A demonstration of the uncomputability of parametric models of language acquisition and a biologically plausible alternative

Evelina Leivada
Universitat Rovira i Virgili, Spain

Elliot Murphy
University of Texas Health Science Center at Houston, USA

Abstract: The logical problem of language acquisition has been at the forefront of psycholinguistics and behavioral neuroscience for decades. One of the most influential answers to the problem of how successful acquisition occurs on the basis of noisy input suggests that the child is aided by innate principles and parameters (P&P). These are conceived as part of our biological endowment for language. Previous work on the computability of parametric models has focused on the process of parameter-setting, leaving settability unaddressed. Settability is a key notion in parametric models since it provides an answer to the logical problem of language acquisition: the setting of one parameter carries implications for the settability of others, minimizing the child's task. However, a mathematical analysis of the expected probability of successful computation of settability relations has not been carried out. We report results from a novel program developed to calculate the probability of successful computation of a network of 62 linguistic parameters as attested in 28 languages, spanning across 5 language families. The results reveal that some parameters have an extremely low probability of successful computation, such that trillions of unsuccessful computations are expected before a successful setting occurs. Using the same program, we performed an additional analysis on a different network, covering 94 parameters from 58 languages and 15 language families. In this case, the estimated number of expected unsuccessful computations rose from trillions to quadrillions. These results raise concerns about the computational feasibility of the highly influential P&P approach to language development. Merging insights from various acquisition models, including some developed within P&P, a biologically plausible alternative is offered for the process of deciphering a target grammar in the acquisition of both spoken and signed languages. Overall, our analysis of the P&P approach to language acquisition centers learnability and computability constraints as the major factors for determining the psychological plausibility of grammar development.

Keywords: acquisition; computation; learnability; parameter; Universal Grammar

Corresponding author(s): Evelina Leivada, Department of English and German Studies, Universitat Rovira i Virgili, Av. Catalunya 35, 43002, Tarragona, Spain. Email: evelina.leivada@urv.cat

ORCID ID(s): https://orcid.org/0000-0003-3181-1917, https://orcid.org/0000-0003-1456-0343

Citation: Leivada, E., & Murphy, E. (2022). A demonstration of the uncomputability of parametric models of language acquisition and a biologically plausible alternative. *Language Development Research, 2*(1), 105–138. https://doi.org/10.34842/2022-585

Introduction

Language acquisition and its guiding principles have been at the forefront of psycholinguistic and developmental research for over five decades. Among the central research questions of the field, the logical problem of language acquisition stands out: How is language acquired, given the noisy nature of the linguistic input that a child receives during the early stages of development? The poverty of the environmental stimulus that characterizes the input sharply contrasts with the richness of the attainment that a neurotypical child will have as a mature speaker/signer (Chomsky, 1965; 1980). In (bio)linguistics and psychology, the highly influential Principles & Parameters framework (P&P) has provided an answer to the logical problem of language acquisition by positing that the child is aided by some innate principles that help them navigate the space of cross-linguistic variation in the process of acquisition (Chomsky, 1981). According to P&P, the child is innately equipped with a cognitive apparatus called Universal Grammar. Universal Grammar can be viewed as a cognitive map that consists of (i) a finite number of universal principles and (ii) a small number of parameters, that are also universal, but come with a set of values to which they are variably set across different languages. Although this idea has been around for several decades and has been criticized on various grounds, recently there has been a renewed interest in it, especially from a computational perspective that integrates Universal Grammar and non-linguistic principles of computation in the process of language development (Yang et al., 2017; Kazakov et al., 2018; Manzini, 2019). Even though the P&P framework removes some of the burden originally placed on the Evaluation Measure, it remains unclear what type of learning algorithm can manoeuvre itself through a space of grammars.

From a theoretical perspective, this organization of Universal Grammar in terms of principles and parameters brings an important benefit. Consider the overall volume of the input data a child has to process in order to acquire their language. Not only is it vast, but the task at hand entails dealing with noisy data and complex rules, whose properties the child has to decipher in the earliest stages of development. The logical problem of language acquisition addresses the question of how the child achieves this monumental task. The answer, within the P&P framework, is that the child's cognitive map consists of a finite number of parameters that form certain paths (Figure 1), such that the variation space is neatly compartmentalized, rendering the child's task considerably easier.

To explain the process, at point zero of acquisition the child has routes of the cognitive map open, but upon setting a few initial parameters to one value instead of another, the child selects a path. This selection brings with it the notion of *settability*: The values of the first-set parameters carry implications about the settability of others that are yet to be set. After selecting a route through setting a parameter to one value, the child is bound against exploring other routes, at least not in the context of that language. Parameters in these other routes will not be set to a value on the basis of the data the child is exposed to, because they are not settable: they do not form part of the route the child has taken. Since the child will never have to deal with them, the variation space that they have to navigate is substantially reduced. This explains (putatively) how the child performs this complex task so fast.

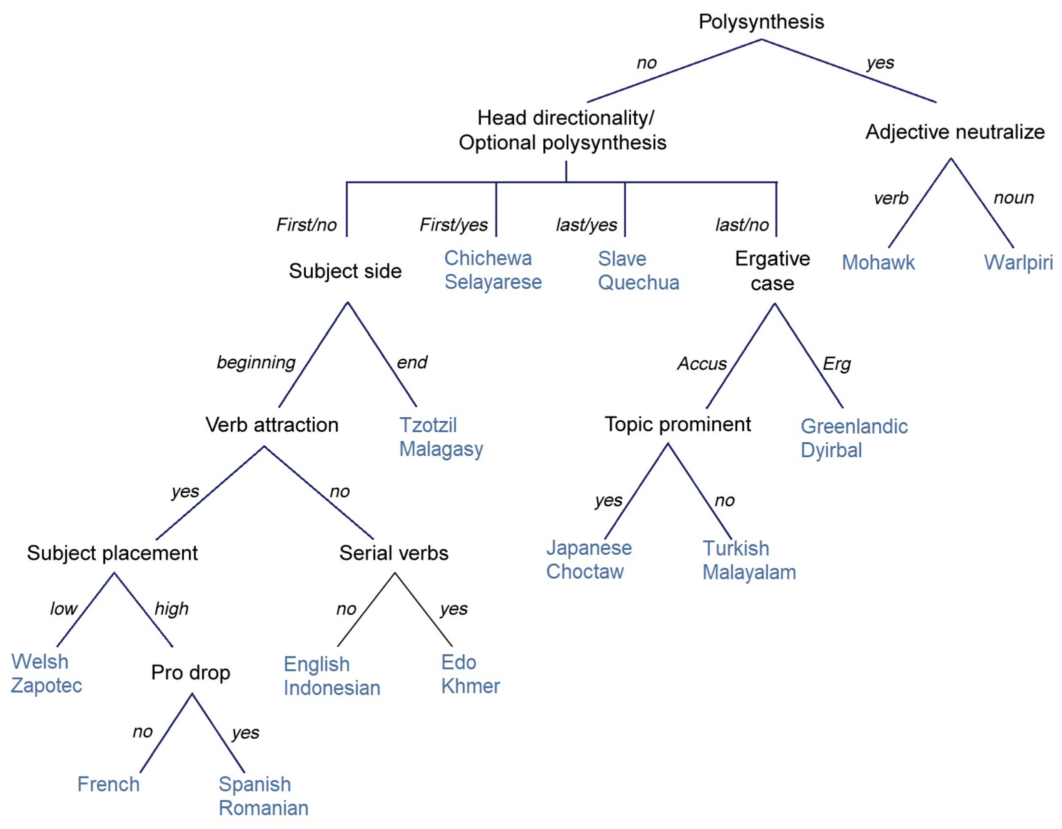


Figure 1. ***A parametric hierarchy (adapted from Baker 2003).***

To define the critical notions of setting and settability, parameter setting refers to selecting a value for a parameter, based on data from the target language. Parameter settability refers to whether a parameter forms part of the route the learner has taken. To give an example based on Figure 1, 'adjective neutralize' is not settable in French (i.e., it does not form part of the route to French), but it is settable and set to a specific value in Mohawk. A settable parameter is always set based on language data. Therefore, setting differs from settability in that the latter only arises given the existence of an *implicational network* among parameters (i.e., a network of dependencies that specifies that the settability of parameter X depends on having set parameter Y to one value instead of another, as shown in Figure 1). In this sense, the crucial difference between the two notions, setting and settability, boils down to the fact that the process of setting/value selection does not bear upon the existence of an implicational network; the latter is only informative about settability.

The processes of setting and settability are formally presented in (1).

(1a) Setting

Given a parametric hierarchy composed by a series of nodes N_i, where $i \in \mathbb{N}$, setting is the process of selecting a binary value $S_i = \{+N_i \text{ or } -N_i\}$. If you get input *z*, select a value. If *z* matches the hypothesized value, set N to this value. If not, select the other value and set N. Reach state N_{valued}.

(1b) Settability

Go to the next node N2. Check whether there is a path that connects N2 to any previous node (in this example, N_{valued}). A path entails a logical expression (e.g., N2=(N−)). If there is no path, set N2 following the process described in *Setting*. If there is a path, determine its satisfiability. A path is satisfied if the parts of the logical expression match the values of previously set nodes (e.g., if the logical expression is N2=(N−), then N_{valued} must be set to −. If it is not, the path is not satisfied and settability of N2 cannot be reached on this path). Repeat for every path that connects N2 to previous nodes. If one (or more than one, but at least one) path is satisfied, follow the process described in *Setting* to set N2. If no path is satisfied, rewrite N2 as $N2_{not\text{-}settable}$.

There are two possible outcomes: $N2_{valued}$ or $N2_{not\text{-}settable}$. Once any of the two is reached, go to the next node N3 and repeat the process. When all nodes have reached one of the two states, N_{valued} or $N_{not\text{-}settable}$, halt the process.

These two notions, setting and settability, have not been investigated to equal degrees. Previous work concerning the computation of parametric models of language acquisition has focused almost exclusively on analyzing setting relations; for example, the number of linguistic examples and initial hypotheses that are needed for the child to set the parameters that correspond to their target language (Gibson & Wexler, 1994; Niyogi & Berwick, 1996). Settability has not been addressed from a computational perspective, in part because until recently it was largely assumed that there is only one way of reaching settability for a given parameter in a given language; an assumption that voids the need for further computation.

To illustrate this assumption, Figure 1 shows that there is a single way to reach the settability of any parameter in this parametric hierarchy (e.g., 'adjective neutralize' is reached exclusively by setting polysynthesis to [+]). The only work that addresses the computation of settability relations challenged this assumption of unique settability (Boeckx & Leivada, 2013), through examining an elaborate network of parameters from the nominal domain (henceforth, the network, Longobardi & Guardiano, 2009; Figure 2).

No	Parameters & Paths	It	Sal	Sp	Fr	Ptg	Ru	Lat	ClG	NTG	Gri	Grk	Got	OE	E	D	Nor	Blg	SC	Rus	Ir	Wel	Heb	Ar	Wo	Hu	Ba
1	± grammatical person	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
2	± grammatical number (+1)	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
3	± grammatical gender (+2)	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	-	-
4	± variable person on D (+1)	-	-	+	-	-	-	?	+	?	-	+	?	?	-	-	-	-	-	-	?	?	-	+	-	-	+
5	± feature spread to N (+2)	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	-	+	-
6	± number on N (Bare Nouns) (+5)	+	+	+	-	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	0	+	0
7	± grammaticalized partial definiteness	+	+	+	+	+	+	-	+	+	+	+	+	+	+	+	+	+	-	-	+	+	+	+	+	+	-
8	± grammaticalized definiteness (+7)	+	+	+	+	+	+	0	+	+	+	+	-	+	+	+	+	+	0	0	+	+	+	+	+	+	0
9	± free null partitive Q (+6)	-	-	-	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	0
10	± grammaticalized distal article (-5 or -6 or +7)	-	-	-	-	-	-	0	-	-	-	-	-	-	-	-	-	-	0	0	-	-	-	-	+	-	+
11	± grammaticalized topic article (-10)	-	+	-	-	-	-	0	-	-	-	-	-	-	-	-	-	-	0	0	-	-	-	-	0	-	0
12	± definiteness checking N (+7)	-	-	-	-	-	+	0	-	-	-	-	-	-	-	-	+	+	0	0	-	-	-	-	-	-	0
13	± definiteness spread to N (+12)	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	+	-	0	0	0	0	0	0	0	0	0
14	± definiteness on attributes (+7, -12)	-	-	-	-	-	0	0	+	+	-	+	+	-	-	-	0	0	0	0	-	-	+	+	-	-	0
15	± definiteness on relatives (+7)	-	-	-	-	-	-	0	-	-	-	-	-	-	-	-	-	-	0	0	-	-	-	+	+	-	0
16	± D-controlled inflection on N (+5)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	0	-	0
17	± grammaticalized cardinal nouns	-	-	-	-	-	-	-	-	-	-	-	-	+	-	-	-	?	+	+	?	+	+	+	?	-	-
18	± grammaticalized cardinal adjectives (+17)	0	0	0	0	0	0	0	0	0	0	0	0	+	0	0	0	?	-	+	?	-	-	+	?	0	0
19	± plural spread from cardinals (+5, -17 or +18)	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	?	0	+	?	0	0	+	0	-	0
20	± grammaticalized mass-to-count	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	?	+	-	+	-	-	-
21	± N-to-predicate incorporation	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	+	-
22	± grammaticalized partial count (-5 or -6 or +7,-21)	+	+	+	+	+	+	0	-	-	+	+	0	+	+	+	+	+	0	0	-	-	-	-	0	0	+
23	± grammaticalized count (+22)	+	+	+	+	+	+	0	0	0	+	+	0	-	+	+	+	-	0	0	0	0	0	0	0	0	-
24	± count-checking N (+21 or +22)	-	-	-	-	-	-	0	0	0	-	-	0	-	-	-	-	-	0	0	0	0	0	0	-	-	-
25	± prepositional Genitive	+	+	+	+	+	-	-	-	-	-	-	-	-	+	+	+	+	-	-	+	+	+	+	-	-	-
26	± free inflected Genitive (-25)	0	0	0	0	0	-	+	+	-	-	-	-	-	0	0	0	0	-	-	0	0	0	0	-	-	-
27	± Genitive O (+25 or -26)	-	-	-	-	-	-	0	0	+	+	+	+	+	-	+	?	-	+	+	+	+	-	-	+	-	+
28	± Genitive S (+25 or -26)	-	-	-	-	-	+	0	0	-	-	-	+	+	+	+	+	-	-	-	-	-	+	+	-	+	+
29	± postpositional Genitive (+27 or +28)	0	0	0	0	0	-	0	0	-	-	-	-	-	+	-	+	0	-	-	-	-	-	-	-	-	+
30	± Genitive over DemP	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	?
31	± poss-checking N	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	+	+	+	-
32	± structured APs	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	-	+	-
33	± feature spread to structured Aps (+32)	+	+	+	+	+	+	+	+	+	+	+	+	+	-	+	+	+	+	+	+	?	+	+	0	-	-
34	± feature spread to predicative APs	+	+	+	+	+	+	+	+	+	+	+	+	+	-	-	+	+	+	+	+	?	+	+	+	+	+
35	± number on A (+6,+33 or +34)	+	+	+	0	+	+	+	+	+	+	+	+	+	0	+	+	+	+	+	+	?	+	+	0	+	0
36	± D-controlled inflection on Adjectives (+33)	-	-	-	-	-	-	-	-	-	-	-	+	+	0	+	+	-	-	-	-	?	-	-	0	0	0
37	± DemP over relative clauses	+	+	+	+	+	+	?	?	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	-
38	± free APs in Modifier Phrase (+32)	+	+	+	+	+	-	+	+	+	-	+	+	-	-	-	-	-	-	-	-	-	-	+	0	-	-
39	± APs in Modifier Phrase (-38)	0	0	0	0	0	-	0	0	0	-	0	0	+	+	-	-	+	+	+	-	-	+	0	0	-	?
40	± overt Mod° (-32 or +38 or +39)	-	+	-	-	-	0	-	-	-	0	-	-	-	-	0	0	-	-	-	0	0	-	-	+	0	?
41	± adjectival Genitive	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	+	+	-	-	-	-	-	-	-
42	± N-raising with pied-piping	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	+	-	-	+
43	± N over external argument (-42)	+	+	+	+	+	+	+	+	+	+	+	+	+	-	+	-	+	+	+	+	+	0	0	+	+	0
44	± N over Genitive O (+26 or +27, -30, +43)	0	0	0	0	0	0	-	-	+	+	+	+	+	0	+	0	0	+	+	+	+	0	0	+	0	0
45	± N over Adjectives (+32, (-26 or -27, +43) or +44)	+	+	+	+	+	+	0	0	-	+	-	-	-	0	-	0	-	-	-	+	+	0	0	0	-	0
46	± N over Manner 2 Adjectives (+45)	+	+	+	+	+	+	0	0	0	+	0	0	0	0	0	0	0	0	0	+	+	0	0	0	0	0
47	± N over Manner 1 Adjectives (+46)	-	+	-	-	-	-	0	0	0	?	0	0	0	0	0	0	0	0	0	+	+	0	0	0	0	0
48	± N over high Adjectives (+47)	0	+	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0	0	-	-	0	0	0	0	0
49	± N over cardinals (+42 or +48)	0	-	0	0	0	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	+	+	?	0	-
50	± strong D (person) (+1, +8 or +28)	+	+	+	+	+	+	0	+	+	+	+	0	-	-	-	-	+	0	0	-	-	+	+	-	?	+
51	± NP over D	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-	+
52	± N strong deixis	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	-	-	+	+	+	+	+
53	± strong anaphoricity (+52)	+	+	-	+	+	-	-	-	-	+	-	-	+	+	+	+	-	+	+	0	0	?	-	+	+	+
54	± DP over Demonstratives (-51, +52)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	+	?	0	-	0
55	± D-checking Demonstratives (-5 or -6 or +7, +52)	+	+	+	+	+	+	0	-	-	+	-	?	+	+	+	-	+	0	0	0	0	+	-	+	-	+
56	± D-checking possessives (-5 or -6 or +8, +50 or -28)	-	-	+	+	?	-	0	-	-	-	-	0	0	0	0	0	-	0	0	+	+	-	-	+	?	-
57	± feature spread on possessives (+33 or +34 or +35)	+	+	+	+	+	+	+	+	+	-	-	+	+	0	+	+	+	0	0	-	?	-	-	+	-	-
58	± feature spread on postpositional Genitive (+29, +57)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0
59	± enclitic possessives	-	-	-	-	-	-	-	+	+	+	+	-	-	-	-	-	?	-	-	-	-	-	-	?	-	-
60	± Consistency Principle (-43 or -44 or -45 or -46 or -47 or +51)	+	0	+	+	+	+	?	?	?	?	-	?	+	+	+	+	-	-	-	0	0	0	0	?	+	+
61	± null N-licensing article (-5 or -6 or -12, +50 or +51)	-	-	+	-	+	0	0	+	+	-	-	0	0	0	0	0	0	0	0	0	0	-	-	?	?	+
62	± obl. def. inheritance (+7, -22, (-25, +26) or +27, +42 or +45 or -50)	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	-	+	+	+	-	0	0
63	± gramm. geographical article (-5 or -6 or +7,-22 or -23 or +45)	+	+	-	+	-	+	0	+	+	+	0	0	0	0	0	0	-	0	0	?	?	-	-	0	-	-
		It	Sal	Sp	Fr	Ptg	Ru	Lat	ClG	NTG	Gri	Grk	Got	OE	E	D	Nor	Blg	SC	Rus	Ir	Wel	Heb	Ar	Wo	Hu	Ba

Figure 2. ***The analyzed network consists of 63 binary parameters from the nominal domain across 28 languages (adapted from Longobardi & Guardiano, 2009). The first column presents the parameters and the settability path(s) on which each parameter is settable. If a settability path is not available in a language, the corresponding parameter is marked with 0 (e.g., if [5settable] depends on [4–], if the latter is in any other state, the former is marked with 0, which indicates that the parameter is not settable in the specific language). ',' means ∧.***

The assumption of unique settability was investigated through the use of a program that calculated whether the settability paths in Figure 2 were satisfied in each language-parameter pairing that exists in the network (Boeckx & Leivada, 2013). For example, if the network specifies that the settability of parameter (P) 14 is reached on the basis of setting P7 to + and P12 to −, the settability path would be: [14settable] = [7+] AND [12−]. The program read these paths in the form of logical expressions and checked whether they were satisfied in the input it received. The input was the states of each parameter in each language, as they are shown in the language columns of Figure 2. Proceeding with the previous example, if in language X, P7 was set to + and P12 was set to −, the program returned the outcome 'true' for [14settable]. If P7 and P12 were in any other state (i.e., set in the opposite value or not-settable), the program returned the outcome 'false', which means that P14 is not settable (on this settability path) for language X.

As the 'OR' nodes in the first column of Figure 2 suggest, the network makes available different paths for the settability of many of its parameters. Until the computation of the settability relations of every language-parameter pairing, it was unclear whether different settability paths existed for different languages or whether the same language could involve more than one path for the same parameter; something that would disprove the assumption of unique settability. Previous work on the computation of settability relations determined that there are different ways to reach settability of a parameter, not only across but also within languages (Boeckx & Leivada, 2013). For example, Table 1 shows that for many languages in the network, parameter 29 is not settable (e.g., It[alian] in the second column). For other languages, the parameter is settable in one (e.g., path 4 in Rom[anian]) or more ways (e.g., paths 3 and 4 in Ba[sque])

Table 1. ***Parameter 29: ± Postpositional Genitive. 1 signals the availability of the corresponding settability path in the relevant language, whereas 0 signals the unavailability of the path. When a number node in the first column has an attached parenthesis on its right (e.g., 2+(1+)), the node inside the parenthesis is the settability path of the node outside the parenthesis, until an independent parameter is reached. In this table, the settability of parameter 29 is possible on the basis of setting either 27 to + or 28 to +. Both 27 and 28 are dependent parameters, settable in two ways each, either through setting 25 to + or 26 to −. Parameter 25 is an independent parameter which means that its settability does not depend on the setting of other parameters (Boeckx & Leivada, 2013).***

4 Paths	**It**	**Sal**	**Sp**	**Fr**	**Ptg**	**Rom**	**Lat**	**ClG**	**NTG**	**Gri**	**Grk**	**Got**	**OE**	**E**	**D**	**Nor**	**Blg**	**SC**	**Rus**	**Ir**	**Wel**	**Heb**	**Ar**	**Wo**	**Hu**	**Fin**	**Hi**	**Ba**
27+(25+)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0
28+(25+)	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	0	0	0	0	0
27+(26-(25-))	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	1	1	0	0	0	0	1	0	1	1	1
28+(26-(25-))	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

However, there is a crucial and thus far unproven assumption behind previous work on the computation of settability relations. The program that was used in Boeckx &

Leivada (2013) was a semi-automatic one: the settability paths were not computed by it, but were given to it as predefined logical expressions. The crucial assumption is that the processing system, be it the human cognitive parser or a custom-made program that simulates the process of computation, *can* successfully compute more than one settability path for a single parameter. If the settability of a parameter can be determined in more than one way, the parser must engage in some kind of computation that exhaustively checks all the paths that lead to it in order to determine whether the parameter is settable. This happens because the process of determining settability is a necessary prerequisite for the process of parameter-setting. Given that (i) not all parameters are settable in all languages and (ii) the learner does not know a priori which parameters are not, because the settability paths become available progressively, depending on the value of earlier set parameters, the learner must engage in some kind of computation that determines whether the parameter that it encounters next in the hierarchy is settable or not.

The aim of the present work is to spell out the computation of settability relations, *locally* for each dependent parameter of the analyzed network. More specifically, by means of treating each settability path as a logical expression (examples (2)-(3)), the satisfiability of each path must be calculated by the parser, be it the human brain or, in this case, a program that will simulate the computational process. In terms of the parametric network that will be analyzed, the notion of satisfiability refers to whether a path involves parameter values that match the input (given in Figure 2), such that this path is available in a language-parameter pairing, making the parameter settable in the specific language.

(2) The logical expression for the second settability path of P10: $(5-) \wedge (2+) \wedge (1+)$

(3) The logical expression for all the paths of P10: $((5-) \wedge (2+) \wedge (1+)) \vee ((6-) \wedge (5+) \wedge (2+) \wedge (1+)) \vee (7+)$

The computation that follows operates on the basis of two important characteristics of the network and the learner respectively. First, if a parameter involves more than one settability path, the computation does not halt after finding a satisfiable path for a parameter. Instead, all paths need to be checked for satisfiability. In order to understand this characteristic, it is necessary to take into account that a parameter's settability paths often materialize at different times. For example, Table 2 shows that for P24, the first path becomes available after P21 is set to +, while the second path materializes after P22 is set to +. Even if the availability of a path for P24 was to be checked when [21+] was achieved, the computation would need to be re-run when [22+] was achieved, because not all languages set P24 on the first path (e.g., Ba in table 2).

Table 2. ***Parameter 24: ± Count-Checking N. 1 signals the availability of the corresponding settability path in the relevant language, whereas 0 signals the unavailability of the path. ',' means ∧ (Boeckx & Leivada, 2013).***

4 Paths	**It**	**Sal**	**Sp**	**Fr**	**Ptg**	**Rum**	**Lat**	**ClG**	**NTG**	**Gri**	**Grk**	**Got**	**OE**	**E**	**D**	**Nor**	**Blg**	**SC**	**Rus**	**Ir**	**Wel**	**Heb**	**Ar**	**Wo**	**Hu**	**Fin**	**Hi**	**Ba**
21+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0
22+(7+, 21-)	1	1	1	1	1	1	0	0	0	1	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
22+((5-(2+(1+))), 21-)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
22+((6-(5+(2+(1+)))), 21-)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The second important characteristic is that the learner cannot remember the parametric nodes that formed part of a previously checked path and reuse this information when checking paths that materialize later. For example, the last two paths of P24 in Table 2 share some parameters. Still, the satisfiability of the logical expression needs to be checked for the last path too. The reason has to do with memory limitations. Even setting aside interference concerns that arise from keeping track of paths that materialize *at different points* (hence are separated by the setting of other parameters that occurs in between their materialization), working memory has a capacity of maintaining four units, on average (Cowan, 2000). A set of three parametric nodes and their values already exceeds this capacity, and most paths are considerably longer than this.

Not only is this information not retainable in memory due to its heavy load, but the parser does not have memory that goes beyond the current state. Parametric models in language acquisition have long been described as involving memoryless processing, in the sense that at any step the learner has no recall of prior input or states, beyond the ones currently entertained (Page, 2004; Fodor & Sakas, 2005; Fodor, 2009). This memoryless character of the learning process has also been a crucial assumption in prior work on the computation of setting relations in parametric models (Niyogi & Berwick, 1996; 1997). We stress that while aspects of contemporary neuroscience support a view of the brain's memory as being capable of a pushdown stack (beyond deterministic pushdown automata), the mature state of a mildly context-sensitive grammar would presumably not be attainable immediately to the infant (Gallistel & King, 2009). This means that when the learner deals with the settability of P24 (Table 2), it cannot shorten its last three paths through rewriting P22 as settable/non-settable (i.e., it will not remember whether P22 was or was not settable in a language and replace the paths that determine its settability with this information), because it lacks the read/write memory of a Turing machine. Put differently, the four paths of P24 that are shown in Table 2 cannot be rewritten as two paths, 21+ and 22+, by means of collapsing the different ways of reaching the settability of the latter. Additionally, such a move, apart from clashing with standard assumptions about properties of the brain, would raise empirical concerns. For example, Table 2 shows that French sets this parameter on two paths that depend on two different ways of reaching the settability of P22 (i.e., paths 2 and 4). Collapsing these two into one would simply not capture the facts for this language.

Taking into account these two characteristics, the present work aims to determine the computability of the settability paths behind the parameters of the analyzed network, through calculating the probability of running into loops that impede halting. For the computation to be successful, the learner needs to check the satisfiability of all the settability paths behind a parameter and halt. For example, if a parameter involves only two settability paths, A and B, further computation is not necessary, because the parser keeps track of the current state and will proceed to the next path without running into a loop: after checking both paths in one of the two possible orders, AB or BA, the computation will halt successfully. We can thus say that a parameter that has two settability paths has two ways of computation (i.e., two ways or orders of parsing the set of two paths): AB and BA. However, as the number of paths grows, the number of ways a set of paths can be checked for satisfiability also grows: two paths have two possible ways of computation (AB or BA), three paths have six ways (ABC, ACB, BAC, BCA, CAB, CBA), etc. In order to determine to what degree the ways of computation grow in the parametric network under examination, the program we describe below was designed to automatically calculate the probability of successful computation for each dependent parameter of the network, by estimating the ratio of successful computation to unsuccessful computation. The former refers to the number of ways the entire set of paths behind a parameter can be computed (i.e., checked for satisfiability) without running into loops; the latter refers to the number of ways that it runs into a single loop.

Method

The Longobardi & Guardiano network (Figure 2) consists of 63 parameters in 23 contemporary and 5 ancient languages, mostly from the Indo-European family. It is one of the most detailed parametric networks in the literature, rendering it an ideal candidate for computing settability relations. The present analysis used the slightly amended version of the network that was presented in previous work on the computability of parametric relations (Boeckx & Leivada 2013), in which parameter 62 was eliminated due to errors in its formulation. This elimination reduces the total number of the discussed parameters from 63 to 62. From these 62 parameters, 21 are settable on more than two paths, and hence these are the parameters analyzed in the present work.

In order to calculate the number of possible ways of successful computation (i.e., no loop), for n number of paths, $n! = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 2 \cdot 1$. For example, if n = 3, in the first random selection of a path, there are three options to choose from. In the second selection, there are n−1 options, and in the third selection, there are n−2 options, since no repetitions are permitted in successful computation. Therefore, for $n = 3$, there are $3 \cdot 2 \cdot 1 = 6$ ways of computation that do not run into a loop.

Calculating the ways of computation that feature a loop (i.e., one repetition of a previously checked path), in the first and the second random selection of a path, there cannot exist any repetition: in the first one, there is nothing to be repeated, and in the second one, the first selection will be remembered as the current path from which the learner is moving. From the third random selection of a path onwards, the total number of ways of unsuccessful computation is the sum of the different ways of unsuccessful computation when we take a subset *k* of *n*. The formula to calculate this is the following:

$$\sum_{k=3}^{k=n} \mathrm{n} \cdot (\mathrm{n}-1) \cdot \ldots \cdot \big(\mathrm{n}-(\mathrm{k}-2)\big) \cdot (\mathrm{k}-2)$$

For example, if a parameter has 5 settability paths ($n = 5$), for $k = 1$ and $k = 2$, there cannot be any repetitions. For $k = 3$, where 3 is the third random selection of a path, the number of ways of computation without repetition is $n \cdot (n-1) = 5 \cdot (5-1) = 20$. In order to calculate the number of ways of computation that feature a repetition in this third selection, this number must be multiplied by the number of paths that can be repeated. This is $k-2$ because the learner keeps track of the current state, so it cannot repeat the path it last checked. Therefore, for the third selection, $(k-2) \cdot 20$ gives a total of 20. For $k = 4$, the possible ways of computation without repetition are $n \cdot (n-1) \cdot (n-2) = 5 \cdot 4 \cdot 3 = 60$. This is multiplied by the number of paths that can be repeated, which is $k-2 = 2$, thus for $k = 4$, the number of ways of computation that have a repetition is $60 \cdot 2 = 120$. For $k = 5$, the possible ways of computation without repetition are $n \cdot (n-1) \cdot (n-2) \cdot (n-3) = 120$. This is multiplied by the number of paths that can be repeated, which is $k-2 = 3$, so for $k = 5$, the total number of ways of computation with repetition is $120 \cdot 3 = 360$. Overall, the total number of ways of unsuccessful computation for $n = 5$ is $360 + 120 + 20 = 500$.

For $n = 5$, the number of possible computations with and without loops is small, hence easy to calculate. However, many of the parameters in the analyzed network involve more than 10 paths. For this reason, a program was developed in Python in order to carry out the computation automatically (see Appendix for code). The program asks the user to provide the number of paths that should be computed. Upon being given a number followed by 'enter', it performs the calculation and asks the user whether they wish to perform another calculation for a different number of paths. Pressing '1' and then 'enter' restarts the process for another calculation, while pressing '2' and 'enter' closes the program. The program can be used to perform these calculations for any parametric model.

Results

The analysis produced two results: (i) the number of ways of successful and unsuccessful computation and (ii) the probability of successful computation for each parameter. Computation here refers not to the process of parameter-setting, but rather to going through the settability paths behind each parameter by means of checking the satisfiability of the logical expressions behind the paths ((2)-(3)). As noted, Figure 2 shows the Longobardi & Guardiano network. However, it provides no information as to how many paths the learner has to go through in order to determine settability and how many ways of computation (i.e., the process of "going through the set of paths") exist. Figure 3 addresses this gap by showing the degree to which the numbers for successful and unsuccessful computation rise in relation to the number of paths. More specifically, the average number of paths for the analyzed parameters is 8. For $n = 8$, there are 40,320 ways of successful computation and 375,368 ways of unsuccessful computation. This means that when a parameter has 8 settability paths, the memoryless parsing process has a total of 415,688 ways of going through them in order to check their satisfiability. For 10 paths, the number rises to 3,628,800 ways of successful computation and 46,253,610 ways of unsuccessful computation, while for 12 paths, the equivalent numbers are 6,227,020,800 and 1.11471e+11.

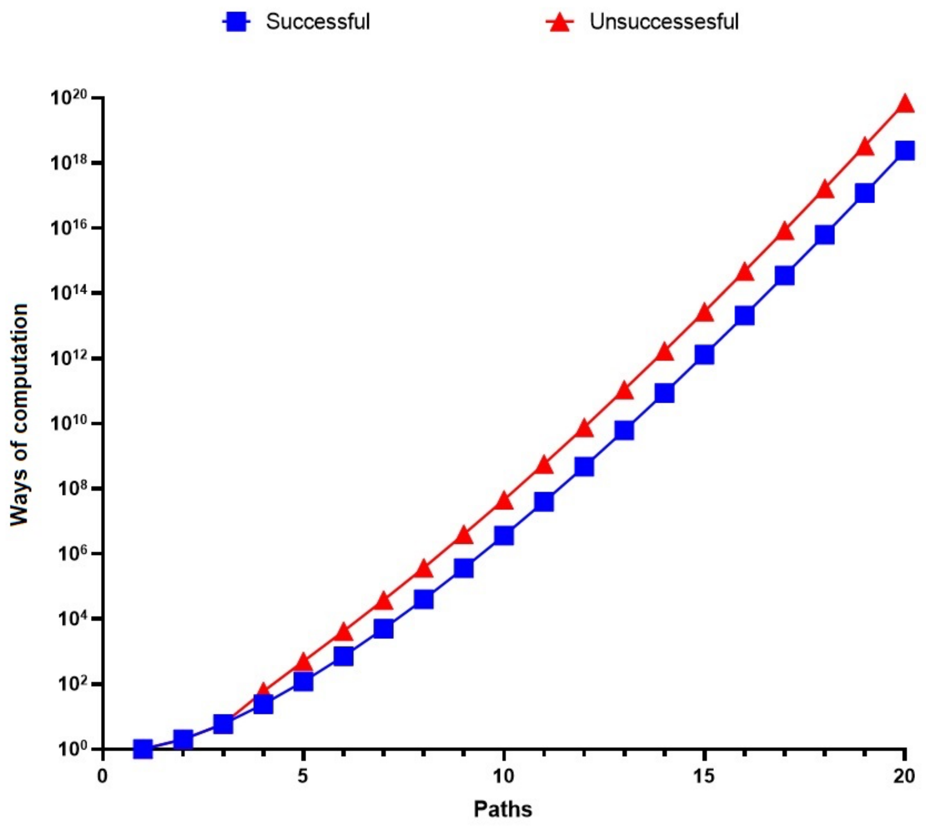


Figure 3. ***Number of ways of successful and unsuccessful computation across paths.***

Focusing on the Longobardi & Guardiano network, Figure 4 shows the ways of successful and unsuccessful computation for the parameters that have 3 or more settability paths. With the exception of the parameters that have just 3 paths, for all other parameters, the number of computations that run into a loop is considerably higher than the number of successful computations.

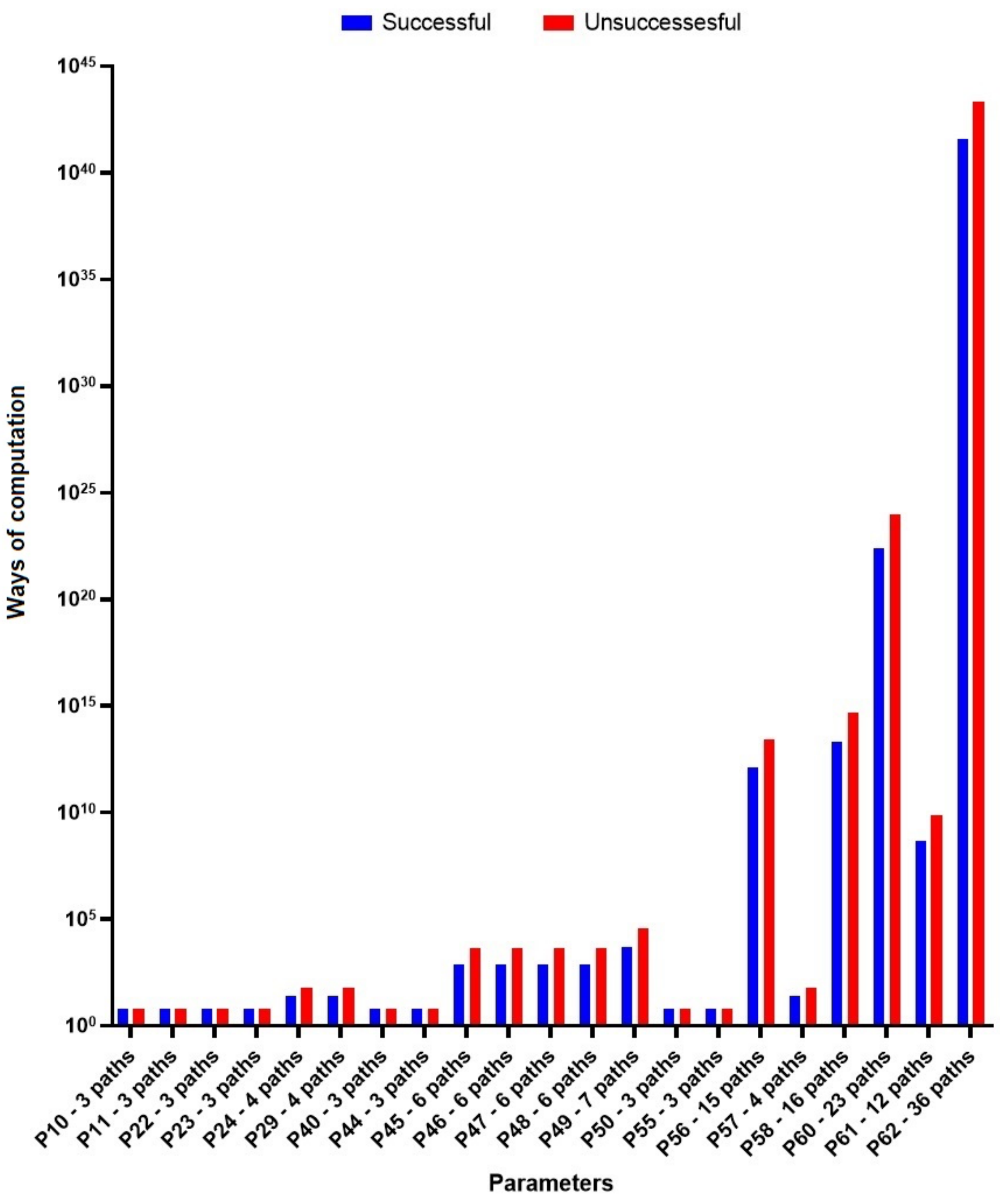


Figure 4. ***Number of ways of successful and unsuccessful computation for the 21 parameters of the analyzed network.***

The analyzed parameters involve a total of 169 settability paths. The probability of successful computation for each parameter independently is given in Figure 5.

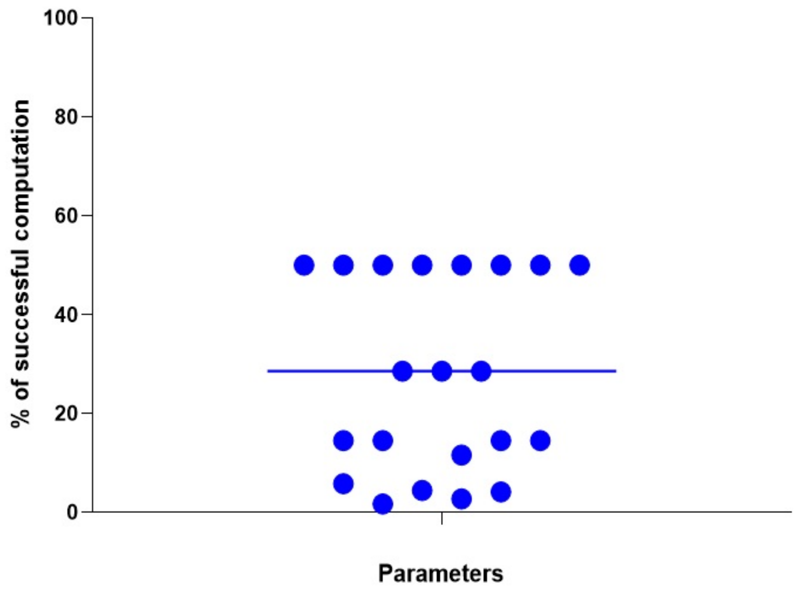


Figure 5. ***Probability of successful computation for the 21 parameters of the analyzed network with each parameter treated as independent.***

If a parameter has 3 or more settability paths, the probability of successful computation is equal or lower than 50%, respectively. However, the analyzed parameters are *dependent* parameters: their settability depends on having set other parameters to one value instead of another. As Figure 4 shows, in the Longobardi & Guardiano network, the first parameter that has 3 paths is P10. For 3 paths, the probability of successful computation on the first try is 50%. The second parameter that has 3 paths is P11. If this is taken as an independent event, the probability of successful computation is again 50%. However, if one wants to calculate the probability of a second successful computation under the assumption that the first parameter was computed successfully in one try, the conditional probability of successful computation in this second step is 25%. Table 3 shows that by the time the fifth parameter with 3 or more paths is encountered, the conditional probability of successful computation is 1.7%.

Table 3. ***Conditional probability of successful computation in one attempt (Longobardi & Guardiano network).***

Parameter	Ways of computation without loop	Ways of computation with one loop	Conditional probability of successful computation
P10 - 3 paths	6	6	50%

P11 - 3 paths	6	6	25%
P22 - 3 paths	6	6	12.5%
P23 - 3 paths	6	6	6.25%
P24 - 4 paths	24	60	1.7856%
P29 - 4 paths	24	60	0.5101%
P40 - 3 paths	6	6	0.2550%
P44 - 3 paths	6	6	0.1275%
P45 - 6 paths	720	4230	0.01858%
P46 - 6 paths	720	4230	0.00269%
P47 - 6 paths	720	4230	0.00039%
P48 - 6 paths	720	4230	0.000057%
P49 - 7 paths	5040	38262	0.00000662%
P50 - 3 paths	6	6	0.00000331%
P55 - 3 paths	6	6	0.00000165%
P56 - 15 paths	1.30767e+12	2.79027e+13	0.0000000740%
P57 - 4 paths	24	60	0.0000000211%
P58 - 16 paths	2.09228e+13	4.82395e+14	0.000000000878%
P60 - 23 paths	2.5852e+22	9.0699e+23	0.0000000000243%
P61 - 12 paths	479001600	7751595852	0.00000000000141%
P62 - 36 paths	3.71993e+41	2.13604e+43	0.00000000000002%

These findings raise concerns about computability, even if one assumes that the learner can somehow keep track of the fact that they have run into a loop, hence know that the computation should be re-run. This is highly pertinent in the context of a memoryless parsing process that knows only the current state. To explain the process from the learner's perspective, if on the first random selection of a path, A is chosen out of a set of paths ABCD, when the repetition of A occurs in the fourth selection (i.e., ABCA), the computation runs into a loop. Of course, the program that simulates the process keeps track of this possibility and flags it as a loop, because it was designed to do so. Yet the learner, who is equipped with a memoryless parser that lacks this feature, has no way of remembering which option was selected in the first/*n*th random selection of a path. If the learner keeps track of the current state, ABCC can be recognized as a loop and be avoided, but ABCA cannot. In other words, the parser is oblivious to the fact that it runs into a loop more often than not.

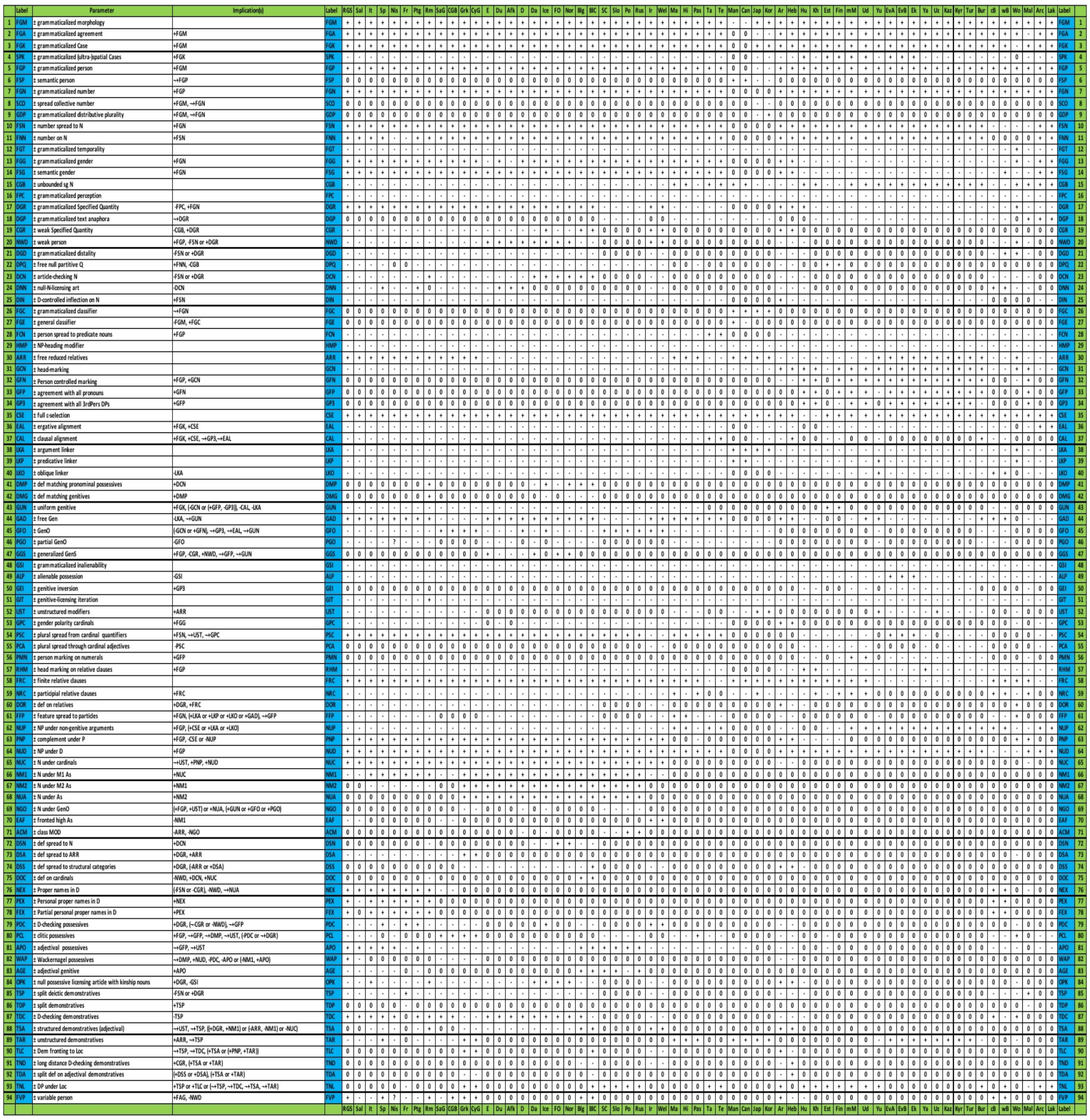
Even if we endow the parser with the ability to recognize a loop and rerun the computation, concerns about computability are not sidestepped. The reason boils down to how the numbers of successful and unsuccessful computations were calculated above. It is important to stress that the developed program treats the presence of a single loop as an instance of unsuccessful computation. This means that if a parameter is settable on 4 paths ABCD, the order ABCA is a possible outcome that the program counts as unsuccessful, but ABCAB or ABAB are not possible outcomes for the program. Put another way, the program is purposely designed to count the event of falling into one single loop as the only case of unsuccessful computation, but in reality, the number of computations that involve a loop are infinite.

Restricting the possible number of unsuccessful computations by limiting the number of the random selection of paths to the number of paths (i.e., if a parameter has 4 paths ABCD, the learner is allowed to perform only 4 events of random path selections, enabling ABAB as an unsuccessful outcome, but not ABABA, thereby limiting the number of unsuccessful computations) does not alleviate concerns about computability. Under this limitation, a parameter with 36 settability paths has a total of $n \cdot (n-1)^{n-1} = 36 \cdot 35^{35} = 3.97e + 55$ ways of computation. Consider the computations that do not involve a repetition (Table 3; $3.71993e + 41$). There are $3.96903e + 55$ ways of unsuccessful computation, which translates to a $9.3 \times 10^{-13}\%$ probability of (the settability relations behind) this parameter being successfully computed in the first try. If a parameter has this probability of successful computation, the expected number of unsuccessful computations before a successful one occurs is:

$$E = \frac{1-p}{p} = \frac{1 - 0.0000000000000093723978 25}{0.000000000000009372397825} = 1.06696e + 14$$

In other words, it is expected that more than 106 trillion unsuccessful computations will occur before a successful computation takes place.

To put the obtained results in comparison, we performed a second analysis using a different pool of data. Ceolin et al. (2021) present an expanded network that consists of 94 parameters from the nominal domain, covering 58 languages from 15 language families (Figure 6).



	Label	Parameter	Implication(s)	Label	RGS	Sal	It	Sp	Nis	Fr	Ptg	Rm	SaG	CGB	Grk	CyG	E	Du	Afk	D	Da	Ice	FO	Nor	Blg	BlC	SC	Slo	Po	Rus	Ir	Wel	Ma	Hi	Pas	Ta	Te	Man	Can	Jap	Kor	Ar	Heb	Hu	Kh	Est	Fin	mM	Ud	Yu	EvA	EvB	Ek	Ya	Uz	Kaz	Kyr	Tur	Bur	cB	wB	Wo	Mal	Arc	Lak	Label	
1	FGM	± grammaticalized morphology		FGM	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	-	-	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	FGM	1
2	FGA	± grammaticalized agreement	+FGM	FGA	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	0	0	-	-	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	-	+	+	FGA	2
3	FGK	± grammaticalized Case	+FGM	FGK	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	0	0	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	-	+	+	+	FGK	3
4	SPK	± grammaticalized (ultra-)spatial Cases	+FGK	SPK	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	-	-	-	-	+	-	+	+	+	+	-	+	+	+	-	-	-	-	-	-	-	-	0	-	-	-	SPK	4
5	FGP	± grammaticalized person	+FGM	FGP	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	0	0	-	-	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	FGP	5
6	FSP	± semantic person	¬+FGP	FSP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+	+	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	FSP	6
7	FGN	± grammaticalized number	+FGP	FGN	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	0	0	0	0	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	FGN	7
8	SCO	± spread collective number	+FGM, ¬+FGN	SCO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SCO	8
9	GDP	± grammaticalized distributive plurality	+FGM, ¬+FGN	GDP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	GDP	9
10	FSN	± number spread to N	+FGN	FSN	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	0	0	0	0	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	-	-	-	-	+	+	FSN	10
11	FNN	± number on N	+FSN	FNN	+	+	+	+	-	-	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	0	0	0	0	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	0	0	0	0	+	+	FNN	11
12	FGT	± grammaticalized temporality		FGT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-	-	-	FGT	12
13	FGG	± grammaticalized gender	+FGN	FGG	+	+	+	+	+	+	+	+	+	+	+	+	-	+	-	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	0	0	0	0	+	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-	+	+	FGG	13
14	FSG	± semantic gender	+FGN	FSG	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	0	0	0	0	+	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-	-	+	+	FSG	14
15	CGB	± unbounded sg N		CGB	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	+	-	+	+	+	+	-	-	-	-	+	+	-	-	+	+	+	+	+	+	+	+	+	+	+	+	-	-	+	+	+	+	CGB	15
16	FPC	± grammaticalized perception		FPC	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	FPC	16
17	DGR	± grammaticalized Specified Quantity	-FPC, +FGN	DGR	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	-	-	-	-	+	+	-	-	-	-	-	0	0	0	0	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-	-	-	DGR	17
18	DGP	± grammaticalized text anaphora	¬+DGR	DGP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-	0	0	-	-	-	-	-	-	-	-	-	0	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	+	+	+	DGP	18
19	CGR	± weak Specified Quantity	-CGB, +DGR	CGR	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-	-	+	+	0	0	0	0	+	+	0	0	0	0	0	0	0	0	0	+	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CGR	19
20	NWD	± weak person	+FGP, -FSN or +DGR	NWD	-	-	-	-	-	-	-	-	-	-	-	-	+	+	+	+	+	+	+	+	-	-	0	0	0	0	+	+	0	0	0	0	0	0	0	0	0	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	+	-	0	0	NWD	20
21	DGD	± grammaticalized distality	-FSN or +DGR	DGD	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	-	-	0	0	0	0	0	0	0	0	0	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	+	+	-	0	0	DGD	21
22	DPQ	± free null partitive Q	+FNN, -CGB	DPQ	-	-	-	-	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	-	0	0	0	0	0	0	-	-	0	0	+	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DPQ	22
23	DCN	± article-checking N	-FSN or +DGR	DCN	-	-	-	-	-	-	-	+	-	-	-	-	-	-	-	-	+	+	+	+	+	+	0	0	0	0	-	-	0	0	0	0	0	0	0	0	0	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-	0	0	DCN	23
24	DNN	± null-N-licensing art	-DCN	DNN	-	-	-	+	-	-	+	0	-	-	-	-	-	+	+	+	0	0	0	0	0	0	0	0	0	0	-	-	0	0	0	0	0	0	0	0	0	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+	+	-	-	0	0	DNN	24
25	DIN	± D-controlled inflection on N	+FSN	DIN	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	-	-	DIN	25
26	FGC	± grammaticalized classifier	¬+FGN	FGC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+	+	+	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	FGC	26
27	FGE	± general classifier	-FGM, +FGC	FGE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	FGE	27
28	FCN	± person spread to predicate nouns	+FGP	FCN	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	+	0	0	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	FCN	28
29	HMP	± NP-heading modifier		HMP	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	HMP	29
30	ARR	± free reduced relatives		ARR	+	+	+	+	+	+	+	+	+	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	+	+	-	-	+	+	+	+	-	-	-	-	-	-	-	-	+	+	+	+	+	+	+	+	+	+	-	-	+	-	-	-	ARR	30
31	GCN	± head-marking		GCN	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	+	+	+	-	+	+	+	+	+	+	+	+	+	+	+	+	+	-	-	+	+	-	-	GCN	31
32	GFN	± Person controlled marking	+FGP, +GCN	GFN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	+	+	0	+	+	+	+	+	+	+	+	+	+	+	+	+	0	0	-	+	0	0	GFN	32
33	GFP	± agreement with all pronouns	+GFN	GFP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+	+	0	+	+	+	-	+	+	+	+	+	+	+	+	+	0	0	0	+	0	0	GFP	33
34	GP3	± agreement with all 3rdPers DPs	+GFP	GP3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+	+	0	-	+	+	0	+	+	+	+	+	+	+	+	-	0	0	0	+	0	0	GP3	34
35	CSE	± full c-selection		CSE	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	-	-	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	CSE	35
36	EAL	± ergative alignment	+FGK, +CSE	EAL	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	-	-	-	-	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	+	+	EAL	36
37	CAL	± clausal alignment	+FGK, +CSE, ¬+GP3,¬+EAL	CAL	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	+	0	0	-	-	-	+	0	0	-	-	0	0	-	0	0	0	0	0	0	0	0	+	-	-	0	0	0	0	CAL	37
38	LKA	± argument linker		LKA	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-	-	-	LKA	38
39	LKP	± predicative linker		LKP	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	+	-	-	-	-	-	-	-	-	-	-	+	-	-	-	-	-	-	-	-	-	-	-	+	-	-	-	LKP	39
40	LKO	± oblique linker	-LKA	LKO	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	-	-	-	-	-	-	-	-	+	-	-	-	-	-	-	-	-	-	+	+	0	-	-	-	LKO	40
41	DMP	± def matching pronominal possessives	+DCN	DMP	0	0	0	0	0	0	0	+	0	0	0	0	0	0	0	0	-	+	-	+	+	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DMP	41
42	DMG	± def matching genitives	+DMP	DMG	0	0	0	0	0	0	0	+	0	0	0	0	0	0	0	0	0	-	0	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DMG	42
43	GUN	± uniform genitive	+FGK, (-GCN or (+GFP, -GP3)), -CAL, -LKA	GUN	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	+	+	0	0	0	0	0	0	0	0	0	0	0	0	-	-	0	0	0	0	GUN	43
44	GAD	± free Gen	-LKA, ¬+GUN	GAD	+	+	+	+	+	+	+	+	-	-	-	-	+	+	+	+	+	+	+	+	+	+	-	-	-	-	+	+	+	+	+	+	+	0	0	0	0	+	+	-	-	0	0	-	+	+	-	-	-	-	-	-	-	-	+	+	+	0	-	-	-	GAD	44
45	GFO	± GenO	(-GCN or +GFN), ¬+GP3, ¬+EAL, ¬+GUN	GFO	-	-	-	-	-	-	-	-	+	+	+	+	-	-	-	+	-	+	-	-	-	-	+	+	+	+	+	+	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	-	-	-	0	0	0	0	GFO	45
46	PGO	± partial GenO	-GFO	PGO	-	-	-	-	?	-	-	-	0	0	0	0	-	-	-	0	-	0	-	-	-	-	0	0	0	0	0	0	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	-	-	-	0	0	0	0	PGO	46
47	GGS	± generalized GenS	+FGP, -CGR, +NWD, ¬+GFP, ¬+GUN	GGS	0	0	0	0	0	0	0	0	0	0	0	0	+	-	-	-	+	0	+	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	GGS	47
48	GSI	± grammaticalized inalienability		GSI	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	GSI	48
49	ALP	± alienable possession	-GSI	ALP	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-	ALP	49
50	GEI	± genitive inversion	+GP3	GEI	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	0	0	-	-	0	-	-	-	-	-	-	-	-	0	0	0	0	-	0	0	GEI	50
51	GIT	± genitive-licensing iteration		GIT	-	-	-	-	-	-	-	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	GIT	51
52	UST	± unstructured modifiers	+ARR	UST	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	0	0	-	-	+	+	0	0	0	0	0	0	0	0	+	-	-	-	-	+	-	-	-	-	0	0	-	0	0	0	UST	52
53	GPC	± gender polarity cardinals	+FGG	GPC	-	-	-	-	-	-	-	-	-	-	-	-	0	-	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	+	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	-	-	GPC	53
54	PSC	± plural spread from cardinal quantifiers	+FSN, ¬+UST, ¬+GPC	PSC	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	-	-	+	+	+	+	+	0	0	0	0	0	0	-	-	-	-	-	-	0	+	+	+	-	0	-	-	-	-	0	0	0	0	-	-	PSC	54
55	PCA	± plural spread through cardinal adjectives	-PSC	PCA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-	-	-	0	0	0	0	-	0	-	-	-	-	0	0	0	0	-	-	PCA	55
56	PMN	± person marking on numerals	+GFP	PMN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	0	-	+	+	0	-	-	-	-	-	-	-	-	-	0	0	0	-	0	0	PMN	56
57	RHM	± head marking on relative clauses	+FGP	RHM	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	-	-	+	+	-	-	-	-	-	-	-	-	+	-	-	-	-	-	-	-	-	-	-	-	RHM	57
58	FRC	± finite relative clauses		FRC	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	-	-	+	+	+	+	+	+	+	+	+	+	+	+	-	-	-	-	-	-	-	-	-	-	+	+	+	+	-	-	FRC	58
59	NRC	± participial relative clauses	+FRC	NRC	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-	+	0	0	-	-	+	-	-	-	-	+	-	+	+	+	0	0	0	0	0	0	0	0	0	0	+	+	-	+	0	0	NRC	59
60	DOR	± def on relatives	+DGR, +FRC	DOR	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	-	-	0	0	0	0	0	0	0	0	0	+	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+	0	0	0	DOR	60
61	FFP	± feature spread to particles	+FGN, (+LKA or +LKP or +LKO or +GAD), ¬+GFP	FFP	-	-	-	-	-	-	-	-	0	0	0	0	-	-	-	-	-	-	-	-	-	-	0	0	0	0	-	-	+	+	-	-	-	0	0	0	0	-	-	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	-	-	+	0	0	0	FFP	61
62	NUP	± NP under non-genitive arguments	+FGP, (+CSE or +LKA or +LKO)	NUP	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	+	-	+	+	0	0	0	0	-	-	0	0	-	-	+	+	+	+	+	+	+	+	+	+	+	+	+	+	-	-	+	+	NUP	62
63	PNP	± complement under P	+FGP, -CSE or -NUP	PNP	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	0	0	-	0	0	0	0	0	0	+	+	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+	+	0	0	PNP	63
64	NUD	± NP under D	+FGP	NUD	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	0	0	0	0	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	-	-	-	+	+	+	NUD	64
65	NUC	± N under cardinals	¬+UST, +PNP, +NUD	NUC	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	0	0	0	0	0	0	0	0	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	NUC	65
66	NM1	± N under M1 As	+NUC	NM1	-	-	+	+	+	+	+	+	-	-	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NM1	66
67	NM2	± N under M2 As	+NM1	NM2	0	0	-	-	-	-	-	-	0	0	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NM2	67
68	NUA	± N under As	+NM2	NUA	0	0	0	0	0	0	0	0	0	0	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NUA	68
69	NGO	± N under GenO	(+FGP, +UST) or +NUA, (+GUN or +GFO or +PGO)	NGO	0	0	0	0	0	0	0	0	0	0	-	-	0	0	0	-	0	-	0	0	0	0	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NGO	69
70	EAF	± fronted high As	-NM1	EAF	-	-	0	0	0	0	0	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EAF	70
71	ACM	± class MOD	-ARR, -NGO	ACM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	-	0	0	0	0	-	-	+	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ACM	71
72	DSN	± def spread to N	+DCN	DSN	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	-	-	+	+	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DSN	72
73	DSA	± def spread to ARR	+DGR, +ARR	DSA	-	-	-	-	-	-	-	-	-	-	+	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	DSA	73
74	DSS	± def spread to structural categories	+DGR, (-ARR or +DSA)	DSS	0	0	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-	-	-	-	+	0	0	0	0	-	-	0	0	0	0	0	0	0	0	0	+	+	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DSS	74
75	DOC	± def on cardinals	-NWD, +DCN, +NUC	DOC	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	+	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DOC	75
76	NEX	± Proper names in D	(-FSN or -CGR), -NWD, ¬+NUA	NEX	+	+	+	+	+	+	+	+	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+	+	0	-	0	0	NEX	76
77	PEX	± Personal proper names in D	+NEX	PEX	+	-	+	+	+	+	+	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+	+	0	0	0	0	PEX	77
78	FEX	± Partial personal proper names in D	+PEX	FEX	+	0	+	+	+	+	+	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+	+	0	0	0	0	FEX	78
79	PDC	± D-checking possessives	+DGR, (¬-CGR or -NWD), ¬+GFP	PDC	-	-	-	+	-	+	+	-	-	-	-	-	0	0	0	0	0	+	0	0	-	-	0	0	0	0	+	+	0	0	0	0	0	0	0	0	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	PDC	79
80	PCL	± clitic possessives	+FGP, ¬+GFP, ¬+DMP, ¬+UST, (-PDC or ¬+DGR)	PCL	-	-	-	0	-	0	0	0	+	+	+	+	0	0	0	0	0	0	0	0	0	0	-	-	-	-	0	0	-	-	+	-	-	0	0	0	0	-	-	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	+	0	-	-	PCL	80
81	APO	± adjectival possessives	¬+GFP, ¬+UST	APO	+	+	+	+	+	-	+	-	-	-	-	-	-	-	-	-	-	-	-	-	+	+	+	+	+	+	-	-	-	-	-	-	-	-	-	0	0	-	-	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	0	-	-	APO	81
82	WAP	± Wackernagel possessives	¬+DMP, +NUD, -PDC, -APO or (-NM1, +APO)	WAP	+	-	0	0	0	0	0	0	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WAP	82
83	AGE	± adjectival genitive	+APO	AGE	-	-	-	-	-	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	+	+	+	+	-	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	AGE	83
84	OPK	± null possessive licensing article with kinship nouns	+DGR, -GSI	OPK	-	-	+	-	-	-	-	+	-	-	-	-	-	-	-	-	+	+	+	+	-	-	0	0	0	0	-	-	0	0	0	0	0	0	0	0	0	+	+	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	OPK	84
85	TSP	± split deictic demonstratives	-FSN or +DGR	TSP	-	-	-	-	-	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	-	-	0	0	0	0	0	0	0	0	0	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	+	0	0	TSP	85
86	TDP	± split demonstratives	+TSP	TDP	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	TDP	86
87	TDC	± D-checking demonstratives	-TSP	TDC	+	+	+	+	+	0	+	+	+	+	-	-	+	+	+	+	+	+	+	+	+	-	0	0	0	0	-	-	0	0	0	0	0	0	0	0	0	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+	-	+	0	0	0	TDC	87
88	TSA	± structured demonstratives (adjectival)	¬+UST, ¬+TSP, ((+DGR, +NM1) or (-ARR, -NM1) or -NUC)	TSA	0	0	-	-	-	0	-	+	0	0	-	-	-	-	-	-	-	-	-	-	+	+	0	0	0	0	+	+	0	0	0	0	0	0	0	0	0	+	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TSA	88
89	TAR	± unstructured demonstratives	+ARR, ¬+TSP	TAR	-	-	-	+	-	0	-	-	-	-	+	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+	+	+	0	0	+	+	+	+	0	0	0	0	0	0	0	0	-	+	+	+	+	+	+	+	+	-	0	0	-	0	0	0	TAR	89
90	TLC	± Dem fronting to Loc	¬+TSP, ¬+TDC, (+TSA or (+PNP, +TAR))	TLC	0	0	0	0	0	0	0	0	0	0	+	+	0	0	0	0	0	0	0	0	0	+	0	0	0	0	-	-	0	0	0	0	0	0	0	0	0	+	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TLC	90
91	TND	± long distance D-checking demonstratives	+CGR, (+TSA or +TAR)	TND	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	0	0	0	0	-	-	0	0	0	0	0	0	0	0	0	-	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TND	91
92	TDA	± split def on adjectival demonstratives	(+DSS or +DSA), (+TSA or +TAR)	TDA	0	0	0	0	0	0	0	0	0	0	-	-	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TDA	92
93	TNL	± DP under Loc	+TSP or +TLC or (¬+TSP, ¬+TDC, ¬+TSA, ¬+TAR)	TNL	0	0	0	0	0	-	0	0	0	0	+	+	0	0	0	0	0	0	0	0	0	+	+	+	+	+	0	0	0	0	0	+	+	0	0	0	0	+	0	+	+	+	+	+	+	+	0	0	0	0	0	0	0	0	+	0	+	0	-	+	+	TNL	93
94	FVP	± variable person	+FAG, -NWD	FVP	+	-	-	+	?	-	-	+	-	+	+	+	0	0	0	0	0	0	0	0	+	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+	+	0	0	0	0	FVP	94
					RGS	Sal	It	Sp	Nis	Fr	Ptg	Rm	SaG	CGB	Grk	CyG	E	Du	Afk	D	Da	Ice	FO	Nor	Blg	BlC	SC	Slo	Po	Rus	Ir	Wel	Ma	Hi	Pas	Ta	Te	Man	Can	Jap	Kor	Ar	Heb	Hu	Kh	Est	Fin	mM	Ud	Yu	EvA	EvB	Ek	Ya	Uz	Kaz	Kyr	Tur	Bur	cB	wB	Wo	Mal	Arc	Lak	Label	

Figure 6. ***The Ceolin et al. network consists of 94 parameters from the nominal domain across 58 languages (Ceolin et al., 2021). The column 'Implication(s)' presents the settability path(s) on which each parameter is settable.***[1]

[1] For the purposes of our analysis, part of the logical expression behind the last parameter (i.e., FVP) was changed from FAG to FGA, following personal communication with Cristina Guardiano.

This network involves 82 dependent parameters, the settability of which depends on the setting of other parameters. Of these, 25 parameters are settable on 3 or more paths, and these are the ones we analyzed. Specifically, we converted the dependencies given in the 'Implication(s)' column (Figure 6) into mathematical expressions in the following way. If a dependency involves two parameters linked by ',' (i.e., ∧), both parameters must form part of every settability path behind this parameter, such that following Boolean logic this was expressed as a multiplication. If a dependency involves two parameters linked by 'OR' (i.e., ∨), each of the two parameters corresponds to a different way of reaching settability, so this was expressed as an addition. Example (4) illustrates the mathematical expression of a hypothetical example that has a structure that is found in the analyzed network (i.e., parameter 20, label: NWD, Figure 6).

(4) Parameter A = +B, +C or -D ⇔ 1 x (1+1) = 2 paths

In (4), the assumption is that parameters B, C, and D involve one settability path each. When this is not the case, the number of paths behind each parameter must be entered.

Table 4 presents the results of the mathematical expression of the relevant parameters in terms of settability paths as well as their probability of successful computation. For the latter, the Python program was used to perform the calculations.

Table 4. ***Dependent parameters with 3 or more settability paths and their (conditional) probability of successful computation (Ceolin et al. network).***

Parameter	Mathematical expression of the dependencies	Ways of computation without loop	Ways of computation with one loop	Prob. of successful computation	Conditional prob. of successful computation
P45 - 4 paths	(1 + 1) x 1 x 1 x 2	24	60	28.5%	28.5%
P46 - 4 paths	4	24	60	28.5%	8.16%
P47 - 4 paths	1 x 1 x 2 x 1 x 2	24	60	28.5%	2.33%
P61 - 5 paths	1 x (1 + 1 + 1 + 2) x 1	120	500	19.3%	0.45%
P62 - 3 paths	1 x (1 + 1 + 1)	6	6	50%	0.22%
P63 - 4 paths	1 x (1 + 3)	24	60	28.5%	0.064%
P65 - 4 paths	1 x 4 x 1	24	60	28.5%	0.018%
P66 - 4 paths	4	24	60	28.5%	0.0052%

P67 - 4 paths	4	24	60	28.5%	0.0015%
P68 - 4 paths	4	24	60	28.5%	0.0004%
P69 - 41 paths	(1 x 1) + 4 x (2 + 4 + 4)	3.34525e+49	2.2083E+51	1.4%	0.0000064%
P70 - 4 paths	4	24	60	28.5%	0.0000018%
P71 - 41 paths	1 x 41	3.34525e+49	2.2083e+51	1.4%	0.00000002%
P75 - 16 paths	2 x 2 x 4	2.09228e+13	4.82395e+14	4.1%	0.0000000011%
P76 - 16 paths	(1 + 1) x 2 x 4	2.09228e+13	4.82395e+14	4.1%	0.000000000047%
P77 - 16 paths	16	2.09228e+13	4.82395e+14	4.1%	0.0000000000019%
P78 - 16 paths	16	2.09228e+13	4.82395e+14	4.1%	0.00000000000008%
P79 - 3 paths	1 x (1 + 2) x 1	6	6	50%	0.00000000000004%
P80 - 8 paths	1 x 1 x 2 x 1 (3 + 1)	40320	375368	9.7%	0.0000000000000032%
P82 - 10 paths	2 x 1 x 3 x 1 + (4 x 1)	3628800	46253610	7.2%	0.000000000000000287%
P88 - 12 paths	1 x 1 x [(1 x 4) + (1 x 4) + 4]	479001600	7751595852	5.8%	0.0000000000000000167%
P90 - 13 paths	1 x 1 x (12 + (1 x 1))	6227020800	1.11471e+11	5.2%	0.00000000000000000088%
P91 - 13 paths	1 x (12 + 1)	6227020800	1.11471e+11	5.2%	0.00000000000000000004%
P92 - 39 paths	(2 + 1) x (12 + 1)	2.03979e+46	1.27643e+48	1.5%	0.0000000000000000000007%
P93 - 26 paths	1 + 13 + (1 x 1 x 12 x 1)	4.03291e+26	1.62279e+28	2.4%	0.00000000000000000000002%

As Table 4 suggests, two parameters in the analyzed network have 41 settability paths each. Repeating the analysis presented above for the Longobardi & Guardiano network (i.e., removing the one-loop restriction, but limiting the path-selection events to number of paths), a parameter with 41 settability paths has a total of $\mathrm{n} \cdot (\mathrm{n}-1)^{\mathrm{n}-1} = 4.95660e + 65$ ways of computation. Subtracting the number of computations that do not involve a repetition (Table 4; $3.34525e + 49$), there are $4.95659e + 65$ ways of unsuccessful computation, which translates to a $6.7 \times 10^{-15}\%$ probability of (the settability relations behind) this parameter being successfully computed in the first try. Thus, the expected number of unsuccessful computations before a successful one occurs is:

$$E = \frac{1 - p}{p} = \frac{1 - 0.0000000000000067491}{0.0000000000000067491} = 14816817458844600$$

Succinctly put, it is expected that more than 14 quadrillion unsuccessful computations will occur before a successful one takes place.

Discussion

We have presented a previously unanalyzed aspect of the P&P approach that seems to entail an unrealistically cumbersome computational burden. We stress here that our report does not in principle repudiate the basic notion of parameters as emergent points of variation that build on innate principles, but rather the more specific conjecture that the infant is presented with an extensive predefined list of such parameters.

Parameters were proposed as a cognitive primitive that help organize and constrain the hypothesis space of a child trying to acquire language in an efficient way (Pearl & Lidz, 2013). Although the notion of parametric variation is theoretically well-formed and useful as a concept, previous research on the computation of parametric models of language acquisition has revealed various computability issues. For instance, it was found that the child would need to set about 30 parameters per second, throughout childhood, to assimilate a parametric model, with obvious consequences about computability (Levelt, 1974; Fitch & Friederici, 2012).

Other work on grammar learning revealed the *local maxima problem*: a learner may posit incorrect hypotheses about the target grammar G_t, forming a grammar G_s from which she can never move out, similar to an absorbing state in the theory of Markov chains (Gibson & Wexler, 1994). Related to this, the *learnability problem* refers to the fact that even if a path from G_s to G_t exists and there are salient cues that guide the learner towards the target, there is a high probability that the learner does not take this path, resulting in non-learnability (Niyogi & Berwick, 1996).

The *problem of low probability of unambiguous input* does not, strictly speaking, raise learnability concerns, but it does raise computability issues. According to this problem, given the scarcity of unambiguous input (i.e., there is no one-to-one correspondence between the surface properties of the input and the correct parameter values that generate G_t), the learning algorithm must wait for a sentence that is fully unambiguous before forming any G_s, yet these sentences have a very low probability of occurring (Sakas, 2000). Further, the notion of an unambiguous linguistic input also presupposes a robust and complex metacognitive, inferential state for the infant.

All these problems raise concerns that relate to forming hypotheses about a G_t in the process of parameter-setting, and not to determining settability. This means that they are problems that pertain not to the parametric model itself, but to the interaction between the input and the learner, and as such, they can be ameliorated under the right conditions. For example, the local maxima problem can be solved if the learner can change more than one parameter setting when encountering input that is not predicted by G_s (Niyogi & Berwick, 1996). Similarly, the problem of low probability of unambiguous input has been sidestepped by suggesting that some sentences in the input function as signatures or unambiguous triggers; that is, they are analyzable only if the learner has selected the correct value for a parameter (Fodor, 1998; Yang, 2002). Focusing on setting relations, the conclusion is that under certain assumptions, parameter-setting *is* computable (Sakas et al., 2017). However, this state of affairs does not take into account the computability of settability relations.

Unlike problems of setting, problems of settability are *intrinsic* to the parametric model. To give a concrete example, the fact that one of the parameters analyzed in the previous section was found to have $3.96903e + 55$ ways of unsuccessful computation, even when restricting the possible number of loops to not exceed the number of possible path-selection events, is not a problem that the learner can overcome by using some particular learning strategy instead of another. No matter the strategy, the fact will remain that before one finds a way of checking these 36 settability paths without running into a loop, trillions of unsuccessful computations are expected to take place. Even under the unrealistic assumption that the child devotes only one second to each computation, execution would take 29,637,856,071 hours, or over 3 million years. This corresponds to the task of computing the settability relations behind a *single* parameter. It seems highly implausible that this amount of computation is entered into the task carried out by the child when acquiring language. To put the number in perspective, the discovery of the Ledi jaw that was recently added to the fossil record of the genus Homo places the earliest occurrence of recognizable Homo to 2.8 mya (Villmoare et al., 2015).

It may, of course, be possible for a deep learning approach to settability to reduce our large estimate of unsuccessful computations, in combination with external learning heuristics (of the kind we will discuss below). However, to our knowledge no such approach has been forthcoming in the literature, and in any event, it would likely necessitate a number of complex priors that may simply re-migrate settability difficulties to postulated AI algorithms that may have no cognitively plausible, implementational correlate (Marcus & Davis, 2021). The burden of proof in this respect lies with deep learning (and related) approaches, and we therefore leave this possibility to future research, in particular given that our approach here has been explicitly to model the computability of settability paths.

The results presented in the previous section demonstrate various problems. First, the memoryless parser cannot keep track of all the loops. Even if we endow it with this ability, the number of unsuccessful computations that run into a loop is in the thousands, and this is the case for parameters that have just 6 settability paths. Restricting the number of loops does not make the task feasible either. Importantly, the parameters that were analyzed represent only one domain of grammar: the nominal domain. One can imagine how much larger the task would be if more parameters are brought into the picture. In addition, setting these non-nominal parameters would also rely on a number of complex, higher-order semantic and conceptual networks, whose developmental trajectory remains relatively elusive (Murphy, 2017). Second, the results suggest that a parametric approach to Universal Grammar is not feasible. Crucially, the results do not provide any kind of evidence against Universal Grammar itself, which remains a robust and necessary concept in some frameworks of language acquisition. The identified problems arise when one suggests that the grammatical relations described as parameters exist in the form of *interlocked primitives* in Universal Grammar. This entails that the results are also not informative about the grammatical properties that are described in the analyzed network: The parameters in the Longobardi & Guardiano and Ceolin et al. networks are correct in the sense that they faithfully represent some differences in the grammars of various languages. Both networks, beyond descriptive and typological evidence, are strongly supported by their phylogenetically plausible conclusions. Our results are informative about the computability of a key characteristic of parametric models: settability. This characteristic is the cornerstone of almost all parametric models of language acquisition, because it provides the answer to the logical problem of language acquisition. Parameters are meant to be understood as a built-in shortcut that aids acquisition (Pearl & Lidz, 2013), but this only happens when they are conceived as *interlocked* parameters, meaning that the setting of one parameter carries implications about the settability of others. If parameters were to be understood as millions of unrelated points of variation, the variation space would not be organized in specific ways, hence would not be an aid in acquisition.

These results challenge another long-standing assumption of parametric models: the *instantaneous* nature of acquisition. Chomsky introduced this metaphor with the aim of talking about an idealized version of development, one that abstracts away from specific stages, on the assumption that these stages are largely uniform and have no impact on the acquired grammar (Chomsky, 1975). Some research since then has proposed that this idealization can be treated as a viable research avenue for the topic of language acquisition (Cinque, 1989; Rizzi, 2000). The problem arises when the 'instantaneous acquisition' metaphor presupposes a Universal Grammar that is rich enough to justify the concept of rapid setting of innate primitives. In other words, the 'instantaneous acquisition' narrative *relies* on the existence of a structurally rich Universal Grammar that involves detailed parametric networks like the one analyzed here. Even if acquisition was instantaneous in the sense that the value of a parameter would be determined automatically without any of the parsing reported in acquisition models, the settability relations behind the dependent parameters would still need to be computed in a stepwise fashion. Unless a learner can perform some trillions of computations in an instant, acquisition cannot be viewed as an instantaneous process.

It is also important to note that the obtained results are informative about any given parametric model that postulates interlocked parameters. One may think that the multiple paths to the settability of a parameter in the two analyzed networks are an artifact of these specific networks, such that the settability problem would vanish if another network was examined. There are two reasons to believe that the opposite is true. First, the grammatical relations behind the parameters in the two networks are correct and their faithful representation of cross-linguistic differences has never been challenged. Second, the neat binary branching of Figure 1 *is* an artifact of presentation. More specifically, it is an artifact of choosing some 'big' macroparameters and a few languages, oversimplifying and ignoring many intermediate points of variation. For example, some languages have both partial polysynthesis and null subjects, which is a combination Figure 1 does not permit. This possibility cannot be captured without adding more parametric nodes in the hierarchy. Once these nodes are added, Figure 1 will resemble the two analyzed networks. Overall, the obtained results confirm Chomsky's early disclaimer about instantaneous acquisition. In his words, the 'instantaneous acquisition' model "is surely false in detail, but can very well be accepted as a reasonable first approximation" (Chomsky 1967: 441-442).

In relation to the computability concerns our analyses raise, a reviewer notes that the formalization of the cross-parametric implications currently adopted in the networks represented in Figures 2 and 6 is not assumed to reproduce or simulate any learning process, and it is not based on any consideration concerning the potential computational effort made by the learner in processing this type of information. Thus, the possibility cannot be excluded that a different formalization of the same implicational network might produce different outputs that could also affect the settability relations we used in our analyses. Although this is true, the parametric inventories we analyzed are firmly grounded on solid descriptive, typological, and phylogenetic evidence (Crisma et al. 2020, Ceolin et al. 2021). As such, determining their computability is important. Naturally, if in future work the implicational network is altered specifically in order to be made computable/learnable, the observed computability concerns will be circumvented. Based on current knowledge, however, the fact remains that two examples of our best parametric inventories raise specific computability concerns at their present state of development.

These concerns beg two important questions about the scope of our results. A reviewer asks what would go wrong if the learner ignores the implicational network and just tries to opportunistically set parameters whenever possible. Relatedly, is it possible that our results do not raise computability concerns for P&P in general, but for one particular instantiation of a P&P model that involves a predefined list of options in the initial state of development? The answer to the first question is that the implicational network provides innate shortcuts that aid acquisition. Asking whether the learner could ignore it would be tantamount to asking whether we can ignore any other innate aspect of our biological make-up. More importantly, however, the learner has no reason to ignore it, because this implicational network is the glue that keeps together the parametric space. If we remove the glue, the learner is left to navigate an extremely large variation space without any shortcuts. This also answers the second question. As mentioned already, our results do not speak about Universal Grammar or the principles of P&P, hence it would be wrong to conclude that we cast doubt on P&P as a whole. We examined a specific aspect of its parametric component. In this context, the answer to the second question is that if we remove the implicational network from the picture, the computability issues we raised may be indeed sidestepped. However, this does not entail that we are left with a parametric model that is free from computability concerns. In the absence of implicational relations, the learner faces the task of navigating an extremely large space of variation. It has been suggested that this large space of variation "brings to light a fatal weakness of the microparametric approach" (Huang & Roberts 2016: 321): Even as few as a hundred independent parameters would raise serious concerns about the realization of only a very small fragment of the set of possible grammars during the entire human history (Huang & Roberts 2016). In a nutshell, removing the implicational network from the picture possibly alleviates the computability problems we raised, but makes the model vulnerable to other issues. Of course, it is entirely possible that parametric models that do not suffer from any type of computability issues are developed in the future. At present, the most promising candidates are those that refer to emergent parametric hierarchies (Huang & Roberts, 2016; Biberauer, 2019). Once these proposals are developed in sufficient technical detail and mapped to cross-linguistic data, future studies that assess their computability will be possible.

Having shown that the process of grammar development does not correspond to fixing values of innate parameters, the question of how the child sets its target grammar becomes again relevant. Merging insights from different acquisition models (Yang, 2002; Chistiansen et al. 2009; Boeckx & Leivada, 2014; Fasanella, 2014; Westergaard, 2014; Yang et al., 2017; Chomsky, 2019), Figure 7 presents a sequence of seven processes that explain how the child extrapolates rules of grammar from the input. The aim here is to provide a detailed, biologically plausible account for this task, while assuming as few Universal Grammar-/language-specific primitives as possible. Figure 7 lists the tasks that the efficient learner has to perform in order to arrive at a target grammar G_t.

We will briefly describe the principles of computation that aid the learner in each of these tasks, as well as their neurobiological basis, effectively presenting the process of acquiring a G_t without resorting to postulating parameters. Importantly, we illustrate this model not to outline its specific algorithmic architecture, which deviates from the central critique and motivation we adopt here. Instead, we provide a general outline of an architecture that could feasibly be instantiated in a number of ways.

One crucial factor that unlocks the process of developing a G_t is very early prosodic information which helps eliminate logically possible (though unsubstantiated on the basis of the input) learning tracks. Therefore, the first step in the process of cracking the grammar 'code' is input segmentation, whereby the learner breaks a continuous acoustic or visuo-motor signal into a sequence of discrete, meaningless symbols that make up larger meaningful chunks. In order to go from continuous, unsegmented input to discrete elements, the learner must treat the input as meaningful across levels of linguistic analysis (Process 1 in Figure 7).

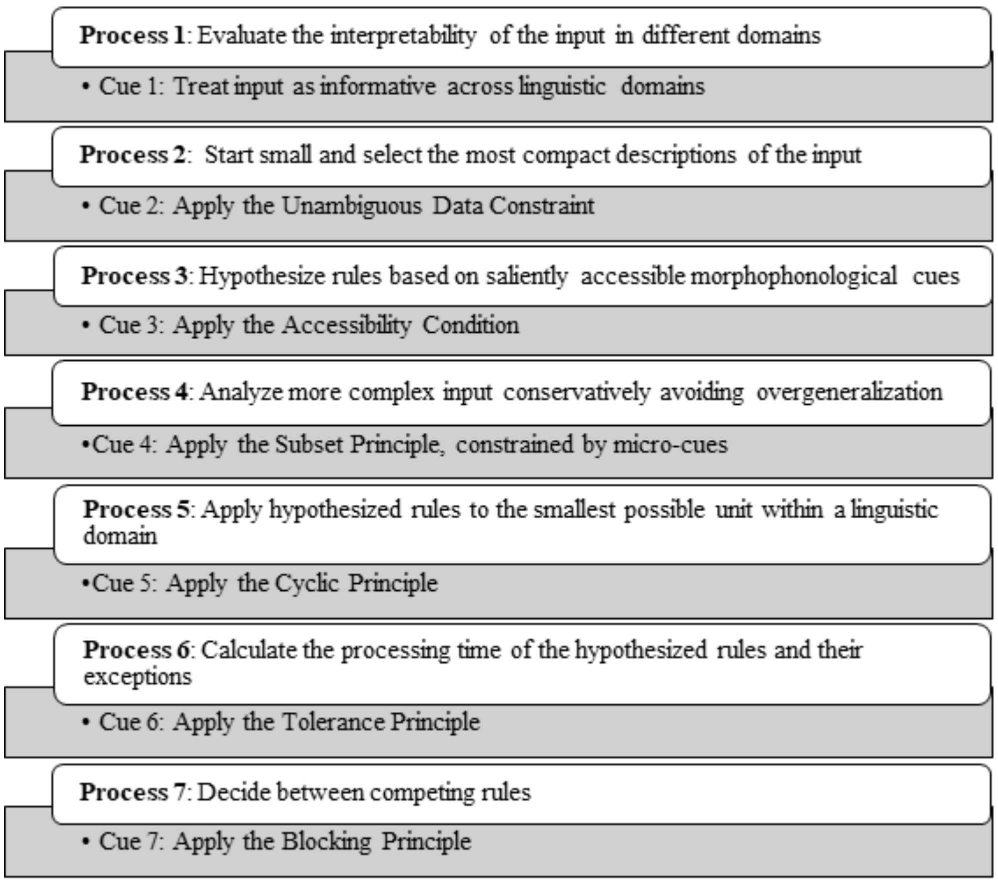

Process 1: Evaluate the interpretability of the input in different domains
• Cue 1: Treat input as informative across linguistic domains
Process 2: Start small and select the most compact descriptions of the input
• Cue 2: Apply the Unambiguous Data Constraint
Process 3: Hypothesize rules based on saliently accessible morphophonological cues
• Cue 3: Apply the Accessibility Condition
Process 4: Analyze more complex input conservatively avoiding overgeneralization
•Cue 4: Apply the Subset Principle, constrained by micro-cues
Process 5: Apply hypothesized rules to the smallest possible unit within a linguistic domain
•Cue 5: Apply the Cyclic Principle
Process 6: Calculate the processing time of the hypothesized rules and their exceptions
• Cue 6: Apply the Tolerance Principle
Process 7: Decide between competing rules
• Cue 7: Apply the Blocking Principle

Figure 7. ***Processes and cognitive cues that are critical in developing a target grammar from the input.***

One crucial factor that unlocks the process of developing a G_t is very early prosodic information which helps eliminate logically possible (though unsubstantiated on the basis of the input) learning tracks. Therefore, the first step in the process of cracking the grammar ‘code’ is input segmentation, whereby the learner breaks a continuous acoustic or visuo-motor signal into a sequence of discrete, meaningless symbols that make up larger meaningful chunks. In order to go from continuous, unsegmented input to discrete elements, the learner must treat the input as meaningful across levels of linguistic analysis (Process 1 in Figure 7). More concretely, the computation progresses from forming statistical observations over phoneme distribution to deciphering word edges, segmenting morphemes, and then determining lexical categories (Christiansen et al., 2009). For spoken languages, the key to this process is the entrainment of the auditory cortex to different aspects of handling the acoustic signal, such as parsing at the syllabic level and integrating various cues while filtering background noise (Ding & Simon, 2014; Benítez-Burraco & Murphy, 2019; Murphy, 2015, 2020). For sign languages, cortical entrainment to the sign envelope is strongest at occipital and parietal regions (Brookshire et al., 2017). After such initial entrainment, endogenous neural activity appears to "take over" and generate inferences about abstract structure, which we assume is the point at which grammatically relevant hypotheses can be made. This modality-independent stimulus-brain coherence underlies the extraction of probabilistic information from the input. Crucially, these processes presuppose a capacity to generate specific lexical categories but also a capacity to represent particular syntactic features that enter into structure-building operations; representations that seem unlike any other symbolic units in the primate world. In carrying out this process, the learner is initially guided by the Unambiguous Data Constraint, which leads them to select and focus on the simplest and cleanest possible data, mainly unambiguous matrix clauses (i.e., Process 2 in Figure 7; Lightfoot, 1991, 2020; Fodor, 1998; Pearl & Weinberg, 2007). This constraint can be viewed as the outcome of two hallmark tendencies of neural organization: the tendency to chunk long sequences and the tendency to organize/compress input in simple ways (Fonollosa et al., 2015; Christiansen & Chater, 2016; Chater & Loewenstein, 2016; Al Roumi et al., 2021). These tendencies are ubiquitous, but differentially manifested in accordance with the individual characteristics of spoken and signed phonology (e.g., single-segment words are rare in spoken languages, but common in sign languages, due to the different chunking strategies involved; Brentari, 1998; Emmorey, 2016). Having selected the relevant input, the learner then analyzes it by hypothesizing rules, based on saliently accessible morphophonological cues (Process 3; Boeckx & Leivada, 2014; Fasanella, 2014). According to the Accessibility Condition, grammatical properties of the G_t are determined by directly inspecting phonological and morphological properties of utterances (Fasanella, 2014). The speaker/signer analyzes an input chunk through hypothesizing a grammar G_i with a probability p_i. Depending on whether G_i matches the input from G_t, G_i is punished or rewarded by decreasing and increasing p_i accordingly (Yang, 2002).

Progressively, the learner tackles more complex input, but does so by avoiding overgeneralizations (Process 4). The Subset Principle guides the learner to generalize as conservatively as possible (Yang et al., 2017). Concerns that have been raised about the computational complexity of the Subset Principle (see Yang, 2016) can be sidestepped through the postulation of emergent (i.e., not innate) micro-cues. As minimal points of syntactic representation, micro-cues anchor the formed hypotheses in narrow domains of application, always on the basis of positive evidence (Westergaard, 2014). This anchoring renders wholesale, computationally costly comparisons of G_i and G_t unnecessary; a notion in line with recent developments in derivational syntactic theory (Chomsky, 2019; Murphy & Shim, 2020). Indeed, one of the implications of our results is that the initial hypothesizing on the part of the child of a large number of conflicting grammars is purely a stipulation from traditional psycholinguistic models, with no grounding in computability concerns. In a similar way that models of syntax no longer typically assume that multiple independent derivational representations of a specific tree are compared during sentence construction (as in early minimalist syntax), so too should language acquisition researchers push computational feasibility (and not competition between G_i and G_t) as a primary constraint on modelling.

Certain generalization tendencies do come into play (e.g., the Input Generalization, a computational bias that suggests that there is a preference for a property of a syntactic head to generalize to other heads, thus giving rise to harmonic patterns; Huang & Roberts, 2016), but they boil down to soft biases that do not translate into extensive overgeneralizations in child language. Their status as soft biases is also evidenced by the fact that they do not translate to absolute typological universals: Phylogenetic modelling has demonstrated that these generalizations are not uniform across language families (Dunn et al., 2011). Research into recently emerged sign languages corroborates this conclusion. There is some evidence for harmonic headedness patterns in the repertoire of first-generation signers of Al-Sayyid Bedouin Sign Language, but variation exists and the preference for one syntactic order over others becomes more stable progressively over different generations of signers (Sander et al., 2005).

Once the learner has hypothesized rules, a cognitive principle that minimizes the domain of application of these rules comes into the picture (Process 5). Similar to how the Subset Principle constrains generalizing across different morphosyntactic environments, the Cyclic Principle constrains the domain of application of the hypothesized rules. According to this principle, when one domain to which a rule can apply is contained in another, the rule applies first to the smaller domain and then proceeds to the wider one (Chomsky, 2019). From a biological perspective, this stepwise cyclical application of rules in grammar is concordant with the overall cyclical nature of auditory and visual perception, which has been linked to dynamic oscillatory activity in the brain (Ho et al., 2017). In addition, these notions seem amenable to ultimately being embedded within a framework of mature syntactic computation that calls upon demands of workspace construction; general resource restrictions on recursive, Markovian computations; limiting access to representational search; and related notions (Chomsky, 2019).

A key component of many acquisition models concerns the process that enables the learner to decide the productivity of a hypothesized rule in light of possible exceptions. The learner must perform some calculation that compares a list of candidates over which a rule applies and a list of exceptions to the rule (Process 6). The Tolerance Principle provides a calculus of the exceptions a learner can tolerate before abandoning a hypothesized rule as unproductive: Assume a rule R is productive over a set of items N only when the number of known exceptions *e* is smaller than the number of N divided by the natural log of N (Yang, 2002; Yang et al., 2017). The Tolerance Principle can also be shown to resolve the acquisition of English dative constructions, a perennial problem in acquisition research (Yang, 2017).

Last, the learner must be able to decide between different productive rules that may apply to the same item (Process 7). The Blocking Principle states that when two rules are available to realize a set of morphophonological values, the more specific one applies (Yang 2002). This ability to inactivate general rules in specific cases (e.g., not apply the regular rule for past tense formation in irregular verbs) provides the list of exceptions that are necessary in the learner's effort to calculate the productivity of a hypothesized rule.

Overall, the list of processes in Figure 7 consists of some landmark cognitive principles that are operative in the process of language growth in the individual. Crucially, it shifts the focus of research to principles of computation, rather than triggered representational primitives. In addition, we have tried to emphasize the limitations on assuming models of idealized observers that choose either optimal or near-optimal hypotheses from an enormous list of explicitly entertained candidate settings. The model does not cover all aspects of acquisition; instead, it has an explicit focus on grammar, leaving other domains (e.g., the lexicon, pragmatics) unaddressed. Its scope is narrowed since our aim has explicitly been to account specifically for the process of cracking the grammar code without assuming innate parameters, in light of the computability problems presented above. Importantly, the program that performed the computations presented does not 'read' the linguistic properties behind the analyzed parameters; it only computes the various permutations between the settability paths behind them. As such, both the program that was used in the analysis of settability relations and the synthesis of cognitive principles that come into play in language acquisition can be embedded in wider contexts (e.g., by using the program to compute settability relations in other parametric models or by expanding the model in Figure 7 to include principles that are relevant in the process of lexical learning), eventually piecing together a more complete and biologically plausible account of the language acquisition process. At a minimum, our framework provides a (putatively) computationally tractable, and (seemingly) psychologically plausible scaffold around which implementational models can be built. We consider the account briefly outlined here to be ripe for future modelling research, in particular with respect to how the notion of computational tractability might map onto the development of general learning biases and computational principles of efficiency. Future research could expand on the list of parameters we have used and make more direct contact with models of cognitive and neural development (Crisma et al., 2020; Ceolin et al., 2020; 2021).

References

Al Roumi, F., Marti, S., Wang, L., Amalric, M., & Dehaene, S. (2021). Mental compression of spatial sequences in human working memory using numerical and geometrical primitives. *Neuron, 109*(16), 2627–2639. doi: https://doi.org/10.1016/j.neuron.2021.06.009

Baker, M. (2003). Linguistic differences and language design. *Trends in Cognitive Sciences, 7,* 349–353. doi: 10.1016/s1364-6613(03)00157-8

Benítez-Burraco, A., & Murphy, E. (2019). Why brain oscillations are improving our understanding of language. *Frontiers in Behavioral Neuroscience, 13,* 190. doi: doi.org/10.3389/fnbeh.2019.00190

Biberauer, T. (2019). Factors 2 and 3: Towards a principled approach. *Catalan Journal of Linguistics, Special Issue,* 45–88. doi: doi.org/10.5565/rev/catjl.219

Boeckx, C., & Leivada, E. (2014). On the particulars of Universal Grammar: Implications for acquisition. *Language Sciences, 46*(B), 189–198. doi: doi.org/10.1016/j.langsci.2014.03.004

Boeckx, C., & Leivada, E. (2013). Entangled parametric hierarchies: Problems for an overspecified Universal Grammar. *PLoS ONE, 8*(9), e72357. doi: doi.org/10.1371/journal.pone.0072357

Brentari, D. (1998). *A prosodic model of sign language phonology*. Cambridge, MA: MIT Press.

Brookshire, G., Lu, J., Nusbaum, H. C., Goldin-Meadow, S., & Casasanto, D. (2017). Visual cortex entrains to sign language. PNAS, *114*(24), 6352–6357. doi: 10.1073/pnas.1620350114

Ceolin, A., Guardiano, C., Irimia, M.-A., & Longobardi, G. (2020). Formal syntax and deep history. *Frontiers in Psychology, 11,* 488871. doi: https://doi.org/10.3389/fpsyg.2020.488871

Ceolin, A., Guardiano, C., Longobardi, G., Irimia, M. A., Bortolussi L., & Sgarro A. (2021). At the boundaries of syntactic prehistory. *Philosophical Transactions of the Royal Society B, 376,* 20200197. doi: https://doi.org/10.1098/rstb.2020.0197

Chater, N., & Loewenstein, G. (2016). The under-appreciated drive for sense-making. *Journal of Economic Behavior & Organization, 126*(B), 137–154. doi: https://doi.org/10.1016/j.jebo.2015.10.016

Chomsky, N. (1965). *Aspects of the theory of syntax*. Cambridge, MA: MIT Press.

Chomsky, N. (1967). The formal nature of language. Appendix to E. Lenneberg's *Biological foundations of language*. New York: John Wiley and Sons.

Chomsky, N. (1975). *Reflections on language*. New York: Pantheon.

Chomsky, N. (1980). *Rules and representations*. New York: Columbia University Press.

Chomsky, N. (1981). Lectures on Government and Binding. Dordrecht: Foris.

Chomsky, N. (2019). Some puzzling foundational issues: The Reading program. *Catalan Journal of Linguistics, Special Issue,* 263–285. doi: doi.org/10.5565/rev/catjl.287

Christiansen, M. H., & Chater, N. (2016). The Now-or-Never bottleneck: A fundamental constraint on language. *Behavioral and Brain Sciences, 39,* e62. doi:

doi.org/10.1017/S0140525X1500031X

Christiansen, M. H., Onnis, L., & Hockema, S. A. (2009). The secret is in the sound: From unsegmented speech to lexical categories. *Developmental Science, 12*(3), 388–395. doi: doi.org/10.1111/j.1467-7687.2009.00824.x

Cinque, G. (1989). Parameter setting in "instantaneous" and real-time acquisition. *Behavioral and Brain Sciences, 12*, 336.

Cowan, N. (2000). The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences, 24*, 87–114. doi: https://doi.org/10.1017/S0140525X01003922

Crisma, P., Guardiano, C., & Longobardi, G. (2020). Syntactic parameters and language learnability. *Studi Saggi Linguistici, 58*, 99–130. doi: https://doi.org/10.4454/ssl.v58i2.265

Ding N., & Simon J. Z. (2014). Cortical entrainment to continuous speech: functional roles and interpretations. *Frontiers in Human Neuroscience, 8*, 311. doi: doi.org/10.3389/fnhum.2014.00311

Dunn, M., Greenhill, S. J., Levinson, S. C., & Gray R. D. (2011). Evolved structure of language shows lineage-specific trends in word-order universals. *Nature, 473*, 79–82. doi: 10.1038/nature09923

Emmorey, K. (2016). Consequences of the Now-or-Never bottleneck for signed versus spoken languages. *Behavioral and Brain Sciences, 39*, e70. doi: https://doi.org/10.1017/S0140525X1500076X

Fasanella, A. (2014). *On how learning mechanisms shape natural languages*. Doctoral Dissertation, Universitat Autònoma de Barcelona.

Fitch, W. T., & Friederici, A. (2012). Artificial grammar learning meets formal language theory: an overview. *Philosophical Transactions of the Royal Society B, 367*, 1933–1955. doi: https://doi.org/10.1098/rstb.2012.0103

Fodor, J. D. (1998). Unambiguous triggers. *Linguistic Inquiry, 29*, 1–36. doi: 10.1162/002438998553644

Fodor, J. D. (2009). Syntax acquisition: an evaluation measure after all? In M. Piattelli-Palmarini, P. Salaburu, & J. Uriagereka (Eds.), *Of minds and language: A dialogue with Noam Chomsky in the Basque Country* (pp. 44–57). Oxford: Oxford University Press.

Fodor, J. D., & Sakas, W. G. (2005). The Subset Principle in syntax: costs of compliance. *Journal of Linguistics, 41*, 513–569. doi:

Fonollosa, J., Neftci, E., & Rabinovich, M. (2015). Learning of chunking sequences in cognition and behavior. *PLOS Computational Biology, 11*(11), e1004592. doi: https://doi.org/10.1371/journal.pcbi.1004592

Gallistel, C. R., & King, A. P. (2009). *Memory and the computational brain*. Malden: Wiley-Blackwell.

Gibson, T., & Wexler, K. (1994). Triggers. *Linguistic Inquiry, 25*(3), 407–454.

Ho, H. T., Leung, J., Burr, D. C., Alais, D., & Morrone, M. C. (2017). Auditory sensitivity and decision criteria oscillate at different frequencies separately for the two ears. *Current Biology, 27,* 3643–3649. doi: https://doi.org/10.1016/j.cub.2017.10.017

Huang, C. T. J., & Roberts, I. (2016). Principles and parameters of Universal Grammar. In I. Roberts (Ed.), *The Oxford handbook of Universal Grammar* (pp. 306–354). Oxford: Oxford University Press.

Kazakov, D. L., Cordoni, G., Algahtani, E., Ceolin, A., Irimia, M-A., Kim, S-S., Michelioudakis, D., Radkevich, N., Guardiano, C., & Longobardi, G. Learning implicational models of universal grammar parameters. (2018). In C. Cuskley, M. Flaherty, H. Little, L. McCrohon, A. Ravignani, & T. Verhoef (Eds.), *The evolution of language: Proceedings of the 12th International Conference (EVOLANG XII)*. Online at http://evolang.org/torun/proceedings/papertemplate.html?p=176.

Levelt, W. J. M. (1974). *Formal grammars in linguistics and psycholinguistics*. The Hague: Mouton.

Lightfoot, D. (1991). How to set parameters. Cambridge, MA: MIT Press.

Lightfoot, D. (2020). *Born to parse: How children select their language*. Cambridge, MA: MIT Press.

Longobardi, G., & Guardiano, C. (2009). Evidence for syntax as a signal of historical relatedness. *Lingua, 119,* 1679–1706. doi: https://doi.org/10.1016/j.lingua.2008.09.012

Manzini, M. R. (2019). Parameters and the design of the Language Faculty. Northern Italian partial null subjects. *Evolutionary Linguistic Theory, 1*(1), 24–56. doi: https://doi.org/10.1075/elt.00003.man

Marcus, G., & Davis, E. (2021). Insights for AI from the human mind. *Communications of the ACM, 64*(1), 38–41. doi: 10.1145/3392663

Murphy, E. (2015). The brain dynamics of linguistic computation. *Frontiers in Psychology, 6*, 1515. doi: 10.3389/fpsyg.2015.01515

Murphy, E. (2017). Acquiring the impossible: developmental stages of copredication. *Frontiers in Psychology, 8,* 1072. doi: 10.3389/fpsyg.2017.01072

Murphy, E. (2020). *The oscillatory nature of language*. Cambridge: Cambridge University Press.

Murphy, E., & Shim, J.-Y. (2020). Copy invisibility and (non-)categorial labeling. *Linguistic Research, 37*(2), 187–215. doi: 10.17250/khisli.37.2.202006.002

Niyogi, P., & Berwick, R. C. (1996). A language learning model for finite parameter spaces. *Cognition, 61*(1–2), 161–193. doi: 10.1016/s0010-0277(96)00718-4

Niyogi, P., & Berwick, R. C. (1997). Evolutionary consequences of language learning. *Linguistics and Philosophy, 20,* 697–719. doi: https://doi.org/10.1023/A:1005319718167

Page, K. M. (2004). Language learning: how much evidence does a child need in order to learn to speak grammatically? *Bulletin of Mathematical Biology, 66,* 651–662. doi: 10.1016/j.bulm.2003.09.007

Pearl, L., & Lidz, J. (2013). Parameters in language acquisition. In C. Boeckx & K. K. Grohmann (Eds.), *The Cambridge handbook of biolinguistics* (pp. 129–159). Cambridge: Cambridge University Press.

Pearl, L., & Weinberg, A. (2007). Input filtering in syntactic acquisition: answers from language change modeling. *Language Learning and Development, 3*(1), 43–72. doi: 10.1080/15475440709337000

Rizzi, L. (2000). *Comparative syntax and language acquisition*. London: Routledge.

Sakas, G. W. (2000). Modeling the effect of cross-language ambiguity on human syntax acquisition. *Proceedings of the fourth Conference on Computational Natural Language Learning and the Second Learning Language in Logic Workshop,* 61–66.

Sakas, G. W., Yang, C., & Berwick, R. C. 2017. Parameter setting is feasible. *Linguistic Analysis, 41,* 391–408.

Sandler, W., Meir, I., Padden, C., & Aronoff, M. (2005). The emergence of grammar: Systematic study in a new language. *PNAS, 102,* 2661–2665. doi: https://doi.org/10.1073/pnas.0405448102

Villmoare, B. et al. (2015). Early Homo at 2.8 Ma from Ledi-Geraru, Afar, Ethiopia. *Science, 347,* 1352–1355. doi: 10.1126/science.aaa1343

Westergaard, M. (2014). Linguistic variation and micro-cues in first language acquisition. *Linguistic Variation, 14*(1), 26–45. doi: https://doi.org/10.1075/lv.14.1.02wes

Yang, C. (2002). *Knowledge and learning in natural language*. Oxford: Oxford University Press.

Yang, C. (2016). *The price of linguistic productivity. How children learn to break the rules of language*. Cambridge, MA: MIT Press.

Yang, C. (2017). Rage against the machine: evaluation metrics in the 21st century. *Language Acquisition: A Journal of Developmental Linguistics, 24*(2), 100–125. doi: doi.org/10.1080/10489223.2016.1274318

Yang, C., Crain, S., Berwick R. C., Chomsky, N., & Bolhuis, J. J. (2017). The growth of language: Universal Grammar, experience, and principles of computation. *Neuroscience and Biobehavioral Reviews, 81*, 103–119. doi: https://doi.org/10.1016/j.neubiorev.2016.12.023

Data, code and materials availability statement

The code is provided in the Appendix.

Authorship and Contributorship Statement

EL was involved in conceptualization of the research, data analysis, and data curation, and wrote the first draft of the manuscript. EM was involved in writing and editing the draft manuscript. All authors approved the final version of the manuscript and agree to be accountable for all aspects of the work in ensuring that questions related to the accuracy or integrity of any part of the work are appropriately investigated and resolved.

Acknowledgements

We thank the editor Brian MacWhinney and three anonymous reviewers for the useful feedback they provided. We are also grateful to Cristina Guardiano who answered questions about the analyzed pools of data, and to Cordian Riener for checking the mathematical aspects of the computation. This work received support from the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement n° 746652 and from the Spanish Ministry of Science, Innovation and Universities under the Ramón y Cajal grant agreement n° RYC2018-025456-I (to EL). The funders had no role in the writing of the study and in the decision to submit the article for publication.

Appendixes

```
import math

def computablePaths(paths):

    return math.factorial(paths)

def notComputablePaths(paths):

    if paths in [0, 1, 2]:

        return 0

    else:

        notCompPath = 0

        for l in range(2, paths + 1):

            temp = 1

            for t in range(0, l - 1):

                temp = temp * (paths - t)

            notCompPath = notCompPath + temp * (l - 2)

        return notCompPath

def calculateProbability(compPaths,notCompPaths, paths):

    totalPaths = compPaths + notCompPaths

    probability = float(compPaths / totalPaths)

    print(f"The probability of a successful computation is {probability * 100}%");

def main():

    finish = 1

    while(finish != 2):
        print("-" * 50);

        print("-" * 50 + "\n");

        paths = int(input("Number of paths: "))

        compPaths = computablePaths(paths)

        notCompPaths = notComputablePaths(paths)

        print(f"For {paths} paths, there are:\nWays of successful computation: {comp-
Paths}\nWays of unsuccessful computation: {notCompPaths}  \n")

        calculateProbability(compPaths, notCompPaths, paths);

        finish = int(input("\nDo you want to calculate another probability? \n1.Yes
2.No\n\n"))

        print("\n");


if __name__ == "__main__":

    main()
```

License