Á. Gallego & D. Ott (eds.). 2015. *50 Years Later: Reflections on Chomsky's* Aspects. Cambridge, MA: MITWPL.
Available for purchase at http://mitwpl.mit.edu

MIND THE GAP

ROBERT C. BERWICK
Massachusetts Institute of Technology

1 Descriptive and explanatory adequacy in *Aspects*

Chapter 1 of Chomsky 1965 (henceforth *Aspects*) doubles down on the goals of linguistic theory. Not only should we aim for *descriptive adequacy* – characterizing the possible human grammars – we should try to meet the more stringent demand of *explanatory adequacy*, reflecting the child's ability to select a descriptively adequate grammar given the data they receive:

> A child who is capable of language learning must have … (v) a method for selecting one of the (presumably infinitely many) hypotheses that are allowed by [the theory of grammar] and are compatible with the primary linguistic data. Correspondingly, a theory of linguistic structure that aims for explanatory adequacy must contain … (v) a way of evaluating alternative proposed grammars… [a] specification of a function *m* such that *m*(*i*) is an integer associated with the grammar G_i as its value (with, let us say, lower value indicated by higher number).
>
> *Aspects,* pp. 30-31

But what is this "way of evaluating alternative proposed grammars"? By way of illustration Chomsky contrasts two possible "grammars" for a dataset containing the possible sequences of English auxiliary verbal elements. There are eight such sequences: first an obligatory *Tense* element, then a *Modal* form or not (*will*, etc.); followed by a Perfective form (*have*) or not; then a *Progressive* (be) or not. Each binary choice allows for 2 possibilities, so there are 2^3=8 auxiliary sequences in all, including the sequence where none of the latter 3 elements is present. These 8 statements, the "data," can be generated by a grammar G1 with eight rewrite rules, as follows:

(1) a. Aux → Tense
b. Aux → Tense Modal
c. Aux → Tense Perfective
d. Aux → Tense Progressive

e. Aux → Tense Modal Perfective
f. Aux → Tense Modal Progressive
g. Aux → Tense Perfective Progressive
h. Aux → Tense Modal
i. Aux → Tense Modal

However, as Chomsky then describes, if our notational system can use parentheses to denote optionality, all eight data patterns can also be generated a second way, as grammar G2 with just a single rule, (2):

(2) Aux → Tense (*Modal*) (*Perfective*) (*Progressive*)

Since the three items in parentheses can either be present or not, it is straightforward to see that this one-rule grammar captures the same 2^3=8 examples as the eight rule grammar. The two grammars are thus both weakly and strongly equivalent in terms of descriptive adequacy. What about explanatory adequacy? G1 contains roughly the same number of rules that would be required to describe *any* random sequence of three binary-valued elements.[1] Consequently, this is the *worst* that a grammar can ever do in terms of explanation, since it fails to capture *any* regularity in the data. It has simply memorized the data as a list. In contrast, G2 is exponentially more succinct than G1. It is in this sense that G2 has "compressed" the original dataset into a smaller rule system, while G1 has not. This kind of compression may be taken as the hallmark of explanatory adequacy as set out in *Aspects*. In the remainder of this chapter we show that this kind of exponential reduction in grammar size can indeed serve as a litmus test for discriminating among otherwise descriptively equivalent grammars. Such gaps apparently arise whenever one grammar fails to capture generalizations and then must resort to memorizing – explicitly listing – the data as if it were a random string, while the alternative grammar "compresses" the data set exponentially. It is in this sense that "exponential gaps" indicate that overly large grammars have missed generalizations in the linguistic data. They lack explanatory adequacy.

It is also clear that the ability to compress data depends on the machinery the representational system provides, an empirical matter, again as Chomsky notes. Rule (2) depends on the availability of something like the parenthesis notation to denote optionality. If the auxiliary verb sequences had formed some *other* set of patterns, for instance, as *Aspects* notes, the cyclic permutation of {*Modal*, *Perfective, Progressive*}, then the parentheses notation would not yield a one-rule description of this new dataset. Rather, a system that included different representational machinery, cyclic permutation, would be able to compress these eight new examples down to one rule. In this way, explanatory adequacy all depends on the available representational armamentarium. Consequently, Chomsky rightly emphasizes that what does the heavy lifting in theory-building here is *not* the particular 'calculus' involved – counting symbols in this particular case – but rather the substantive, empirical constraints on the machinery of the mind/brain.

[1] Actually ternary valued, since there are three possible elements in each position or 3^3 or 27 possible random strings of length 3 rather than just 2^3. This corresponds to the number of bits required to encode any string over an alphabet of size 3. While one actually needs bits to more properly account for the coding of the notational system's alphabet, this detail does not change the comparative advantage of G2 as compared to G1. See Berwick (1982), chapter 4, for further discussion.

The remainder of this chapter is organized as follows. In Section 2 we show that succinctness gaps arise in the case of somewhat artificially constructed examples such as the difference between certain regular and strictly context-free grammars. Section 3 turns to more realistic linguistic territory and establishes that the same kind of exponential size difference holds between the 1970s-1980s models of transformational generative grammar (TGG) and Generalized Phrase Structure Grammar (GPSG): TGGs can be exponentially more succinct than strongly equivalent GPSGs. Intuitively, the reason why is that TGG factors apart movement from underlying phrase structure, while GPSG unnecessarily folds movement into the nonterminal labels of phrase structure rules. More recently, using essentially the same argument, Stabler (2013) has shown that a particular formalization of 'minimalist' grammars is exponentially more succinct than an otherwise equivalent extended version of context-free grammars called multiple context-free grammars. Taken together, results like these suggest that one should indeed 'mind the gap' between the sizes of alternative linguistic theories that are otherwise descriptively adequate. Section 4 concludes with a comparison of this basic result about the size of grammars to the related systems of evaluation known as minimum description length (MDL) and Bayesian model selection.

2 Regular grammars, context-free grammars, and explanatory adequacy

To see how the "mind the gap" litmus test for explanatory adequacy works in practice, we first consider a simplified, artificial example of two otherwise descriptively adequate theories for the same (finite) language. Section 3 then turns to a more realistic setting. Our artificial example focuses on the *finite* palindrome language over the alphabet of two symbols, {*a, b*}, with sentences less than or equal some fixed length, *n,* for even *n*. For instance, if *n*=4, then this language encompasses the six sentences, {*aa, bb, abba, baab, aaaa, bbbb*}. We denote this set of sentences by L_{PAL}-*n,* with *n* even, and let $|L_{PAL}\text{-}n|$ denote the size of this dataset. Since L_{PAL}-*n* for any even *n* is a finite language, it is also regular, and therefore describable by a (deterministic) finite-state automaton (FSA), $FSA_{PAL}\text{-}n$, as well as by a right-linear, regular grammar (RG), as shown in Figure 1(a). Here the nonterminals in the RG correspond to the states in the FSA, and the total size of the RG is 39 (putting to one side the exact bit length encoding). This language is also equally well describable via a pushdown state automaton (PDA) with 4 states and 10 rules as well as a context-free grammar with 4 rules as shown in Figure 1(b)[2].

[2] The restriction to a *finite* language is crucial here. If we move to *infinite* regular languages, then the succinctness gap between finite-state grammars and context-free grammars generating the same language becomes even more enormous – it need not be bounded by any recursive function (Meyer and Fischer, 1971). And in fact, the PDA and grammar in Figure 1(b) will recognize exactly the palindromic sentences of any length. To ensure that sentences longer than *n* are not recognized, the CFG should really have to use, e.g., nonterminals S_1, S_2, etc. to 'count' the length of the generated sentences. Note however that this only increases the size of the CFG (or PDA) by a linear amount. The grammar or PDA would still be exponentially more succinct than the equivalent RG. To simplify the presentation here, as with the FSA we have not included a special failure state; this would absorb any sentences longer than four or any non-palindromic sequences of length four or less. We have also glossed over details about measuring the RG or PDA/CFG size that are not relevant here. In reality, as noted in the main text, we must count the number of bits encoding the PDA, including its alphabet symbols, transitions, and pushdown stack moves. A more exact accounting in information theoretic terms would also include the *probability* of each automaton move; see Li and Vitányi (1997) for more details.

While both theories are descriptively adequate, it is not hard to show that as we consider palindrome languages of increasing length, the PDA becomes and exponentially more succinct than the corresponding FSA; similarly for their corresponding grammars. Further, it is easy to see by inspection of the grammar in Figure 1(a) that the finite-state representation in effect has simply listed all the possible strings in L_{PAL-4}, with $|L_{PAL-4}| \approx |RG_{PAL-4}|$. In contrast, the PDA/CFG *does* capture the palindrome generalization because the PDA or CFG for this language is smaller than the sentences it describes. The reason why this is so is straightforward. Each time *n* increases, say from 4 to 6, the FSA or grammar must be able to distinguish among all the possible new 'left-hand' parts of each palindrome sentence before the middle, in order to ensure that it can correctly decide whether the following 'right-hand' part is the same as the left. For the change from n=4 to 6, this means being able to distinguish among all the distinct strings of length 3, the first half of all the new palindromes of length 6, and there are 2^3=8 of these (*aaa, aab, abb,* etc). The FSA must be able to do this in order to recognize that the second half of a length-6 sentence is properly paired with the first half. But the only way an FSA can do this is to have distinct states for each one of these 2^3 possibilities. Therefore, in general, the palindrome-recognizing FSA will have to distinguish among $2^{n/2}$ strings. Thus as *n* increases, the number of states in the corresponding FSA or finite-state grammar must increase in proportion to $2^{n/2}$ – an exponential rate of increase in *n.* Put another way, the FSA must *memorize* all strings of length $n/2$ or less by using its states.

In contrast, for the palindrome recognizing PDA, increasing *n* does *not* demand a corresponding increase in its size, since it can use the same set of fixed general *rules* that "match up" corresponding pairs of *a*'s and *b*'s and these do not need to be altered at all: the size of the PDA grows linearly with respect to *n.* In this sense, the PDA has captured the essence of the palindrome-type language pattern, matching up paired symbols. In contrast, a similar-sized FSA could have just as well described *any* set of random strings up to length $n/2$, since the number of states in an FSA is the same for both palindromic strings and *any* random set of sentences of length $n/2$. In linguistic terms, we say that the FSA or corresponding grammar has "missed the generalization" for the palindrome pattern, because its description – the FSA or grammar size – is almost exactly the same size as the data to be described – that is, proportional to 2^n.

$Q_0 \rightarrow a\ Q_1 \mid Q_0 \rightarrow b\ Q_2$
$Q_1 \rightarrow a\ Q_3 \mid Q_1 \rightarrow a\ Q_3 \mid Q_1 \rightarrow b\ Q_4$
$Q_2 \rightarrow a\ Q_5 \mid Q_2 \rightarrow b \mid Q_2 \rightarrow b\ Q_6$
$Q_3 \rightarrow a\ Q_7 \mid Q_4 \rightarrow b\ Q_7$
$Q_5 \rightarrow a\ Q_8 \mid Q_6 \rightarrow b\ Q_8$
$Q_7 \rightarrow a \mid Q_8 \rightarrow b$

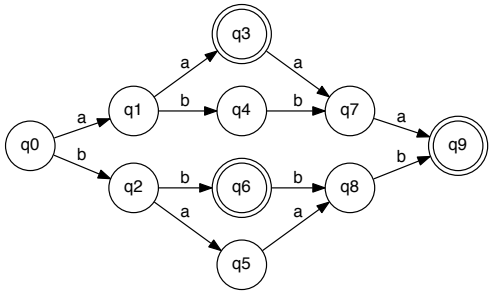

Figure 1(a). A regular grammar and a graphical representation of a deterministic finite-state automaton with 10 states and 10 directed arcs accepting L_{PAL-4}, the palindrome sentences over $\{a, b\}$, of length up to 4. The automaton starts in state q_0. State q_9, the double circle, is the final accepting state. The labeled directed arcs between states denote FSA transitions. Note how the

FSA uses distinct states to keep track of all possible left-hand parts of the palindrome sentences, of length up to 2; there are are 2^2=4 of these.

$$\text{Start} \rightarrow S$$
$$S \rightarrow a\ S\ a \mid \lambda$$
$$S \rightarrow b\ S\ b \mid \lambda$$

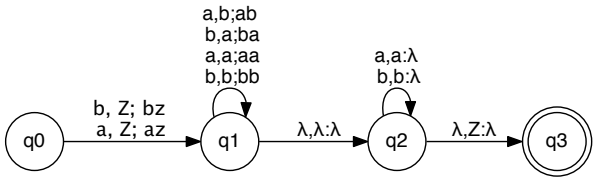

Figure 1(b) A context-free grammar and an equivalent push-down automaton for $L_{PAL\text{-}4}$ with 4 states and 10 push/pop transition operations that move the PDA between states. The PDA starts in state q_0 and its final state is q_3. The symbol λ denotes the empty string and *Z* a special bottom of the stack element. The operations on each arc are in the form *input*: *pop top* of stack symbol; new stack contents. For example, the operation *b*, *Z*; *bz* says that if a *b* is the current input symbol, and the result of popping off the top-of-stack is a *Z,* then the new stack configuration is *bZ*, i.e., *bZ* is pushed onto the stack, and the PDA moves to state q_1.[3]

3 Linguistic examples: GB, GPSG, and Minimalist grammars

The RG vs. CFG example of the previous section may appear artificial, but it does bear some resemblance to the pairing of Subject NPs and verbs in 'nested' or center-embedded sentences, demonstrating that two descriptively adequate theories can have radically different sizes, and illustrating that descriptive succinctness serves as a good litmus test for successfully capturing generalizations in data. More importantly however, one can also demonstrate this same effect in a more realistic setting, in the analysis of Generalized Phrase Structure Grammar (GPSG), as first observed in Berwick (1982).[4] Similar to the way in which FSGs can record finite palindrome patterns, but at a cost of an exponential growth in size of the resulting automaton/grammar when compared to PDAs/CFGs, GPSGs can record filler-gap dependencies via an expanded set of nonterminals, with a comparable exponential growth. As before, this indicates that the GPSG description has missed some generalization about filler-gap dependencies, and has simply listed the possibilities where a more compact representation is possible.

[3] To simplify the presentation, as in the FSA we have not included a special failure state; this would adsorb any sentences longer than four or any non-palindromic sequences of length four or less. The PDA as presented will of course accept all and only the palindromes of any length.

[4] An earlier version of this material first appeared in Berwick (1982), chapter 4. Details of the results described in this section may be found in this chapter. The main result that GPSGs grow exponentially in size with the number of 'filler' items that may be displaced, is established as theorem 4.1 there.

To demonstrate the exponential blow-up associated with GPSG and filler-gap dependencies, we first recall how GPSG encodes a single filler-gap dependency through an expanded set of nonterminals. We begin with a sentence that has all its 'fillers' in canonical argument positions, e.g.,

(3) John kissed Mary

As usual, an object *wh*-question version of this sentence has a 'filler, ' e.g., *who*, at the front of the sentence, linked to the 'gap' position as the argument of *kissed*:

(4) Who [did John kiss *gap*]

The GPSG analysis of this sentence links the filler *who* to the (unpronounced) gap via an analysis where ordinary nonterminals like S, VP, or NP have 'slashed' counterparts S/NP, VP/NP, and NP/NP, where XP/YP is to be interpreted as a constituent tree of type X that dominates (and is missing) a constituent of type YP expanded as an unpronounced element somewhere below (Gazdar, 1981). 'Slash' rules are introduced by context-free rules of the usual sort, but using the 'slashed' nonterminal names, e.g.:

(5) Comp → *Who* S/NP

In this way, one can arrange for a sequence of 'slash' rules to serve as a way to 'store' the information that a *wh*-NP is at the front of the sentence, as in (5), and pass that information down a (c-commanding) chain of nonterminal expansions, terminating with the rule NP/NP → ε, where ε is the empty string (the gap). In the terminology of contemporary transformational grammar, the GPSG rules construct something like a chain, with its head positioned at the filler, and then the encoding of that filler as intermediate slashed nonterminals, ending in a rule that expands as the empty string in the 'gap' position. The resulting syntactic tree would look something like the following in Figure (3) where irrelevant syntactic details have been omitted:

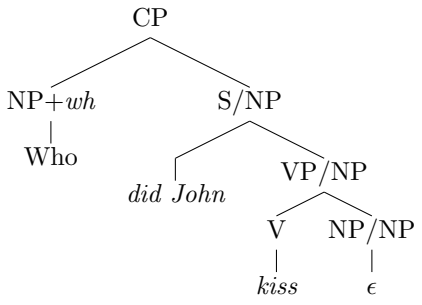

Figure 3: Slashed nonterminals for the sentence *Who did John kiss* link a *wh*-NP filler to its 'gap' position.

The question of succinctness of this representation comes into play when there are multiple 'displacements' from lower clauses to higher ones, in examples like the following, where a *wh*-NP filler *which violins*, fills a gap position after *on*, while a second NP filler *these sonatas* is the argument of *to play*:

(6) [Which violins]$_i$ are [these sonatas]$_j$ difficult to play [ ]$_j$ on [ ]$_i$?

How are we to deal with this kind of example, with two active 'fillers' via the slash-nonterminal representation? We can extend the 'slash' notation to explicitly carry these along via the nonterminal names, e.g.:

(7) CP → *wh*-NP$_1$ (*which violins*) S/NP$_1$; S/ NP$_1$ → NP$_2$ (*these sonatas*) S/NP$_1$NP$_2$; S/NP$_1$NP$_2$ → (pro) VP/NP$_1$NP$_2$; VP/NP$_1$NP$_2$ → V NP$_2$/NP$_2$ PP/NP$_1$; NP$_2$/NP$_2$ → ε; PP/NP$_1$ → NP$_1$/NP$_1$; NP$_1$/NP$_1$ → ε;

Consequently, in addition to 'single' slashed categories such as S/NP$_1$, handling multiple fillers requires additional categories like S/NP$_1$NP$_2$ (along with subsequent chains VP/NP$_1$NP$_2$ etc.), in effect listing the fillers and their order (here, via indices) so that they can be 'discharged' in the 'gap' positions in which they are to be interpreted. But in general, there might be any number of such 'displaced' constituents *n* as the length of sentences grows, and one could potentially choose to 'move' a filler from its canonical argument position or not, as well as possibly overlapping filler and gap chains. That is, it might be that the order S/NP$_2$NP$_1$ is sometimes be more acceptable than S/NP$_1$NP$_2$ (Stowell, 1982).[5] Without any additional constraints then, having to handle *n* displaced fillers in any possible order (including no displacement), implies that in the worst case a descriptively adequate GPSG for such languages would require distinct slashed nonterminals and rules for any of the possible 2^n subsets of the *n* fillers, an exponential number. Thus the size of GPSGs to handle filler-gap relations grows exponentially with the number of filler-gap pairs, the usual warning sign of a failure to capture some generalization, as with the FSG case described in Section 2.[6]

Put another way, encoding derivational 'movement' via the introduction of new nonterminal names is possible, but expands the grammar size *unnecessarily*: every intervening nonterminal and context-free rule along the nonterminal path from filler to gap is affected and requires slash-category additions, when in fact these intervening constituents almost always do not play any role in the filler-gap relation. The expansion in terms of rules is really unnecessary. In contrast, a GB-based account introduces no such unwarranted expansion in the number of nonterminals and rules, because it does not 'record' at each intermediate nonterminal the possible combinations of displaced constituents. These intervening elements are *irrelevant* to the separate rule of 'Move alpha,' which does not make reference to these intervening nonterminals (apart from a fixed set of constraints that do play a role, such as whether an intervening nonterminal is a bounding node like *CP*, landing sites, and the like). Another way to state this result is that GPSG has 'multiplied out' all the possible phrase structure chains that can intervene between fillers and gaps, even though the identity of the particular phrase structure elements involved does not really matter. In contrast, GB keeps these two components separate, at a real savings in grammar size.

---

[5] Fodor (1978) argues that the filler-gap order must always follow a nested, pushdown stack form, but this result is open to question; see Berwick (1982), Chapter 3.

[6] Theorem 4.1 in Berwick (1982) has a more detailed account of this result. The approach there uses (deterministic) frontier-to-root tree automata, and shows that if there were a non-exponential size GPSG for handling filler gap relations via new nonterminals, then this would imply that there is a non-exponentially sized FSA for recognizing the palindrome languages of a given length, shown to be impossible in the main text. This 1982 result should be compared to the one in Stabler (2013) described immediately below.

More recently, an exponential succinctness gap has also been demonstrated by Stabler (2013), when comparing the relative grammar sizes of a formalization of 'minimalist' grammars (MGs) to an extension of context free grammars known as multiple context-free grammars (MCFGs). Stabler demonstrates that two otherwise strongly equivalent MGs and MCFGs can differ exponentially in size: the MCFG contains exponentially more rules than the equivalent MG – and for the same reason as with the GPSG/GB contrast in Section 2. In an MG, the role of 'movement' is taken over by an operation called 'internal merge' – that is, the operation of merge between two syntactic (set-formulated) objects where one, the constituent that would be the target of move alpha in GB theory and the "moved constituent," is a subset of the other. The output is a new syntactic object where the moved constituent is copied to its new 'filler' position – there is no 'gap left behind which would be occupied by a phonologically empty element in older GB theory, but simply the original constituent. 8(a-b) illustrate the syntactic structure before and after an internal merge operation (irrelevant details suppressed); note that the second argument to the merge operator, [$_{DP}$ what], is contained in (is a subset of) the first argument.

(8a) Merge([$_{CP}$ people [$_{VP}$ eat [$_{DP}$ what]]], [$_{DP}$ what]) →
(b) [$_{CP}$ [$_{DP}$ what] [people [$_{VP}$ eat [$_{DP}$ what]]]]

Aside from the initial position of the constituent *what* and its 'landing site', this movement (actually copying) does not need to refer to any intermediate syntax in between (such as VP). If there is more than one 'mover', then this simply adds a single new lexical entry, as described in more detail in Stabler (2013). In contrast, in MCFGs, such effects are handled by the use of variables associated with nonterminals that can hold strings such as *what*. For example, the nonterminal VP(x_1, x_2) stands for an ordinary VP that has two variables that can hold the values of two strings, x_1, and x_2. The first variable can correspond to the verb *eat*, while the second can correspond to *what*. The following MCFG rule indicates that the value of the first variable – the verb – can be associated with the V nonterminal, while the second variable holding *what* can be associated with the DP:

(9) VP(x_1, x_2) → V(x_1) DP(x_2)

This *wh* element variable can then be 'passed up' through each higher nonterminal until *what* reaches its landing site in the CP, as per the following MCFG rule. Note in particular that in the MCFG nonterminal CP that the order of the two variables is reversed and their values concatenated together into one string x_2 x_1, which places *what* before the verb:

(10) CP(x_2 x_1) → C VP(x_1, x_2)

It should be apparent that the MCFG proceeds just as with GPSG when it encodes movement, in that *all* intervening nonterminals are modified along the chain from the initial position of *what* all the way to its landing site – as before, affecting intervening nonterminals by introducing variables even though those nonterminals are not really implicated in the 'movement.' As a result, if there are multiple possible 'movers,' each corresponding to a different kind of lexical item or feature, then the MCFG must multiply out all these possibilities as with GPSG. This amounts to an exponential number of choices since the possible movements can be any of the subsets of the movers, and each might move or not, as Stabler notes. Stabler concludes: "MGs can be exponentially smaller than their strongly equivalent MCFGs because MCFGs explicitly

code each movement possibility into the category system, while MGs can, in effect, quantify over all categories with a given feature." This exponential succinctness gap between MGs and MCFGs again serves to signal that MGs can capture a generalization that the MCFGs cannot, in this case, a lexical generalization that is quantified over all nonterminals with a particular feature.

4 Conclusion: explanatory adequacy and aspects of simplicity

Putting together the results from the previous sections, we see that a grammar's relative succinctness can be one way to fruitfully probe into whether a linguistic theory is able to capture natural generalizations or not, exactly as anticipated in *Aspects*. We conclude here by showing that this approach is in line with two other approaches to "evaluating" grammatical theories, one called "Minimum Description Length" (MDL, Rissanen, 1978), and the other, Bayesian model selection.

We first consider MDL. Suppose we have a family of grammars $\mathcal{G}$ and a given set of sentences, a 'corpus,' D.[7] MDL defines the 'best' grammar G over some family of grammars $\mathcal{G}$ as that grammar which minimizes the sum of two components. The first is $|G|$, the size of the grammar as measured in bits. This factor is perhaps the one most familiar to linguists. The second factor is $|D_G|$, the size of the data as encoded or generated by G. The intuition behind this measure is that a good theory will be able to "compress" the original data D such that $|D_G|$ is smaller, usually much smaller, than D itself. If D is not compressible, then there is no smaller description of the data by any theory aside from a listing of the data itself. Note that if there are some data examples that the given grammar cannot generate – exceptions to rules, in traditional terminology – then these must be added in to the $|D_G|$ size factor by an explicit listing of these examples without any compression. We can write out the MDL formulation in the following way:

(10) $|G| + |D_G|$

Conventionally, this is given as the following optimization problem:

(11) $\text{argmin}_{G \in \mathcal{G}} \; |G| + |D_G|$

Without working through all the details, this MDL approach will yield the same "exponential gap" litmus test as described in Sections 2 and 3: exponential gaps in grammar size will show up in the first $|G|$ factor, while explicit listing of data – if one is not using an appropriate notational framework – will show up in exponential gaps in the second factor. For one way of using this MDL framework in a concrete linguistic application, the inference of morphological and syntactic regularities starting from strings of phonemes and proceeding through to syntax, see de Marcken (1996). More recently, MDL has been explicitly incorporated in several other models of language acquisition, e.g., using categorial grammars (Villavicencio, 2002); or slightly augmented context-free grammars (Hsu and Chater, 2010). A related approach to MDL uses the

[7] One might rightly enquire as to whether the number of sentences in D is infinite or not (see Note 2). For our purposes here this does not matter. For one way of systematically handling this question, using the notion of "uniform computability" as in the case of a sequence of fixed Boolean circuits, see Berwick (1982).

notion of *program size complexity:* it attempts to find the length of the shortest program that can compute or generate a particular language. Chater *et al*. (2003) and Hsu, *et al.* (2013) discuss this approach in the context of language learnability; more on its relationship to linguistic theories can also be found in Berwick (1982).[8]

Further, the MDL approach itself can be closely identified with Bayesian evaluation methods as first pioneered by Horning (1969), who used Bayesian inference to select grammars within an acquisition-by-enumeration approach. In a Bayesian framework, we fix some set of (observed) data D and some description of D via a class of (stochastic) context-free grammars, $\mathcal{G}$, that can generate D, with some prior probability. A Bayesian inference approach would then attempt to find the particular grammar $G \in \mathcal{G}$ that maximizes the posterior probability of G given the data D, i.e., argmax $_{G\in\mathcal{G}}$ $p(G\ |D|$. Letting $p(G)$ denote the *prior* probability assigned to the (stochastic) context-free grammar g, we can use Bayes' rule in the usual way to find this maximum by computing $p(G|D) = \text{argmax}_{G\in\mathcal{G}}\ p(D|G)$ x $p(G)/p(D)$. Since D is fixed, it can be ignored to find the G that maximizes the product, and we have the usual computation that attempts to find the grammar G that maximizes the product of the prior probability of G times the likelihood of the data given some particular grammar, $p(D|G)$. In other words, Bayesian inference attempts to find the G that satisfies the following formula:

(12) $G = \text{argmax}_{G\in\mathcal{G}}\ p(D|G) \times p(G)$

To pass from this formulation to the MDL version one way to proceed is as follows. The MDL principle as applied to stochastic context-free grammars says that the 'best' grammar G *minimizes* the sum of the description length of the grammar and the description length of the data given G. More precisely, if $|G|$ is the length of the shortest encoding of grammar G and $|D_G|$ is the length of the shortest encoding of the data D given the grammar G, then MDL attempts to find:

(13) $G = \text{argmin}_{G\in\mathcal{G}}\ |G| + |D_G|$

Using near-optimal coding schemes, Shannon's source coding theorem (1948) implies that the description of the length of D with respect to a particular grammar G can be made to closely approach the value $-\log_2 p(D|G)$. We can further assume, as is standard, that one way to define the prior probability of a stochastic context-free grammar $p(G)$ is as $2^{-|G|}$. Larger grammars are penalized in the same sense that we have used throughout this chapter and have lower probability. Taking $\log_2$ of (8) we get:

(14) $G = \text{argmax}_{G\in\mathcal{G}}\ \log_2 p(D|G) + \log_2 p(G)$

Substituting $-|D_G|$ for $\log_2 p(D|G)$ and $2^{-|G|}$ for $p(G)$ and pulling out the negative sign we get:

[8] The notion of program size complexity was developed independently by Solomonoff (1960, 1964) and Kolmogorov (1965); for a recent comprehensive survey of this field, see Li and Vitányi (1997). In this framework one can show that there is an 'optimal' universal programming system (a representation language or grammar) in the sense that it is within a constant factor of any other optimal programming language. A detailed analysis of this quite useful approach lies beyond the scope of this chapter; for a concrete example in the context of language acquisition, see Hsu and Chater (2010).

(15) $G = \text{argmax}_{G \in \mathcal{G}} \; -|D_G| + (-|G|) = \text{argmin}_{G \in \mathcal{G}} \; |D_G| + |G|$

In this particular case then, the MDL formulation (13) and the Bayesian formulation (14) coincide.[9]

It should be emphasized that this is not the only notion of "explanatory adequacy" that might prove valuable in choosing among otherwise descriptively adequate linguistic theories. Other approaches might stress the computational complexity of acquisition – the sample complexity or the number of examples required to acquire language (Berwick, 1982, 1985); or the computational complexity associated with the use of language, that is, parsing or production.

Summarizing, we have found that three different ways to formulate grammar evaluation in light of explanatory adequacy all amount to the same kind of calculation, ultimately grounded on the notion of the *size* of a grammar plus the size of the linguistic data as encoded by that grammar. Further, this measure can be applied to actual alternative theoretical proposals in linguistics, distinguishing between proposals that offer generalizations of data as opposed to those that do not. Finally, this analysis shows that currently popular approaches to learning in cognitive science, such as Bayesian methods, turn out to be worked-out versions of explanatory adequacy as discussed in *Aspects*. In this respect, contrary to what is sometimes thought, the informal notion of size and simplicity as litmus tests for linguistic theories can be placed within a coherent framework. We take all this as lending support to the view, first stated in *Aspects*, that explanatory adequacy has an important role in evaluating linguistic theories, incorporating a view that one must attend to how grammars are acquired or inferred from data.

References

Berwick, R.C. 1982. *Locality Principles and the Acquisition of Syntactic Knowledge*. Ph.D. thesis, Department of Electrical Engineering and Computer Science, MIT.

Berwick, R.C. 1985. *The Acquisition of Syntactic Knowledge.* Cambridge, MA: MIT Press.

Chater, N. and Vitányi, P. 2003. Simplicity: a unifying principle in cognitive science? *Trends in Cognitive Sciences* 7(1), 19–22.

Chomsky, N. 1951. *Morphonemics of Modern Hebrew*, S.M. thesis, University of Pennsylvania, New York: Garland Publishing/Taylor and Francis.

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

De Marcken, C. 1996. *Unsupervised Language Acquisition*. Cambridge, MA: Ph.D. thesis, MIT Department of Electrical Engineering and Computer Science.

Fodor, J.A., 1978. Parsing strategies and constraints on transformations. *Linguistic Inquiry* 9(3), 427-473.

Gazdar, G. 1981. Unbounded dependencies and coordinate structure. *Linguistic Inquiry* 12, 155-184.

Horning, J. 1969. *A Study of Grammatical Inference*. Ph.D. thesis, Department of Computer Science, Stanford University, Stanford, CA.

[9] This Bayesian estimation is usually called a 'maximum *a posteriori* estimate' or MAP estimate. Note that a Bayesian account would give us much more than this, not only just the maximum *a posteriori* estimate, but also the entire posterior distribution – the whole point of Bayesian analysis, on some accounts.

Hsu, A., Chater, N., Vitányi, P. 2013. Language learning from positive evidence, reconsidered: A simplicity-based approach. *Topics in Cognitive Science* 5(1), 35-55.

Hsu, A., Chater, N. 2010. The logical problem of language acquisition: a probabilistic perspective. *Cognitive Science* 34, 972-1016.

Kolmogorov, A.N. 1965. Three approaches to the quantitative definition of fnformation. *Problems Information Transmission* 1(1), 1–7.

Li, M. and Vitányi, P. 1997. *An Introduction to Kolmogorov Complexity Theory and Its Applications*. New York: Springer Verlag.

Meyer A. and Fisher, M. 1971. Economy of description by automata, grammars, and formal systems. *IEEE 12th Annual Symposium on Switching and Automata Theory,* 125-129.

Rissanen, J., 1978. Modeling by shortest data description. *Automatica* 14 (5), 465–658.

Shannon, C. 1948. A mathematical theory of communication. *The Bell System Technical Journal* 27**,** 379–423, 623–656.

Solomonoff, R. 1960. A preliminary report on a general theory of inductive inference. Report V-131. Cambridge, MA: Zator Company.

Solomonoff, R. 1964. A formal theory of inductive inference part I. *Information and Control* 7(1), 1–22.

Stabler, E. 2013. Two models of minimalist, incremental syntactic analysis. *Topics in Cognitive Science* 5(3), 611-633.

Villavicencio, A. 2002. The Acquisition of a Unification-Based Generalised Categorial Grammar. Cambridge: Cambridge University, TR-533.