\gamma, \{\alpha, \beta\}) = \{\gamma, \alpha, \beta\}$

The elementary form of binary Merge postulated in (2) generates as an output the structure in (4) if fed the input of (3).

(4) $f_{\text{2-aryMERGE}}(\gamma, \{\alpha, \beta\}) = \{\gamma, \{\alpha, \beta\}\}$

22 It has been demonstrated mathematically (see Turing 1952 on morphogenesis, Mandelbrot 1982 on fractals) and simulated computationally (see Minsky 1985 on Turing machines, Wolfram 2002 on cellular automata) that evolution does tend converge on simple procedures generative of infinite complexity.

23 I defer to future work a proof of the theorem that (binary) "Parallel Merge" and its equivalents (see Ciko 2011) are surpassingly suboptimal and indeed that *any* multidominance model *cannot* be optimal vis-à-vis my formulation of *minimax* Merge. Notwithstanding protestations to the contrary, models of "intersecting sets" (see Seely 2011) reduce to multidominance models.

24 Union operations—effectively equivalent to union Merge—in the node contractions of Tree-Adjoining Grammars (see Sarkar and Joshi 1996) and reduced phrase markers (see Goodall 1987) are equally problematic, but a critique is beyond the scope of this squib.

In efficient computation, neither SO is modified under merger; this can be stated as the No-Tampering Condition NTC (see Chomsky 2005). But union Merge, by design and admission (see Jackendoff 2011), violates NTC: it erases the brackets—by an implicit associativity operation—of the complex SO(s).[25] In (3), for instance, the complex SO $\{\alpha, \beta\}$ is dismantled; the information representing the structural relation of those arguments to the exclusion of γ is deleted. And because in its tampering it is associative, union Merge flattens hierarchy—dissolving the fundamental syntactic property of structure-dependence (see Berwick et al. 2011)—which must then be reconstructed somehow to meet empirical demands, perhaps by stipulating implicit hierarchy in an inexplicably recorded or inferred order of mergers (see Zwart 2011, Fortuny 2008, Langendoen 2003).[26] Indeed, union Merge (Unification) must resort to a complex form of binary Merge, as Unifiers concede:

> Unification alone cannot create constituent structure: it only creates a Boolean combination of preexisting features and structures. In order to build structure, one needs a skeletal constituent structure that can be unified with two or more items. [One such schema] is a set {x, y} with variable elements x and y as parts. This can be unified with specific elements A and B to form the set {A, B}—in effect, the output of [binary] Merge. (Jackendoff 2011: 602-603)[27]

Not only is Unification unwieldy and bursting with redundancies, the only

25 Extensive erasure operations can render a system intractable: "Landauer's principle states that the erasure of data stored in a system has an inherent work cost and therefore dissipates heat" (Rio et al. 2011: 61). For this reason, erasure is probably the costliest operation in computation.

26 Recovering structure erased in the computation of union Merge could be incomputable ("nonrecursive") for the reason recovering deep structure deleted in the phrase structure grammar of Chomsky 1965 is (perhaps) incomputable: "If what a machine must do to recognize whether or not a given sentence (surface string) is in the language generated by some transformational grammar is to recover its deep structure, and if deep structures can be arbitrarily large compared to the surface strings derived from them, then the recognition procedures for such languages are not even recursive" (Berwick 1984: 190). Formally, "Let G be a transformational grammar. Let f_G be the cycling function of G, where $f_G x$ is 0 if x is not in $L(G)$ [the language generated by G], and otherwise is the least number s such that G assigns x a deep structure with s subsentences. If f_G is bounded by an elementary (primitive) recursive function, then $L(G)$ is elementary (primitive) recursive [...]. If the cycling function is not bounded, then $L(G)$ is not even recursive" (see Peters and Ritchie 1973). With union Merge, associativity deletes structure which must be recovered. For an unbounded SO, that structure is unbounded, and thus cannot be recovered.

27 An infinite number of such schemas are necessary given that the number of possible SOs to be unified is infinite.

procedures that can implement it are exponentially (i.e., inefficiently) bounded; simple binary Merge, however, which is simply combinatorial/compositional set-formation as in (4), does not modify its arguments, and thus complies with NTC—and can be implemented by polynomially bounded procedures (Robert C. Berwick, personal communication). It is this form of Merge—computable, tractable, compact, efficient, strongly generative—that reflects computational optimality (see Watumull 2010 for a formulation of Merge in these *minimax* terms).

6 Concluding Remarks

If Merge is necessarily binary as in (2) and (4), it necessarily generates hierarchically structured 2-sets (sets containing two elements), and thus such "binary branching" structures are predicted to be universal.[28] As stated in the introduction, my model "thus restricts the space of possible linguistic structures," and thus I concur with Gazdar, Klein, Pullum, and Sag (1985: 2) (emphasis original) that "The most interesting contribution a generative grammar can make to the search for universals of language is specify formal systems that have putative universals as *consequences*, as opposed to merely providing a technical vocabulary in terms of which autonomously stipulated universals can be expressed."

Given the binarity prediction, a presumption of dubiousness must obtain of non-compositional models (see Bresnan 2001) and theories that "the appropriate complexity for syntax is relatively flat: headed phrases that are linearly ordered and that correspond to constituents in Conceptual Structure, but not more" (Culicover and Jackendoff 2005: 108).[29] To be explicit, on my theory, flat structures—or structures not composed exclusively of 2-sets generally—are predicted to be impossible at worst or, at best, gross deviations from the virtual logical necessity of the principles established by the theories of computability and computational complexity. I should thus expand my prediction to deny (or assume as highly improbable) the existence of non-configurational languages.[30] If a language were as flat as Simpler Syntax (Culicover and Jack-

28 Technically, Merge only generates 2-*sets*, not *trees*: only the former are psychologically *real*; the latter mere *notations* that have now grown to mislead researchers into forests of fallacies (see the multidominance literature surveyed in Citko 2011). Thus technically my theory predicts the universality of structures of 2-sets—i.e., $\{\gamma_n, ..., \{\gamma_1, \{\alpha, \beta\}\}...\}$—representable (misleadingly) by binary branching trees.

29 The presumption of dubiousness extends to Categorial Grammars, which allow for flat structures (see Moortgat 1989, Bar-Hillel et al. 1964).

30 (i) is the hypothesized base rule of Hale 1981 and Chomsky 1981 in which W^* is a sequence of zero or more maximal projections; on my theory, such a rule—and any updated version—cannot be a rule of natural language.

endoff 2005) posits or, a fortiori, "had no phrase structure at all," but merely "units composed only of words linked by semantics and linear precedence rules" (Everett 2010: 7), then (i) countless complex associativity operations would need to be stipulated—and *conceptually motivated*—to erase the hierarchical structure binary Merge automatically generates and (ii) the effects of hierarchy (structure-dependence) would need to be recovered; these conditions are probably unsatisfiable because (i) could be intractable and (ii) could be incomputable (see footnote 26).

Thus I am arguing that the purported evidence for non-binary-branching structures needs to be reanalyzed consistent with my theory. This is not unreasonable: to my knowledge, no syntactic structure necessitates an analysis in terms of non-binary Merge, which perhaps seems simpler than binary Merge in some instances, although really it is not.[31] Thus, contrary to the "Boasian tradition," I am arguing that language *cannot* be "described [...] *without* any preexistent scheme of what a language must be" (Joos 1957: v) (emphasis added). The preexistent scheme must assume a generative procedure, and that procedure must be assumed *not* to be n-ary; it ought to be assumed to be binary.

(i) XP $\rightarrow W*$ X

Thus a conjecture such as "For Japanese, [D-structure] is a 'flat' structure formed by [(i)]" (Chomsky 1981: 132)—and any updated version—must be false.

31 The empirical phenomena adduced in Jackendoff 2011 are not problematic for simple binary Merge, but my arguments (Watumull 2012b) are too lengthy to construct here.

References

Aaronson, Scott. 2011. Why Philosophers Should Care About Computational Complexity. Ms., MIT.

Bar-Hillel, Yehoshua, Haïm Gaifman, and E. Shamir. 1964. On Categorial and Phrase Structure Grammars. In *Language and Information*, ed. by Yehoshua Bar-Hillel, 99-115. Reading: Addison-Wesley.

Berwick, Robert C. 1984. Strong Generative Capacity, Weak Generative Capacity, and Modern Linguistic Theories. *Computational Linguistics* 10: 189-202.

Berwick, Robert C., Paul Pietroski, Beracah Yankama, and Noam Chomsky. 2011. Poverty of the Stimulus Revisited. *Cognitive Science* 35: 1207-1242.

Biberauer, Theresa, Anders Holmberg, and Ian Roberts. 2011. A Syntactic Universal and its Consequences. Ms., University of Cambridge, Newcastle University.

Bresnan, Joan. 2001. *Lexical-Functional Syntax*. Oxford: Blackwell.

Carnap, Rudolph. 1955. Meaning and Synonymy in Natural Languages. *Philosophical Studies* 6: 33-47.

Chomsky, Noam. 1955. *The Logical Structure of Linguistic Theory*. New York: Springer.

Chomsky, Noam. 1965. *Aspects of the Theory of Syntax*. Cambridge: MIT Press.

Chomsky, Noam. 1981. *Lectures on Government and Binding*. Dordrecht: Foris.

Chomsky, Noam. 1995. *The Minimalist Program*. Cambridge: MIT Press.

Chomsky, Noam. 2005. Three Factors in Language Design. *Linguistic Inquiry* 36: 1-22.

Chomsky, Noam. 2008. On Phases. In *Foundational Issues in Linguistic Theory: Essays in Honor of Jean-Roger Vergnaud*, ed. by Robert Freiden, Carlos P. Otero, and Maria Luisa Zubizarreta, 133-166. Cambridge, MA: MIT Press.

Churchland, Paul M. 1988. Perceptual Plasticity and Theoretical Neutrality: A Reply to Jerry Fodor. *Philosophy of Science* 55: 167-187.

Citko, Barbara. 2011. *Symmetry in Syntax: Merge, Move, and Labels*. Cambridge: Cambridge University Press.

Culicover, Peter W., and Ray Jackendoff. 2005. *Simpler Syntax*. Oxford: Oxford University Press.

Deutsch, David. 2011. *The Beginning of Infinity: Explanations that Transform the World*. New York: Viking.

Edmonds, Jack. 1965. Paths, Trees, and Flowers. *Canadian Journal of Mathematics* 17: 449-467.

Enderton, Herbert B. 1977. Elements of Recursion Theory. In *Handbook of Mathematical Logic*, ed. by Jon Barwise,527-566. Amsterdam: North-Holland Publishing Company.

Everett, Daniel L. 2010. The Shrinking Chomskyan Corner: A Final Reply to Nevins, Pesetsky, and Rodrigues. Ms.,Illinois State University.

Fodor, Jerry, and Massimo Piattelli-Palmarini. 2010. *What Darwin Got Wrong*. New York: Farrar, Staus and Giroux.

Fortuny, Jordi. 2008. *The Emergence of Order in Syntax*. Amsterdam: John Benjamins.

Gallistel, C.R. 2003. The Principle of Adaptive Specialization as It Applies to Learning and Memory. In *Principles of Learning and Memory*, ed. by Rainer H. Kluwe, Gerd Lüer, and Frank Rösler, 259-280. Basel: Birkhäuser.

Gallistel, C.R., and Adam Philip King. 2009. *Memory and the Computational Brain: Why Cognitive Science Will Revolutionize Neuroscience*. New York: Wiley-Blackwell.

Gazdar, Gerald, Ewan Klein, Geoffrey K. Pullum, and Ivan A. Sag. 1985. *Generalized Phrase Structure Grammar*. Cambridge: Harvard University Press.

Gödel, Kurt. 1934 [1986]. On Undecidable Propositions of Formal Mathematical Systems. In *Kurt Gödel: Collected Works, Vol I: Publications 1929-1936*, ed. by Solomon Feferman, John W. Dawson, Stephen C. Kleene, Gregory H. Moore, Robert M. Solovay, and Jean Van Heijenoort, 346-371. Oxford: Oxford University Press.

Goodall, Grant. 1987. *Parallel Structures in Syntax*. Cambridge: Cambridge University Press.

Hale, Ken. 1981. *On the Position of Warlpiri in a Typology of the Base*. Indian University Linguistics Club, Bloomington.

Jackendoff, Ray. 2011. What is the Human Language Faculty? Two Views. *Language* 87: 586-624.

Joos, Martin. 1957. *Readings in Linguistics: The Development of Descriptive Linguistics in America since 1925.* Washington: American Council of Learned Societies.

Kayne, Richard S. 1981. Unambiguous Paths. In *Levels of Syntactic Representation*, ed. by Robert May and Jan Koster, 143-183. Dordrecht: Kluwer.

Langendoen, D. Terence. 2003. Merge. In *Formal Approaches to Function in Grammar: In Honor of Eloise Jelinek*,ed. by Andrew Carnie, Mary Willie, and Heidi Harley, 307-318. Amsterdam: John Benjamins.

Mandelbrot, Benoit B. 1982. *The Fractal Geometry of Nature.* New York: W.H. Freeman.

Marr, David. 1982. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information.* New York: Freeman.

Minsky, Marvin. 1967. *Computation: Finite and Infinite Machines.* Englewood Cliffs: Prentice Hall.

Minsky, Marvin. 1985. Why Intelligent Aliens Will Be Intelligible. In *Extraterrestrials*, ed. by Edward Regis, 117-128. Cambridge: MIT Press.

Moortgat, Michael. 1989. *Categorial Investigations: Logical and Linguistic Aspects of the Lambek Calculus.*Dordrecht: Foris.

Moro, Andrea. 2008. *The Boundaries of Babel: The Brain and the Enigma of Impossible Languages.* Cambridge: MIT Press.

Neumann, Jon von. 1928. Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen* 100: 295-320.

Pesetsky, David. 1989. Language-Particular Processes and the Earliness Principles. Ms., MIT.

Peters, P. Stanley, and R.W. Ritchie. 1973. On the Generative Power of Transformational Grammars. *Information Sciences* 6: 49-83.

Rio, Lídia del, Johan Åberg, Renato Renner, Oscar Dahlsten, and Vlatko Vedral. 2011. The Thermodynamic Meaning of Negative Entropy. *Nature* 474: 61-63.

Roberts, Ian. 2011. Parametric Hierarchies. Presented at SyntaxSquare, MIT, 8 December.

Rosenberg, Alex. 2011. How Jerry Fodor Slid Down The Slippery Slope To Anti-Darwinism, And How We CanAvoid The Same Fate. Ms., Duke

University.

Sarkar, Anoop, and Aravind Joshi. 1996. Coordination in Tree Adjoining Grammars: Formalization and Implementation. In *Proceedings of the 16th Conference on Computational Linguistics*, Volume 2, ed. by J. Tsujii, 610-615. Morristown: Association for Computational Linguistics.

Seely, Daniel. 2011. Maximizing Minimal Merge. Presented at *UG: The Minimum Workshop*, University of Durham, 18 December.

Shannon, Claude E. 1948. A Mathematical Theory of Communication. *Bell System Technical Journal* 27: 379-423,623-656.

Turing, Alan M. 1936. On Computable Numbers, With An Application To The Entscheidungsproblem. *Proceedings of the London Mathematical Society* 42: 230-265.

Turing, Alan M. 1950. Computing Machinery and Intelligence. *Mind* 59: 433-460.

Turing, Alan M. 1952. The Chemical Basis of Morphogenesis. *Philosophical Transactions of the Royal Society of London*, Series B, Biological Sciences 237: 37-72.

Yang, Charles. 1999. Unordered Merge and its Linearization. *Syntax* 2: 38-64.

Watumull. 2010. *Merge as a Minimax Solution to the Optimization Problem of Generativity*. MPhil Thesis, University of Cambridge.

Watumull. 2012a. A Turing Program for Linguistic Theory. *Biolinguistics* 6(**2**): 222-245.

Watumull 2012b. The Science of Deduction in Syntax. Ms. in preparation.

Weinberg, Steven. 1976. The Forces of Nature. *Bulletin of the American Academy of Arts and Sciences* 29: 13-29.

Wolfram, Stephen. *A New Kind of Science*. Champagne, Il: Wolfram Media, Inc.

Zwart, Jan-Wouter. 2011. Structure and Order: Asymmetric Merge. In *The Oxford Handbook of Linguistic Minimalism*, ed. by Cedric Boeckx. Oxford: Oxford University Press.

Jeffrey Watumull

Christ College
Cambridge, UK
CB2 3BU

jw647@cam.ac.uk

MIT Department of Linguistics and Philosophy
Laboratory for Information and Decision Systems
Cambridge, MA 02139, USA

watumull@mit.edu