

Unambiguous Triggers

Janet Dean Fodor

Triggers for parameter setting may be ambiguous. Strategies for dealing with ambiguity include guessing, parallel processing, and waiting for unambiguous input. The Trigger Learning Algorithm of Gibson and Wexler (1994) is a guessing system. Gibson and Wexler show that under some reasonable assumptions it may never attain the target grammar. I propose instead a deterministic device that waits for unambiguous triggers to set parameters. This learner must be able to *detect* parametric ambiguity. This is not possible on the classical conception of triggers as sentences that flip parameter switches. It is possible if a trigger (and the parameter value it triggers) is a piece of tree structure, perhaps a single feature, made available by Universal Grammar, and adopted into the learner's grammar if input sentences cannot be parsed without it.

Keywords: acquisition, ambiguity, parameter setting, parsing, trigger

1 Problems of Ambiguity

1.1 Ambiguity and Learning

Gibson and Wexler (1994) have argued that the major word order parameters (specifier-head, complement-head, and verb-second) lack adequate triggers for parameter setting. Their work on word order provides another illustration of the concern first raised by Clark (1989): that triggers may be ambiguous, and that once an ambiguity has led to error, the import of subsequent triggers may be distorted so that recovery is impossible. The conception of learning as triggering solved a host of problems that troubled the more traditional hypothesis-testing models. Unlike hypothesis formation and testing, triggering is not a “creative” process but is simple and “mechanical”; it should therefore be more uniform across learners as well as leading more reliably to the correct grammar. However, just because triggering is so automatic, problems of mis-triggering can arise. These include mis-triggering of parameters by ungrammatical input (see Grodzinsky 1989, Kapur 1994), by well-formed but exceptional constructions (Fodor 1994), and by ambiguous inputs as discussed by Clark, and by Gibson and Wexler (henceforth GW). If we are to continue to model

I am grateful to Virginia Valian, Ted Gibson, and an anonymous reviewer for their good advice on content and presentation. For helpful discussion I thank participants at the Conference on Linguistics, Cognitive Science and Childhood Language Disorders at CUNY Graduate Center in 1994, at the OTS Workshop in Utrecht in 1995, and at the Conference on Generative Approaches to Second Language Acquisition at CUNY Graduate Center in 1995; Virginia Valian and the students in our seminar at CUNY; Stefano Bertolo, Ted Gibson, Jay Keyser, David Pesetsky, Ken Wexler, and other members of the RTG group at MIT. A preliminary version of some of this material appeared under the title “Fewer but Better Triggers” as Fodor 1995.

grammar acquisition as parameter triggering, we need to show how learners can either avoid or recover from the kinds of damage that would otherwise result from misleading inputs such as these.

Sentence (1) is not ambiguous in adult English, but it is ambiguous for a learner that does not yet know that the target language is English.¹

(1) Mary saw me.

As GW observe, a sentence like (1) is compatible with English parameter settings (SV, VO, and –V2) and with German parameter settings (SV, OV, and +V2), among others. (For ease of reading I use SV and VS to denote the parameter values specifier-initial and specifier-final, respectively, and OV and VO for the values complement-initial and complement-final. +V2 is the parameter value for a verb-second language; other languages are –V2.) The learner must discover which combination of parameter settings is correct. Fortunately, other sentences of the language provide sufficient information to make this possible. (For example, *Mary has seen me* selects VO; *Unfortunately Mary saw me* reveals –V2.)² But if it happens that (1) is encountered before these unambiguous inputs are, what does the learner do with (1) in the meantime? Suppose, for example, that the current grammar has the values SV, OV, and –V2 (regardless, for now, of how these were arrived at). These do not license (1), so at least one parameter needs to be reset. Resetting either the complement-head parameter from OV to VO, or the V2 parameter from –V2 to +V2, would accommodate (1); nothing about (1) indicates which is correct.³ This is parametric ambiguity. What could or should a learner do in these circumstances? What does a human learner in fact do?

Problems due to ambiguity are familiar in other areas of psycholinguistics and can provide perspective for studying how ambiguity affects learning. In sentence-processing research, strategies for dealing with ambiguity are commonly classified into (a) *serial* processing (select one analysis and revise it later if necessary), (b) *parallel* processing (compute both/all analyses and use subsequent information to eliminate all but one), and (c) *delay* or “wait-and-see” (wait for more information before making any decision). For parametric ambiguities in learning, the same three broad options present themselves. Consider a learning device faced with an input *I* that is not licensed by the current parameter settings but that could be licensed by resetting either parameter P_i or parameter P_j . (For simplicity, we may consider just two-way ambiguity.) The learner could (a) guess which one of the two to reset and revise this guess later if necessary, (b) compute both

¹ I use the term *learner* to refer to the learning mechanism in the mind/brain (analogously to the standard use of the term *parser* to denote the mental parsing mechanism).

² A reviewer points out that some adverbs can precede XP and the finite verb in German root clauses (e.g., *Nun ich kann singen* ‘now I can sing’ as well as *Nun kann ich singen*). I will ignore such facts here. In this article I adopt GW’s account of the linguistic facts, which is deliberately simplified in order to focus attention on the logical problems that ambiguity poses for language acquisition.

³ The V2 phenomenon is well chosen as a working example for studying parametric ambiguity. Learners cannot disambiguate it semantically since there is no shift of meaning between the two syntactic analyses (as also for Clark’s Case assignment ambiguity). If the comparison between English and German is representative, there is no prosodic disambiguation either. (See Morgan, Meier, and Newport 1987 on the availability to learners of structural information derived from prosodic contours in other constructions.)

alternatives until subsequent input decides between them, or (c) make no grammar change until unambiguous input becomes available. There is no way to tell a priori which of these approaches to ambiguity is favored by the human language acquisition device. All three have characteristic advantages and disadvantages. Serial processing is efficient when successful but is obviously risky. There is evidence that the human sentence processor operates at least partly in serial mode, but the perils are considerably less in sentence parsing than in learning. In parsing, the worst penalty would be noncomprehension or miscomprehension of one utterance; in acquisition, it would be a possibly lifelong misapprehension about which sentences are in the language. A parallel approach is safer, though costly in resources. A delay system is also safe, but might prove too slow to be realistic. Also, as will become clear, it is not easy to achieve.

GW's learning algorithm is of type (a); it is a serial system that makes mistakes and relies on being able to correct them later. For a parametrically ambiguous input like (1), GW's system would choose at random between shifting to VO and shifting to +V2. The reasons for adopting this guessing strategy will be discussed below. A learning system of type (b), with parallel processing of both possible parameter settings, is presented by Valian (1990b). I return to this briefly in section 1.3. The learning model that I advocate here is of type (c); it is a wait-and-see device that makes no grammar changes at all when input is parametrically ambiguous. Thus, all three approaches are being investigated. The latter two, parallel and delay, differ from GW's serial learner in two important respects. As noted, they are conservative learners, accepting only what there is decisive evidence for. Also, they respond differentially to ambiguous and unambiguous inputs. They must therefore be able to *detect* (not consciously) when an input is ambiguous, in order to respond appropriately to it: to compute two analyses for it, or to halt decision making until more information is available. By contrast, a serial system need never know whether its response to the input is the only one possible or whether alternatives exist. It *ignores* ambiguity, rather than attempting to contend with it. This is what GW's learning device does. It acquires grammars with some success despite having no special mechanisms or strategies at all for coping with ambiguous input.

However, as might be anticipated, this success is not complete. In learning just as in parsing, a serial approach to ambiguities results in "garden paths." At a choice point the learner adopts an analysis and proceeds without attention to the existence of alternatives. It is bound to make a wrong selection sometimes. As will be illustrated below, that mistake can then distort other choices that arise later, engendering more mistakes. The learner may thus wander further and further from the target grammar. If successful learning is to be guaranteed, either parametric garden paths must be avoidable or else they must be curable after they have occurred. But GW demonstrate that their learning system is incapable, in some circumstances, of either remedy. It cannot avoid garden paths because it makes grammar changes based on ambiguous input (i.e., it guesses). But it also cannot cure some garden paths, because of constraints (the Single Value Constraint and the Greediness Constraint; see appendix for discussion) that GW impose in order to facilitate learning by limiting the grammars that learners are allowed to entertain. As GW demonstrate, the consequence is that some target languages may never be achieved. How to remedy this is the important question that their work raises.

1.2 Triggers and Triggering

GW use facts about word order to illustrate the general problem of how to ensure convergence on the target grammar when triggers are unreliable. For simplicity they restrict consideration to an artificially limited set of eight target languages in which there are no other sources of word order variation than the three major word order parameters. They make the following working assumptions: that C always precedes IP; that the order of I and VP is the same as the order of V and O; that in a +V2 language, movement of the finite verb to C is obligatory; that there are no null subjects; that there is no other transformational reordering of constituents, such as scrambling; and that there is no free word order (i.e., the two values of the specifier-head parameter are mutually exclusive, as are the two values of the complement-head parameter). Thus, the only possible base word orders are VOS, OVS, SVO, and SOV, and the only (overt) transformational movements permitted are of V to C, together with movement of XP to [Spec, CP], in +V2 languages. For –V2 languages GW do not mention movement of V to positions within the inflectional projections between V and C. Since it is not critical to the discussion here, I will make the simplifying assumptions that only I is projected between VP and CP and that the finite verb (Aux or main) always moves overtly to I in a –V2 language (unlike English, for instance). GW also say nothing about movement of the subject from VP to IP, so we may suppose for simplicity that subjects are base-generated in [Spec, IP]. All adverbs are sentence-initial. It goes without saying that the learning problems will increase rather than decrease as and when these temporary working restrictions on word order variability in target languages are lifted.

GW (p. 409) give formal definitions, shown in (2), which capture the ideal conception of triggers as fully reliable stimuli for grammar change.

- (2) a. A *global trigger* for value v of parameter P_i , $P_i(v)$, is a sentence S from the target grammar L such that S is grammatical if and only if the value for P_i is v , no matter what the values for parameters other than P_i are.
- b. Given values for all parameters but one, parameter P_i , a *local trigger* for value v of parameter P_i , $P_i(v)$, is a sentence S from the target grammar L such that S is grammatical if and only if the value for P_i is v .

These definitions entail that a sentence is a trigger (local or global) only if it is an unambiguous signal that a certain parameter needs to be set to a certain value; that value is a necessary condition for generation of the sentence.⁴ Sentence (1) would thus not qualify as a trigger for either VO or +V2 (or for any of the parameter values in this system). If grammar changes resulted exclusively from encounters with unambiguous triggers as characterized by (2), learning could proceed

⁴ This is not quite true. As shown in the appendix, (2b) admits parametrically ambiguous triggers also, but a simple revision can block this. In the meantime it will simplify discussion to proceed as if (2) did succeed in picking out just the unambiguous triggers. Also, the word *grammar* in GW's definitions shown in (2a) and (2b) was presumably a slip for *language*.

fully deterministically. The learning mechanism would change a parameter to a new value when and only when it encountered an unambiguous trigger for that value; parametrically ambiguous inputs would never induce a grammar change. Hence, no errors would occur (except due to other causes such as performance slips or incorrect input); there would be no straying down parametric garden paths. However, a conservative learning procedure of this kind would need to be able to tell when an input is parametrically ambiguous. It would have to be able to recognize the genuine triggers somehow, to sort them out from all other input sentences that (because they are parametrically ambiguous) do not satisfy the definitions in (2).

GW evidently assume that the learning mechanism cannot make this discrimination between (unambiguous) triggers and (ambiguous) nontriggers. Though they define triggers as unambiguous, their learning algorithm does not make reference to triggers so defined, and it does not rely exclusively on unambiguous inputs. The parameter-setting mechanism that GW present has all the characteristics of a *nondeterministic* system. It is a probabilistic device that establishes parameter values in large part by random guesswork, on the basis of unreliable information including parametrically ambiguous inputs like (1). Despite its name, GW's Triggering Learning Algorithm (pp. 409–410) treats triggers and nontriggers alike, and makes grammar changes based on both.

(3) *The Triggering Learning Algorithm (TLA)*

Given an initial set of values for n binary-valued parameters, the learner attempts to syntactically analyze an incoming sentence S . If S can be successfully analyzed, then the learner's hypothesis regarding the target grammar is left unchanged. If, however, the learner cannot analyze S , then the learner uniformly selects a parameter P (with probability $1/n$ for each parameter), changes the value associated with P , and tries to reprocess S using the new parameter value. If analysis is now possible, then the parameter value change is adopted. Otherwise, the original parameter value is retained.

A parameter value is not adopted, under this procedure, unless it permits analysis of the input sentence. But note that it is adopted regardless of whether any *other* parameter change would also have permitted analysis of the sentence, that is, regardless of whether the input is parametrically ambiguous. Thus, the value +V2 might be adopted in response to (1), even though (1) is not (under the definitions in (2)) a trigger for +V2, and even though adopting +V2 could very well be an error. In general, the TLA allows (in fact, it requires) that the grammar be changed in situations where the change is not decisively motivated by the input. Having incorporated this assumption into the definition of triggering, GW then demonstrate the problems that it creates.

An undirected nondeterministic learning device must wade through enormous numbers of wrong parameter value combinations before hitting on the correct grammar. For a deterministic learner, 20 binary parameters could be set by 20 triggering events (fewer, if there are default values). For an unconstrained guessing system, 20 binary parameters implies a semirandom walk through a space of over 1 million parameter value combinations. (If the number of parameters is greater, this difference in scale widens even further.) GW note (p. 408) that this is “an enormous search problem” and propose to make it more feasible for learners by imposing certain constraints on the learner's search patterns. These constraints are claimed to improve search efficiency overall,

but they do so at a cost.⁵ Though convergence is now feasible, it is guaranteed only for some of the possible target grammars but not for all. This is the problem of “missing triggers” that is the primary focus of GW’s discussion. They prove that not every target language contains trigger sentences that would shift the learner from an arbitrary wrong grammar to that target grammar, under the constraints imposed. Therefore, the learner may get stuck permanently at a wrong grammar (called a *local maximum*).

GW present and evaluate a number of solutions to this problem. The solution they favor (pp. 414–415) is to assume that parameters can have default settings and that these defaults may be “locked in” for some period of time; during that time the learner can explore only those areas within the parameter space that are compatible with the defaults. This can alleviate learnability problems if the defaults are so chosen that the grammars that are off-limits at first are the troublesome ones that afford no escape to other grammars. (See Bertolo 1995.) GW consider, but do not adopt, the alternative move of preventing these search problems from arising in the first place, by giving up the guessing procedure that creates them. If the learning procedure could, after all, be endowed with the ability to discriminate between parametrically ambiguous and unambiguous inputs, then it could rely solely on the true triggers (in the sense of definition (2)) to set parameters. Each parameter would be set just once, to the correct value (or left, correctly, at its default value); trial-and-error behavior would be minimal or nonexistent. Such a model would reflect, more closely than the TLA implementation does, Chomsky’s conception of triggering as instantaneous correct learning by switch setting, made possible by the shift to a principles-and-parameters theory of Universal Grammar (UG; Chomsky 1986:sec. 3.5.1). This ideal picture is surely worth maintaining if possible. To find that it is untenable would be an extremely important—though disappointing—discovery.

I will argue in section 2 that direct, error-free triggering *is* a realistic possibility. In the model I present, the one absolute rule for language learners is *Do not learn from ambiguous input*. I will try to show that learners are capable of obeying this injunction because they *can* detect parametric ambiguity with great reliability. However, devising a mechanism that is capable of this (and is otherwise plausible) is far from easy. As I will demonstrate, it demands a new conception of triggers (as structures, not sentences) and a new conception of their relation to parameter values (viz., identity). A trigger is a small piece of tree structure (a few nodes, perhaps only partially specified in features; in the limiting case a single feature) that is made available by UG and is adopted into a learner’s grammar if it proves essential for parsing input sentences. Once in the grammar, it can be used in generating and parsing new sentences, just as X-bar schemata are, and other grammar contents such as lexical entries. The learner’s task is to find

⁵ GW imply but do not say that the TLA search process governed by the Single Value Constraint and the Greediness Constraint is more efficient than an unconstrained search. Later in their discussion (pp. 442–443) they argue for imposing the constraints on grounds of conservatism (minimizing differences between successive grammars hypothesized), limiting the resources consumed, and linguistic naturalness, but all three points are arguable. GW do not quantify the effort involved in the learning task, or compare demands with the constraints and without them, on measures such as the average number of inputs prior to convergence or the average number of grammar changes prior to convergence. Berwick and Niyogi (1996) report the outcomes of simulations showing that more inputs are needed with the Greediness Constraint and/or the Single Value Constraint than without them.

out which of these treelets the target grammar contains, by establishing which of them occur as ingredients of the sentence structures in the input sample. How the learner can do this is discussed below.

This conception of triggers marks a considerable shift from the more conventional view represented by the TLA, in which a trigger is a sentence (a word string) whose occurrence in the input provides the learner with a reason to flip a parameter switch from one of its settings to another. I will argue that the conceptual change to triggers as little bits of structure is in keeping with current thinking in linguistic theory. I will also try to show that it is essential to a psychologically plausible learning theory: nothing less will permit efficient error-free parameter setting in the face of ambiguous input. The practical obstacles in the way of deterministic learning on the standard view of triggers are overwhelming, as GW imply.

I will set about the project by establishing as precisely as possible the sources of nondeterminism in the TLA. Once the contributing factors are isolated, it will be possible to construct a recipe for the design of a deterministic alternative. The goal is to modify the TLA just sufficiently to create a learning system that detects and discards ambiguous inputs and uses only unambiguous triggers to set parameters.

1.3 Triggering and Parsing

The TLA sets parameters without limiting itself to the trustworthy triggers defined in (2). As the TLA is formulated, any input sentence the learner encounters, ambiguous or not, may be used to set any parameter value that could contribute to licensing it (except where prevented by Greediness and the Single Value Constraint). The first goal is to understand *why* the TLA is lax in this respect, in order to establish whether its imprecision is avoidable.

Let us consider more closely the discrepancy between triggers as defined by (2) and the notion of triggers implicit in the TLA. I will refer to the latter as *TLA triggers*. (For simplicity, we may ignore the difference between local and global triggers for now. It is discussed in detail in the appendix.) Suppose that grammars G and G' differ only in that G has $P_i(v)$ but G' has $P_i(v')$, where v and v' are the two values of binary parameter P_i . Then for a learner whose current hypothesis is G' , a TLA trigger for $P_i(v)$ is a sentence S such that S is not grammatical in $L(G')$ (the language generated by grammar G') but is grammatical in $L(G)$. The TLA in effect implements the method of differences: if changing $P_i(v')$ to $P_i(v)$ changes S from ungrammatical to grammatical, then the observation that S is grammatical (that it occurs in the target sample) is a reason for thinking that P_i has the value (v). However, it is *only* a reason; it is not decisive evidence. It is compatible with there being other combinations of parameter values, not including $P_i(v)$, that render S grammatical. So S may be a TLA trigger even if it is parametrically ambiguous. Indeed, S can be a TLA trigger for $P_i(v)$ and also be a TLA trigger for a value of a completely different parameter, even a conflicting one.⁶

⁶ As a first approximation, it might be said that a sentence is parametrically ambiguous just in case it is licensed by two or more distinct combinations of values for all the parameters. However, this would greatly overestimate parametric ambiguity. Some parameters are *irrelevant* to the licensing of a sentence; for instance, the complement-head parameter

Consider example (1) in this light. Though it does not qualify as a trigger under (2), it is a TLA trigger. The sentence pattern SVO is ungrammatical in the language whose grammar has values SV , OV , and $-V2$, but it is grammatical in the language obtained by switching $-V2$ to $+V2$, all else remaining the same. By definition, then, SVO is a TLA trigger for the value $+V2$. However, it is also true that SVO becomes grammatical if, starting from the grammar SV , OV , and $-V2$, the complement-head parameter is switched from OV to VO . Thus, SVO is a TLA trigger for the value VO also. The fact that SVO is parametrically ambiguous is a matter of indifference to the TLA, though sufficient to disqualify it as a trigger in the stricter sense of (2). The closest that SVO can approach to (2) is with a disjunction: this sentence is grammatical if and only if [*either* the value for the $V2$ parameter is $+V2$ *or* the value for the complement-head parameter is VO]. Since this disjunction is ineliminable, SVO is not a true trigger for either $+V2$ or VO . This comports with the practical fact that a learner encountering an SVO sentence cannot deduce from it how either of the two parameters is set in the target language.

The question to be addressed is whether the TLA can be brought into line with GW's more stringent definitions in (2), by providing the learning device with the capability of distinguishing between an input sentence that cannot be licensed *without* a certain parameter value, and one that merely can be licensed *with* that parameter value. If the TLA could thereby be strengthened into a more precise implementation of (2), then the learner could reject all parametrically ambiguous inputs and learn without error. To achieve this, it seems, the learner would need knowledge of the various many-to-one and one-to-many relationships that hold between sentences and parameter values.

GW consider three ways in which a learner might know which sentences are triggers for which parameter values. (a) The associations between parameter values and the sentences they distinctively license might be innately given, "simply built in as part of the learner's knowledge" (p. 444). (b) The learner "can somehow logically deduce" these relationships "from her knowledge of grammar" (p. 445). (c) The learner might apply an on-line test to each input sentence to establish what parameter changes would allow it to be generated. GW reject the first alternative. They write (p. 445), "The possibility that the learner has these deductions built in is not attractive, since this extra knowledge is completely redundant with the rest of the grammar. Thus, we may rule out this possibility."⁷ They regard alternative (b) as more plausible than (a), but it too is

is irrelevant to the licensing of a VS sentence. Parameter P_i is irrelevant to the licensing of sentence S (equivalently: is not expressed by sentence S ; see Clark and Roberts 1993) if and only if for every combination of values for the other parameters, whether or not that combination licenses S is unaffected by the value of P_i . We would not want S to count as parametrically ambiguous with respect to P_i in that case. For example, although VS can be licensed by VS , VO , $-V2$ and by VS , OV , $-V2$, it is not ambiguous with respect to the complement-head parameter in the way that SVO is, but only vacuously so. Thus, parametric ambiguity is best defined over *relevant* parameters only. Presupposing this from now on, we may say that a sentence S is (nonvacuously) parametrically ambiguous with respect to a (relevant) parameter value (or combination of values) if at least one but not all grammars that license S have that parameter value (or combination of values).

⁷ This is still alternative (a), not (b), even though it is "deductions" that are said to be built in. The wording merely reflects the fact that GW present (a) and (b) together as alternative implementations of the same general approach, in contradistinction to (c).

ultimately disfavored. GW liken the deductive process in (b) to the kind of reasoning required by a traditional hypothesis-testing model, with its familiar disadvantages. They note (p. 445), “[T]he general form of the deductive processes that the learner can perform is not clear, nor is it clear how to constrain these processes so that they do not require too many resources.”

Having excluded (a) and (b), GW focus their attention on (c), the on-line parsing test for triggerhood. The TLA has such a test built into it. On examination it becomes clear that the particular test incorporated in the TLA is not powerful enough in principle to be able to screen out misleading nontriggers. The mechanics of this test are thus the real source of the nondeterministic character of GW’s learning model. This on-line triggerhood test constitutes the interface between parsing and learning in GW’s model, and it *necessitates* nondeterministic parameter setting. That is, GW’s model is stochastic not just for mathematical or expository convenience, as some learning models are, but inherently and necessarily so, given that it relies on a trigger identification test that lacks a mechanism for dealing with ambiguity.

This is how the TLA’s triggerhood test works. The learner responds to a novel input (an input sentence that is unparseable on the current grammar) by attempting to parse it with one parameter, selected at random, reset. If the parse is successful, that parameter value is adopted as part of the grammar; otherwise, the current grammar is not changed. For an unambiguous input, the random selection of a parameter value to undergo the parse test does no real harm; it may delay achievement of the target grammar (because most parameter values so selected will fail the test), but it never results in adoption of an incorrect value. If the parameter value tested is incorrect, or is irrelevant to the input sentence, it will fail the parse test; if it passes the parse test, it must be correct.⁸ By contrast, for an ambiguous input, two or more different parameter values could pass the parse test. The random selection of which parameter value to test then translates into a random choice of which of them to adopt. Suppose the current grammar has values SV, OV, and –V2, and the current input is the parametrically ambiguous (1). If the learner selects the value VO for testing, that value will permit a successful parse of (1) and will thus be adopted. But equally, if the learner had picked +V2 to test, it would have permitted a successful parse and would have been adopted. Thus, the “successful-parse” criterion neither filters out ambiguous inputs nor attempts to adjudicate between their alternative analyses; it does not even register *that* they are ambiguous. (The TLA parsing test is useful, but only in preventing adoption of superficially unsuccessful parameter values; this is the Greediness Constraint, discussed in the appendix.)

Could the successful-parse criterion be modified so that it does not let ambiguous inputs through? Apparently not. Suppose we gave up the assumption that the choice of a parameter value to test is random and instead allowed the choice to be shaped by preferences of various kinds. The problem would remain. For example, there might be a priority ranking giving precedence to VO over +V2 (or vice versa) to undergo the parse test; as a result, that parameter value would

⁸ The TLA parse test gives decisive information only if the input is unambiguous, the learner knows that it is, and the Single Value Constraint is obeyed. In those circumstances a successful parse test tells the learner that a relevant parameter has been switched to the correct value. See Fodor, in press a, for discussion.

be favored for adoption into the grammar. But because it is independent of the input, any such ranking would be bound to lead to error for one target language or another: what is correct for English is wrong for German, and vice versa. Suppose instead that the parse test was not restricted to trying out just one new parameter value at a time, but could try two or more new values in combination. (This amounts to giving up the Single Value Constraint, which restricts the TLA to testing only grammars that differ by one parameter value from the current grammar. See the appendix.) This could affect the probability of a successful parse, but the ambiguity problem would remain; the TLA could still adopt a wrong value, or a wrong combination of values, for an ambiguous input like (1). There is only one modification of the on-line licensing test that would make it significantly more discriminating: to give up the restriction that only one new *grammar* (with one or more new parameter values) is tried out on each input sentence. Parametric ambiguity is a matter of whether there is more than one grammar (more than one complete set of parameter values) that could license the input. To establish that there is not, and hence that it is safe to employ that input as a trigger for grammar change, the learning device would have to conduct an *exhaustive* check of all possible grammars. That is, it would have to test-parse the input sentence with every currently unadopted parameter value, and every combination of them. But the mathematics are prohibitive here. To detect parametric ambiguity in the limited domain of three word order parameters would require up to seven parses (following the initial failed parse with the current grammar) of every novel input; for 20 binary parameters it would require up to 1 million parses of each input. Unambiguous inputs would require the greatest labor, since all grammars must be checked to prove that only one succeeds.⁹

Clearly, GW were wise to formulate the triggering algorithm in such a way that it does not presuppose testing on this scale; it is not a practical possibility. Yet without an exhaustive search, the successful-parse criterion for triggerhood is a leaky sieve. It lets in nontriggers along with the triggers, so learners make errors and drift through many wrong grammars instead of setting the parameters correctly once and for all. One might hope to find a compromise solution—a criterion that is not too lax, with a workload that is not too heavy. But halfway measures do not look promising. For instance, nothing much would be gained by parsing each input with two new grammars, or three new grammars; the risk of error comes from the grammars *not* tested and so would still be significant. It also would not help to parse each input just once, but with all parameter values switched on simultaneously. Even if that were possible (overlooking mutual exclusiveness of values of the same parameter), it would provide no information. The parse would certainly succeed (unless the sentence were beyond the bounds of core grammar), but it would be entirely unclear which parameter value(s), or even how many, were responsible for the success. Some one or more of the values just switched in must have contributed to make the sentence

⁹ The learner is assumed to keep no record of outcomes of prior learning events (apart from the effects they had on the current grammar), so a parameter combination is not excluded from future consideration once it has been tested and failed. Keeping records on a million combinations of parameter values would not be feasible, while doing so for 20 individual parameter values would be pointless because a value may be part of a failing combination and yet be correct. But even if it were practical, eliminating some grammars by this means would hardly change the scale of the TLA's grammar-testing problem.

parsable; but nothing more than this could be concluded (without, as before, testing all possibilities individually).

Valian (1993) has proposed a refinement of the parse test designed to improve its efficiency. This is essential for the learning system that she proposes (Valian 1990b, 1993), which is a parallel device that weighs the alternative settings of a parameter for some time before selecting between them. A parallel system (like a delay system, as I propose) has to know when an input is parametrically ambiguous (see section 1.1), so it must somehow surmount the computational load problem. Valian proposes that the parsing test is limited to a set of potentially *relevant* parameter values, which are identified by a pretest. On this model, at any stage of learning there can be some parameters that are definitively set, never to be changed again, while others are unset since evidence for one setting or the other is still being weighed. Each parameter has associated with it a characterization of a set of inputs that could be relevant to what the value of that parameter should be; this is its “evidence set.” On encountering a novel input sentence, the learning device first checks all of the still-unset parameters to see if the input falls within the evidence set of any of them. Then it parses the input with the established values of all the parameters that are already set and with *both* values of every parameter that is not yet set but for which the outcome of the pretest was positive. For example: sentences without overt subjects (but not sentences with overt subjects) are in the evidence set for the null subject parameter, so if that parameter is not yet set, a subjectless input will pass the relevance test and will then be parsed using the current grammar together with each value of the null subject parameter in turn; an input with an overt subject will be parsed only with the current grammar (unless it is relevant to some other unset parameter).

Clearly, this procedure will be more successful at identifying parametric ambiguity than the TLA is. But the TLA, after all, would succeed also if it were run as an exhaustive test with as many parses per input as necessary to detect ambiguity. The question, then, is whether Valian’s procedure is significantly more economical, and thus more plausible psychologically, than an exhaustive version of the TLA parse test. It is not clear how cost-effective the two-stage procedure would be. It depends in part on whether the pretest can be run in parallel, since running 20 relevance pretests seriatim, one for each of 20 parameters, would itself exceed plausible limits. The advantage of the pretest is that it is assumed to be based on a very simple parse of the sentence (perhaps just as a sequence of NPs and predicates, without establishing other structural relations), not crucially dependent on any particular values of other parameters. Because of this, there can be just one relevance test per parameter, rather than one per combination of parameters (per grammar), thus avoiding the exponential extravagance of an exhaustive version of the TLA. Twenty tests is a lot of work, but much better than a million. However, at least two sources of complexity still remain. One concern is how many parameters get through the relevance screening to be tested at the second stage. The fewer that do, the greater the advantage over a simple one-stage search procedure. But the nonspecificity of the preliminary parse entails that the pretests for relevance are quite crude, and this would reduce their efficiency in weeding out parameter values that are not worth running the full parse test on. For instance, whether or not a clitic has climbed out of its clause may not be entirely evident without a complete parse of the sentence. So the evidence set for the null subject parameter, which presumably includes sentences with

climbed clitics, must be defined loosely enough that it won't miss any potential instances of clitic climbing. If it did, the learner would underestimate parametric ambiguity and hence would be liable to error. But then it follows that the null subject parameter—and others too—might often remain in the pool for the second stage of testing even when ideally it would have been screened out at the first stage. This then feeds the second concern, which is that the pretest cannot tame the problem of scale at the second stage, which results from the need to check all *combinations* of parameter values in order to reliably detect ambiguity. At this point, if not before, the combinatorics raise the complexity stakes very fast. For example, if 7 of 20 parameters pass the relevance test for a given sentence, then over a hundred parses of that sentence are still needed at the second stage. There is no doubt that establishing relevance is the key to avoiding the labor of a vast search through the whole domain of grammars, but the kind of relevance pretest that Valian proposes appears not to be powerful enough. The workload in detecting parametric ambiguity would be less prohibitive than for GW's procedure run in exhaustive mode, but it seems that it would still be too high for psychological plausibility.

Attempts to expand and streamline the TLA parsing test for deterministic learning are thus not encouraging. It seems necessary to break away from that model and find a different design for an error-free triggering algorithm. The difficulties that the TLA encounters show where to look. We can conclude from the observations above that any parsing test that could serve as a perfect sieve to catch just the unambiguous triggers would have to be able (a) to check all possible combinations of parameter values essentially simultaneously, (b) to tell when more than one combination could license the input, and (c) to report *which* parameter values license it when only one combination does. Only a mechanism with these capabilities could efficiently deliver to the learner the information it needs in order to ignore ambiguous triggers and set correct parameter values on the basis of unambiguous ones. If a triggerhood test of this kind is truly impossible, as GW judged it to be, that is well worth knowing. It would underwrite the necessity of the stochastic approach taken by GW and focus research efforts on developing it further. However, if such a test were possible, it could make an enormous contribution to accuracy and efficiency in learning models of all varieties. In the next section I show how a redefinition of triggers as ingredients of tree structures makes it possible to achieve goals (a)–(c); it permits parallel testing of all grammars, with outcomes attributable to individual parameter values.

2 Learning without Ambiguity

2.1 Parametric Ambiguity and Structural Ambiguity

One insight of the TLA that should not be lost is the pivotal role of sentence parsing in grammar acquisition. A learner's input is a stream of speech sounds in conjunction with some nonlinguistic stimuli; it has no structure other than what is imposed on it by the learner. Thus, learners have access to the structural properties of the target language only insofar as they can parse their input. This fact is often deliberately disregarded in learnability research, as a strategy for isolating other learnability problems of interest. (Exceptions are Berwick 1985 and Valian 1990a.) But it obviously cannot be ignored in the study of parametric ambiguity, especially once we require the

learner to actively monitor the input for ambiguities. It is arguable that all parametric ambiguity reduces to structural ambiguity. Since structural ambiguity is the province of the parsing routines, not specifically a phenomenon of learning, this has important implications for the range of possible mechanisms for coping with ambiguity in acquisition. We are in need of a detector for parametric ambiguity. If parametric ambiguity is a species of structural ambiguity, then a structural ambiguity detector would do the job. Let us consider what it would take to make this work.

The relation between parametric and structural ambiguity can be illustrated with GW's word order examples. An *SVO* sentence such as (1) (*Mary saw me*) is parametrically ambiguous between +V2 and -V2. It is also structurally ambiguous. As a +V2 construction it has four analyses (on GW's working assumptions, summarized in section 1.2). They all have *saw* in C and *Mary* in [Spec, CP]. They differ with respect to the positions of the traces of *saw* and *Mary*, which might precede or follow the object, and precede or follow each other. On the -V2 parameter setting the sentence has one analysis, with *saw* in I and its trace in V; *Mary* is in [Spec, IP] and is not part of a movement chain. The learning device would know the correct structural analysis of (1) if it knew the correct settings of the parameters, as adults do. Suppose, though, that (1) is a novel input not yet licensed by the learner's grammar. Then the converse is of more interest: the learner would know the correct parameter settings if it knew the correct structural analysis. Parsing the sentence and determining its import as a trigger are one and the same thing.

This is true throughout GW's example domain. It is easy to check that in every case, parametric ambiguity is correlated with structural ambiguity. Plausibly, this is not just an empirical generalization but is true as a matter of principle. A counterexample would be a sentence with a unique grammatical derivation that could be licensed by two or more different combinations of parameter settings.¹⁰ For this situation to exist, it would have to be the case that nothing in the structure of the sentence reflected which of the parameter value combinations was responsible for it. Though this is logically possible, it seems more in keeping with current linguistic theory to suppose that this kind of "invisible licensing" does not occur. If a constituent moves, for instance, there must be some other constituent or feature in the sentence structure that causes it to move (Chomsky 1993); likewise for other syntactic operations.¹¹ If this is generally true, then one syntactic structure (S-Structure, or complete structural description) could not have two differ-

¹⁰ To illustrate: A possible counterexample would be Clark's (1989) Case assignment ambiguity, where, according to Government-Binding Theory at the time, the subject of an infinitive clause could receive accusative Case either from the matrix verb or from the infinitive, a parametric choice. If the sequence of structures constituting the derivation shows only the Case assigned, not what assigned it, then the two derivations would be indistinguishable; see Fodor 1992b for discussion. However, exceptional Case marking is now treated differently (Chomsky 1993), and the two derivations do involve distinct structural configurations.

¹¹ For monostratal theories comparable points apply, though the details differ. Note that to exclude "invisible" licensing by parameter values does not mean that licensing elements must be perceptible (phonetically realized). For instance, the distinction between strong and weak features in the sense of Chomsky (1993) is not coextensive with which features have overt morphological realizations. More of a concern, though I will not address it here, is that features that motivate movement may be erased in the process and thus not present at S-Structure (input to Spell-Out). It is a common assumption that, since traces at S-Structure record underlying positions of moved constituents and thus recapitulate the derivation, the parser need not compute representations deeper than S-Structure. In that case the learner would not observe structural triggers directly, but only their consequences.

ent parametric sources. For each parameter value involved in licensing a sentence, there would be, somewhere in the derivation, a feature specification or structural configuration associated uniquely with that value.

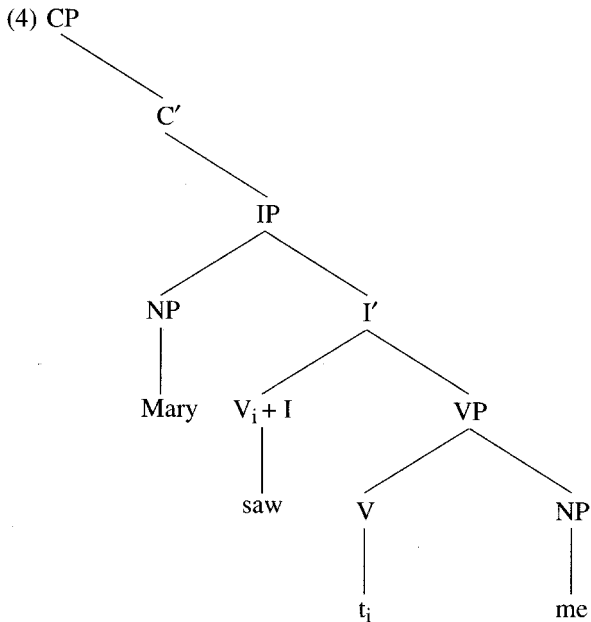
Not all structural ambiguity is due to parametric ambiguity. In adult languages, which are parametrically fixed, there can be more than one derivation of the same surface word string. For example, the parameter settings for (adult) English license two derivations for *Mary saw old men and women*. Even in GW's restricted word order domain, a word string without overt Case marking can be structurally ambiguous relative to a single grammar; for example, for any grammar with +V2, the sequence *Mary saw Bill* could be a surface SVO sentence or a surface OVS sentence. Thus, a structural ambiguity detector would pick up too much; some of the ambiguities it would catch would not be parametric ambiguities. However, for purposes of safe learning this imprecision is not damaging. The important direction of generalization is: every parametrically ambiguous sentence is structurally ambiguous. It follows that any learning device that can reliably detect structural ambiguity could screen out all potential cases of parametric ambiguity. It might discard all structurally ambiguous examples, or it might subject them to a further test for parametric ambiguity. Either way, only univocal triggers would be left to learn from.

It appears, then, that to build an error-proof learner we should supply it with a parser that detects structural ambiguity. In fact, such parsers exist. A parallel parser registers the existence of multiple structural analyses, but so do most familiar parsers even though they do not then pursue all analyses throughout the sentence. Even a serial parser makes a decision at a choice point in a sentence where the grammar offers two ways of incorporating the current word into the structure, and it could be programmed to signal the existence of the choice point before choosing one of the options to pursue; this is called *flagged serial parsing* (see Inoue and Fodor 1995). The one important difference between standard parsing models and what the learning device needs is that standard parsers operate with just one grammar at a time, whereas the learner needs a parser that is sensitive to ambiguity *across* grammars. For instance, the five different derivations of an SVO sentence like (1) in GW's word order domain are licensed by five different grammars. Thus, the learner's parser would have to operate with five (mutually inconsistent) grammars at the same time, and in more realistic cases with many more. This is exactly as observed in section 1.3: the need for a multigrammar parse, and the consequent problem of scale. It is clear, then, that the reduction of parametric ambiguity to structural ambiguity does not by itself solve the problem of ambiguity detection. So far it has given some reassurance that the identification of reliable triggers does not call for a totally novel kind of mechanism: the parser is at least the right *sort* of device for the job, though it still needs a significant boost in power in order to be able to check all grammars. This is good news. It means we can maintain GW's idea of using the parsing routines (which must exist, independently of learning) to do the work of relating sentences to the parameter values that license them.

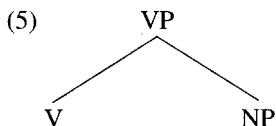
The relation between parametric and structural ambiguity begins to do some serious substantive work if we reformulate it with a different emphasis. If the discussion above is correct, it follows that *every* input sentence is parametrically *unambiguous* under a complete structural description. For instance, an *SVO* sequence with a structure in which the lexical verb is in I is

unambiguous evidence for parameter values SV , VO , and $-V2$. An SVO string with a structure in which the verb is in C , and the trace of the verb precedes both the object and the trace of the fronted subject, is unambiguous evidence for VS , VO , and $+V2$. This is obviously not helpful to a learner that is unaware of the structure of any given SVO input. But let us put this fact aside for the moment and consider instead the intriguing possibility that triggers *are* sentences under full structural descriptions. Then every input sentence (now construed as structure, not string) would be a fully reliable trigger in the sense of (2) for every parameter value that contributed to its generation.

For example, the sentence structure (4) would be an unambiguous trigger for the values SV , VO , and $-V2$.

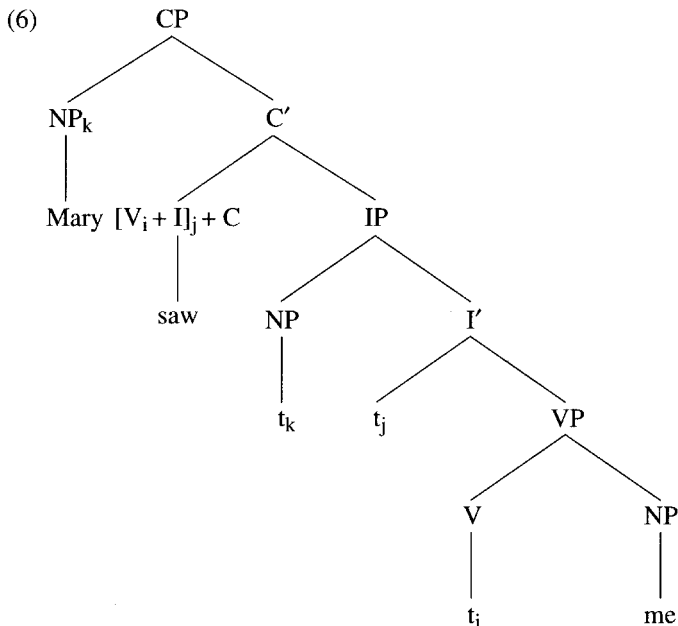


In fact, once triggers are allowed to include structure, we can go further still. What makes (4) a trigger for the VO value of the complement-head parameter is not its surface VO order, the fact that the object NP is to the right of the overt verb; it is the fact that the object is to the right of the trace of the verb. The complement-head parameter is concerned with the underlying order of constituents. The surface position of V is not really of interest, except that it just happens, together with other constraints that GW assume, to entail that the underlying V was to the left of O in this example. (Recall that GW assume that I is to the left or right of V if V is to the left or right respectively of O . Hence, if a verb surfaces in I to the left of O , the VP must be V -initial.) So the trigger for the value VO is not really (4), but just that piece of (4) shown in (5). This seems entirely appropriate now: the trigger for the underlying order of verb and object is the underlying order of verb and object. In fact, since underlying positions are indicated by trace positions in S -Structure, we can conveniently ignore other structures in the derivation (but see footnote 11).



The underlying order of verb and object is reflected in the surface order of the tail of the verb's chain and the tail of the object's chain, which (5) captures. More precisely still: the "real" trigger for VO is whatever determines the configuration in (5), according to the linguistic theory that is being assumed. In a theory such as Tree-Adjoining Grammar it might be (5) itself. In HPSG (Head-Driven Phrase Structure Grammar) it might be a schematic version of (5) showing just the feature HEAD[LEX+] on the left daughter. In Government-Binding Theory it might be a government direction feature of the verb. In the Minimalist Program it might be a weak NP-feature of Agr_O that blocks object movement into preverbal position before Spell-Out. For neutrality in what follows I will take (5) itself to be the VO trigger.

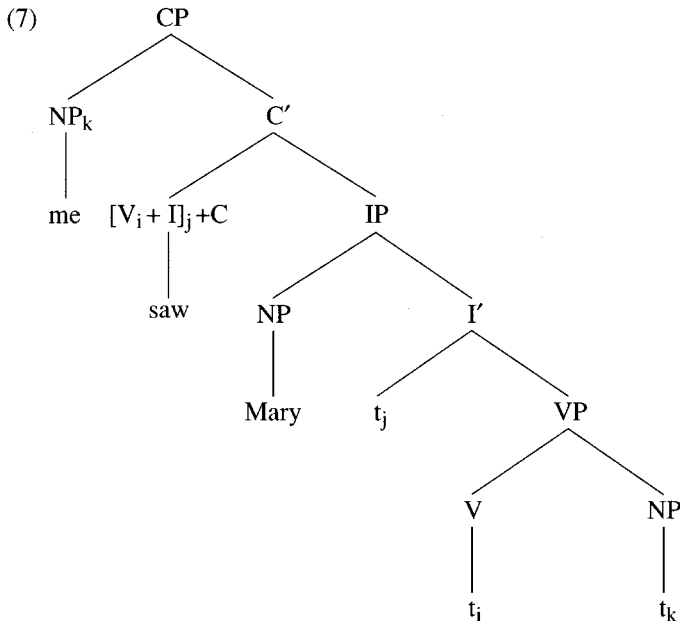
For the other parameter values, also, the trigger will be just the relevant fragment of the S-Structure, or whatever the linguistic theory holds responsible for it. For the SV value of the specifier-head parameter, the relevant fragment of (4) is the subtree with the specifier of IP in relation to its sister I', regardless of what occupies these positions at the surface level. In (4) the subject is in situ and the lexical verb is in I; for the same string in a +V2 language the [Spec, IP] position would contain the trace of the subject and I' would dominate the trace of the finite verb, as shown in (6) for SV, VO, +V2. For the V2 parameter, the trigger for +V2 would be the lexical verb in the head of CP, or whatever deeper feature is responsible for that: for instance,



the feature [+fin] on C, if +V2 is due to obligatory movement of V to a finite C (Holmberg and Platzack 1991), or a strong C feature that attracts V + I (Chomsky 1993). For the -V2 value, the trigger might be absence of a verb in C, or presence of a nonempty verb in I at S-Structure, or better still, just C[-fin] or a weak C feature on the assumption that this entails no movement of V to C.

Each parameter value is thus associated with its own structural (or in the limiting case, featural) signature, that is, with whatever constitutes its essential contribution to the sentence structures that are licensed by grammars which have that value. This has two consequences relevant to present goals. One is that a parametrically ambiguous sentence will have two (or more) structures that differ from each other with respect to which trigger structures are contained within them. For example, sentence (1) has, among others, the structures (4) and (6); (4) contains the structural trigger for -V2, and (6) contains the structural trigger for +V2. Though we have not solved the practical parsing problem yet, it is clear that in principle a cross-grammar parallel parse of (1) would yield structures (4) and (6), which transparently reveal their parametric differences. A more modest parsing procedure would establish one structure and merely note the possibility of an alternative one; but that would suffice to establish ambiguity and warn the learner away. The second and rather remarkable consequence of this shift to triggers as fragments of trees is that each parameter value now has a global trigger, that is, a trigger that occurs in every language which has that parameter value (see the appendix). GW, taking triggers to be whole sentences (or sentence *patterns*; see below) observed (p. 425) that global triggers are rare, because parameter values interact in sentence derivations and therefore the contribution of one parameter manifests itself differently in different languages. GW noted that there are no global triggers at all in their three-parameter word order domain. Consider, for instance, the VO value. The trigger for VO in English would be an *SVO* sentence, but in a specifier-final, -V2 language it would be a *VO S* sentence, and in a +V2 language it would have to be something more elaborate such as *S Aux VO*. No one sentence pattern appears in all VO languages; each language needs its own local trigger. By contrast, the treelet (5) (or its deeper source) is common to all and only VO languages. It will be present in the structure of every sentence that has an object and is generated by a grammar with the VO value of the complement-head parameter, even a sentence such as *Me saw Mary* with surface OV order, *if* that sentence was generated by a VO grammar. The structure of such a sentence in an SV, VO, +V2 language is shown in (7); the VO trigger (5) is discernible within it. Thus, where the TLA model has only ambiguous local triggers for the complement-head parameter, (5) is a fully reliable, global trigger for the VO value. Its mirror image is a fully reliable, global trigger for the OV value. Of course, there is no way to tell from a word string such as *Me saw Mary* that (7) is its structure in some target language. Therefore, it would be impossible for a learner to know whether this word string, as uttered on a particular occasion, does or does not instantiate the global trigger (5). Thus, (5) is an unambiguous trigger, as desired, but for practical purposes it appears to be of no use at all.

This contrasts sharply with GW's conception of triggers, which is much more down to earth. GW chose a middle-level characterization of triggers—more than just words but less than tree



structures (or full syntactic derivations). They refer to triggers as “sentence patterns” (p. 412 and elsewhere) and give examples such as *Subject Verb Object* or *Adv V Aux S*. The grounds for adopting this intermediate characterization in terms of sequences of syntactic categories and functions seem eminently reasonable. At this level triggers are characterized in a way that best captures how they manifest themselves to learners. Learners cannot hear hierarchical structure (see footnote 3), so triggers cannot be hierarchically structured entities. Learners can hear word strings (assuming appropriate lexical entries have been acquired somehow), so triggers could be word strings. But considering the resources of a normal learner, triggers could be more than word strings. If the lexicon can provide syntactic categories for the words, triggers could be category strings. If the nonlinguistic environment sometimes provides information about grammatical functions such as subject and object, then triggers could be the kind of category/function strings that GW assume. This does seem a very plausible estimate of what a learner might reasonably be expected to achieve (at least on some proportion of input sentences, enough for learning to proceed). Anything beyond this level of analysis (such as tree structures) would seem to presuppose the grammatical knowledge that the learner is in the process of acquiring and so would not be possible.

GW’s characterization of triggers thus has the merit of making triggers accessible to learners. By contrast, the characterization of triggers as syntactic subtrees does not. As just noted, it leaves unexplained how learners could recognize the triggers to which they are exposed. Thus, the conception of triggers as tree fragments solves one problem and creates another one. With triggers as *strings*, the problem is that many inputs that look like triggers are in fact not triggers because they are ambiguous; and since they can’t be filtered out, they can’t be prevented from mis-

triggering parameters. With triggers as *structures*, every sentence utterance is an unambiguous trigger guaranteed not to mis-set parameters, but learners do not know which triggers they are hearing; learners would apparently have to perceive these structural triggers somehow in the input word strings, despite the fact that they are not superficially evident.

Even though it does not look promising, the idea of structural triggers is an important part of the solution to the ambiguity problem.¹² It is the key to designing a system that can analyze input using multiple grammars without computational overload. Crucially, structural triggers make it possible to fold all possible grammars into a single grammar, so that simultaneous testing of multiple grammars can be effected by a standard parsing device (flagged serial or stronger), applying just one grammar as usual. Only one more step is needed to make everything work, and that is to give up the picture of the learner's task as that of *spotting* these nonperceptible trigger structures in audible word strings. Rather, the trigger structures can be *contributed* to word strings, to enable a parse where otherwise none would be possible (see Fodor 1989). The learner should adopt a parameter value just in case its trigger uniquely bridges a gap in the structural analysis of an input sentence. The mechanics are set out in the next section.

2.2 Parsing with Structural Triggers

The ambiguity problem has divided into the problem of trigger *specification* and the problem of *recognition* of trigger instances. We have found that in contrast to TLA triggers, trigger specifications can be simple, global, and linguistically authentic. We may therefore assume that the set of triggers is specified by UG. Triggers, then, are small structural templates that are innate, are stored by the language faculty, and constitute the parametric options offered by UG for languages to make use of if they choose to. Each one of these UG-provided treelets serves both as trigger and as the parameter value triggered. As trigger, its occurrence in an input sentence causes adoption of the parameter value; as parameter value, it contributes to licensing (and indirectly to producing and parsing) more sentences. The learner's task is (a) to find these bits of structure in the sentences of the input; (b) to adopt a trigger structure into the current grammar if it is indubitably present in an input sentence; but (c) not to adopt a trigger structure if the input sentence has any analysis, on any grammar not yet decisively ruled out, that does not include it. (a) is the trigger recognition problem, which has not so far been satisfactorily solved. Note that it is not in the first instance

¹² There is other motivation for structural triggers. In Fodor 1992a, 1994, I considered the danger of mis-triggering of parameters by constructional idioms and other exceptional forms. The fact that these exist and are (apparently) not highly constrained creates massive ambiguity for learners. Every trigger for a parameter might instead be an exceptional form to be acquired individually. I proposed as a solution that parameter values have *designated* triggers. A designated trigger is a canonical exemplar of what the parameter value licenses (i.e., all its other properties are unmarked). It is always treated by learners as a trigger, so it is not a possible idiom in any learnable language. All other sentences that could be licensed by the parameter value are treated as nontriggers and learned as special cases, though they may be absorbed under the parameter value if the language also contains the designated trigger. This solves the otherwise puzzling problem of how, for example, a Principle B violation such as *I'm going to make me a sandwich* can occur both in a language that lacks Principle B, and as an exception in a language that respects Principle B. The implementation of designated triggers leads to essentially the same conception of structural triggers/parameter values as the problem of ambiguity between competing parameter values considered here.

a learning problem but a perceptual problem. There would be no difficulty at all about setting parameters if learners could hear underlying phrase structure; they could just read the structural triggers off it. The parsing test for triggers merely substitutes for this ability.

In the ideal parsing test the learner would parse the input sentence just once, without undue effort, and would know as a result exactly what structure or structures it has, and which structural triggers/parameter values are contained in each. This sounds like a lot to ask, but with structural triggers it is easy. This is because structural triggers are ingredients of grammars and ingredients of sentence trees. All the structural triggers in the UG pool can be added into the current grammar to create a larger grammar. The input can be parsed with that grammar, in the normal fashion. The output of the parse will be one or more sentence structures, in each of which it is possible to see the triggers/parameter values that contributed to the parse. The ones that the learner should adopt, if they are not already in the current grammar, are those without which the sentence could not have been generated, that is, any trigger/value that appears in every analysis assigned to the sentence. Parameter values so established will always be correct.

I now describe this procedure with more care for the details. The grammar G for a particular natural language is the sum of (a) all UG principles; (b) a language-specific lexicon (partly constrained by UG, but with idiosyncratic acquired facts about words); (c) universal structural resources provided by UG, such as the X-bar schemata; and (d) a set of parameter values/structural triggers drawn from the larger set made available by UG. A sentence can be parsed with G just in case its structural analysis can be built up from a combination of these bits and pieces that G contains (its lexical items with their projections, X-bar and other such schematic structures, the trigger structures in G) together with their combined entailments under UG principles. Imagine now a grammar that consists of a learner's current grammar but including *all* of the UG-defined trigger structures. For convenience, call this a *supergrammar*. Let us suppose that if the learner's current grammar affords no analysis of an input sentence, the learner tries parsing it with the supergrammar. The only requirement on the parser is that it must record a choice point in the parse, if one arises, where two or more structural analyses present themselves. It must not miss such a point, but it need only note the first one that arises. If the parser has the capacity to compute both (all) analyses in parallel from the choice point to the sentence end, that could provide additional useful information, but it is not essential to deterministic learning. (This is fortunate, since it seems unlikely that the human parsing routines have the capacity for full parallel parsing. Many models of the human parser assume purely serial parsing; some propose partial parallel processing (see Gorrell 1989, Gibson 1991).) If this parser, applying the supergrammar, finds only one analysis for a sentence, that sentence is parametrically unambiguous. In that case it is safe for the learner to adopt any new trigger structure(s) present in the parse tree. If the parser noted a choice point, the sentence may be parametrically ambiguous. In that case no new parameter values should be adopted, even if they appear in the parser's output, unless (a) a value was used in parsing words prior to the first choice point (in which case it must appear in any and all parses that ensue), or (b) the parser fully explored all analyses that presented themselves (in which case a value can be adopted if it appeared in all of them). Otherwise, the sentence can be parsed and comprehended for conversational purposes, but the grammar should not be changed in response

to it. If the learner's attempt to parse the input with the supergrammar is unsuccessful, there are various possible reasons: the input may have been ungrammatical;¹³ or the learner's lexicon may lack an appropriate entry for one or more of the words; or the processing load may have exceeded the parser's capacity, as in the case of multiply center-embedded sentences.¹⁴ Clearly, no parameter setting is warranted.

The two values of a parameter are standardly assumed to be mutually exclusive. This is not a necessary truth. Of course, no one construction can have both values (e.g., no one C can be simultaneously [+fin] and [-fin]), but it does not follow that a language cannot have both values as options (it would be a superset language, subject to the Subset Principle). For example, Chomsky (1993) suggests that Arabic may have both strong and weak Tense features. But suppose for now that this is never the case in natural language. Then a learner that has one value of a parameter, and then adopts the other one, must give up the first (return it to the UG pool); it must have gotten into the grammar as a result of faulty input or perhaps an innate default. There is no problem here. Note in particular that there is no incoherence in supposing that the supergrammar contains both values of a parameter, even if they are mutually exclusive. They are both in the pool that the parser can draw on, and no clash arises unless they are made use of in the same parse (in which case the parse is obviously untenable and can be ignored). This is not the case if parameter values are conceived of as alternative positions of a switch; finding out which is correct by setting the switch both ways at once is obviously not feasible.

Because parameter values are potential building blocks included among the ingredients of grammars, they can be combined into a supergrammar for parsing with. Because they are also ingredients of tree structures, their contributions to the parse can be individually determined. A contributing value will be visible in the tree built by the parser. In fact, it is not even necessary to scan the tree to find the contributing parameter values, because the process of accessing the value for use in the parse could itself be recorded. Structural triggers are closely comparable to lexical entries in the way that they contribute to the parser's task of building up a structure for a word string. All familiar models of the human parser have the capacity to keep track of which lexical entries have been accessed and used in a parse (and which rules, if the grammar contains rules), so we can assume the same is true of the 'lexicon' of trigger structures that UG makes available for parsing with.

¹³ The structural triggers model is not specifically designed to cope with the problem of ungrammatical input. However, it facilitates the kind of frequency tracking that has been proposed as a safety measure; see Kapur 1994. This is because the structural triggers learner knows which parameter values contribute to the licensing of each sentence, and whether there are alternatives to them. Therefore, a parameter value that seems to be motivated, but only very rarely, could be rejected as due to faulty input.

¹⁴ Inability of the parser to compute one of the legitimate parses for a parametrically ambiguous sentence could be a source of learning errors. It is particularly likely, of course, if a child's parser has less computational capacity than an adult's, as seems plausible. Note that any such learning errors would be expected to reflect innate parsing preferences (especially as these are generally considered to be the result of a least-effort tendency). On the other hand, error avoidance requires only detection of the *existence* of alternative supergrammar parses, which is much less demanding of resources than true parallel parsing.

The complexity of the supergrammar parse test is affected very little by how many parameters there are; there is no exponential increase of a sort that would make the process impracticable if it should turn out that 20 is an unrealistically low estimate. Let us suppose that for every parameter there are two trigger structures, one for each value (though in fact there could be just one, its presence or absence in a grammar being the parametric variable). Consider the work involved in parsing a sentence with the learner's current grammar, which contains some number of trigger structures. The effort involved in the supergrammar parse is this plus the effort due to increasing grammar complexity by adding more trigger structures, a maximum of 40 more if there are 20 parameters, and in general up to $2n$ for n parameters. The extra parsing cost would be comparable to that due to adding 40 (or 60 or 200) more entries to the lexicon, which cannot be very great or else people with large vocabularies would parse sentences much more slowly than people with small vocabularies. (Preschool children are reported to acquire nine new lexical items per day on average for four years.) Like lexical entries, trigger structures are resources that the parser can draw on if it needs to; if they are not needed, they do not enter into the parsing process, and so their effect on workload is slight. In the worst case parsing complexity is a function of the square of the size of the grammar (Earley 1970), so there is an increase but no exponential growth in processing load as grammar size increases.¹⁵

Because of this, the learner could just as well dispense with the initial parse test that checks to see whether the current grammar already licenses the input. It could simply parse *every* sentence with the supergrammar. If this results in an analysis involving no triggers beyond the current grammar, the grammar would be left unchanged; otherwise, parameter value adoption would occur as already described. Thus, it is not necessary to make the assumption that a child can hold an input string in memory long enough to repeat the parsing process if not successful on a first try. One parse is all that is ever needed, for comprehension of familiar structures and for learning of novel ones. A welcome aspect of this approach is that a child never has more work to do than adult perceivers do. When the input is parametrically unambiguous, the trigger structures that the learner's parser calls on in the course of the parse test will be just those that adults use in parsing the sentence; the workloads are identical. Only for parametrically ambiguous inputs does a learner have more parses to consider than adults do. But these are cases where learning must *not* occur; so, as observed, the learner can interrupt the computation as soon as the existence of a second analysis has been detected. With structural triggers, therefore, we can avoid attributing to children psychological processes that are of greater complexity than adults are known to be capable of.

¹⁵ The supergrammar may shrink over time. Once a parameter value is established, the other value of that parameter can be erased from the UG pool and the supergrammar. The advantage would be a reduction of parametric ambiguity in later input sentences, making more of them usable for the setting of other parameters, and a decrease in the parser's workload. (This contrasts with the TLA; see footnote 9.) However, eliminating a parameter value from all further consideration is a serious matter, since even a conservative learner could be in error owing to faulty input or parsing mistakes. We may suppose that to give itself a chance to put such errors right, the learner keeps records of frequency of use of each parameter value and waits until the imbalance between two competing values mounts high enough for confidence in accepting one and permanently rejecting the other; see footnote 13. Thus, safety could be paramount at first, but efficiency maximized later. In this respect the structural triggers model offers an implementation of Valian's "balance" model discussed in section 1.3.

To summarize: The most general principle for grammar change is *Adopt a parameter value/trigger structure if and only if it occurs as a part of every complete well-formed phrase marker assigned to an input sentence by the parser using the supergrammar*. To the extent that the human parsing routines lack the capacity for full parallel parsing, this formula will generally be applied in the more modest form, *Adopt a parameter value/structural trigger if and only if it occurs as a part of a unique complete well-formed phrase marker assigned to the input by the parser using the supergrammar*. This wastes a little information that the input could potentially supply, but not a great deal: it means that the learner extracts no information about one parameter from an input that is ambiguous with respect to some other parameter. However, in other respects information is not wasted. Unlike what happens with the TLA, almost no input sentence goes by unused because the learner could not find a grammar to parse it. Moreover, multiple parameters may be set at a time, as long as the input sentence provides clear evidence for all of them. For instance, a *V S O* input could set all three word order parameters, since it has a unique parse that includes trigger structures for them all. Thus, there is no need for a Single Value Constraint (though learners may be hesitant in practice to acquire too many new facts in a single learning event; see Berwick 1985:chap. 2). The structural triggers learner acquires the correct target grammar very efficiently, because it makes good use of the information provided, and also because, unlike the TLA, it establishes the values of individual parameters rather than evaluating whole grammars. Learning is once more just a matter of answering 20 or so questions, as Chomsky envisaged when he presented the principles-and-parameters theory as a solution to learnability as well as linguistic problems.

2.3 *The Sufficiency of Unambiguous Triggers*

The structural triggers learner (henceforth STL) learns only from unambiguous sentences; it throws away ambiguous input in order to avoid making errors. It is critically important, therefore, that there be sufficient information in unambiguous sentences to set all the parameters that need to be set. Is there any reason to think this is the case? For convenience in what follows I will use the term *trigger* both for structural triggers and in a secondary sense to denote a *sentence* whose structure contains the structural trigger in question. For example, the sentence *Mary me saw*, whose structure contains the structural trigger for – V2 (i.e., the lexical verb, or finiteness feature, in I not C) can be called a trigger for – V2. A trigger in this sense, if parametrically unambiguous, is a trigger by GW's definitions in (2).

Note first that the STL does not suffer from the “missing triggers” problem that causes learning failures in the TLA. There are no local maxima to trap this learner, because it is not garden-pathed by ambiguity, and it has no Single Value Constraint (see the appendix). The danger for the STL is that there might be cases where only ambiguous sentences provide the evidence that determines the right grammar. There are several ways this might come about. A familiar example is the subset value of a subset/superset parameter. Every sentence of the subset language is parametrically ambiguous since it is compatible with either value of the parameter; so the subset parameter value has no unambiguous triggers. The classic solution is to assume that the subset

value is the innate default, so that it is unnecessary for learners to acquire it from the input. This approach could be extended to non-subset parameters also, if one of the values lacks unambiguous evidence. (As usual, a default postulated for learnability reasons should be compatible with linguistic evidence of markedness.)¹⁶

However, there are some cases where even an innate default cannot make up for missing input evidence. The general form of the problem is that the evidence that determines a parameter value may be a *combination* of inputs, each of which is ambiguous in itself. The word order domain offers an example. A good cue for the positive setting of the V2 parameter is the cooccurrence of *SV(O)* and *OV(S)* sentences in the same language.¹⁷ Normally this would not be the *only* evidence a language affords for +V2, but the situation is possible: it would be the case in a simple language with no constituents other than subjects, direct objects, and main verbs. But now note that since *SV(O)* and *OV(S)* are each individually compatible with –V2 grammars as well as with +V2, they are both parametrically ambiguous. A learner that rejected all ambiguous inputs would therefore reject both of these sentence types and would be left with no basis for acquiring +V2. (In fact, it would have no basis for learning anything in this case.) Note that +V2 cannot be assumed to be the default, because in this very simple domain a +V2 language is a proper superset of another language (the *SV, VO, –V2* language). Thus, +V2 cannot be learned from the input and it cannot be preset. Though this illustration involves an exceedingly impoverished language, it serves as a warning to check whether a comparable situation could arise for any parameter in natural languages. If it does, a learner that ignores ambiguous inputs would fail to learn.

It is not possible to give a general proof that natural languages never present such problems. To establish that there is always sufficient unambiguous input for learning, it would be necessary to show that for each nondefault parameter value, every language which has that value contains at least one (simple, common) parametrically unambiguous sentence type that could establish it. But ambiguity is not a systematic phenomenon. It is a matter of coincidence of two analyses, so it occurs haphazardly. Therefore, every parametric domain in natural language needs to be checked individually to make sure enough information is present. Outcomes cannot be anticipated here.

¹⁶ If there are innate default values, the parameter-setting mechanism is not fully deterministic in the sense of all settings being indelible (Marcus 1977); default values must be alterable, even if values set by learning are not. In fact, the STL would gain nothing from the imposition of an absolute prohibition against resetting acquired values. That would only impede error correction; see footnote 15. Thus, the STL is not strictly deterministic, though it revises its decisions considerably less often than explicitly probabilistic devices such as the TLA.

¹⁷ Variability of what precedes the verb is a highly characteristic property of +V2 languages. However, a combination of sentences such as *SV(O)* and *OV(S)* cannot be a trigger; a trigger must be a single sentence. This is a significant consequence of the standard assumption that there is no memory for past inputs. Learners are thereby deprived of forms of evidence such as patterns of alternation that are routinely employed by linguists. There are some indirect means by which information inherent in relationships among sentences could be exploited. For example, if reversion to a previously held parameter value were prohibited, a parameter could not flip-flop repeatedly. If the specifier-head parameter had first been set to *SV* and was then reset to *VS*, the next *SV* sentence would necessarily signal +V2. This presupposes nondeterministic learning. The STL might make use of frequency tracking of alternative parameter values (see footnote 13). If +V2 were successful in the supergrammar parse test more often than either *SV* or *VS* alone, the ambiguity might eventually be resolved in favor of +V2. (In fact, natural languages contain unambiguous triggers for V2, so these strategies are not essential here, but they might be used in other cases.)

Table 1

Some unambiguous word order triggers

Grammar	One parametrically unambiguous sentence pattern	Example sentence
SV, OV, -V2	S O V	Mary me saw.
SV, VO, -V2	Adv S V O	Soon Mary saw me.
VS, OV, -V2	Adv O V S	Soon me saw Mary.
VS, VO, -V2	V O S	Saw me Mary.
SV, OV, +V2	Adv Aux S O V	Soon will Mary me see.
SV, VO, +V2	Adv Aux S V O	Soon will Mary see me.
VS, OV, +V2	Adv Aux O V S	Soon will me see Mary.
VS, VO, +V2	O1 Aux V O2 S	Books will give me Mary.

However, if the word order domain is any guide, the existence of parametric ambiguity does not entail a lack of unambiguous triggers. For the three word order parameters that GW studied, and considering all the sentence patterns in their table 3 (of single-clause sentence types in the eight languages), an average of 58% of the items a learner would be exposed to are ambiguous with respect to at least one parameter; yet there is still more than enough unambiguous information. For brevity I present in table 1 just one parametrically unambiguous sentence pattern for each language (illustrated with English lexical items) such that this sentence fixes the value of every parameter for that language. In each case there are other unambiguous triggers as well; this is a sample only. Since input is rarely wasted by the STL, even one instance of an unambiguous trigger is enough to set a parameter. Also, a sentence need not be unambiguous with respect to *all* parameters in order to be useful. A sentence that is parametrically ambiguous with respect to parameter *P1* but unambiguous with respect to *P2* can be used to set *P2* once *P1* has been set (or if parallel parsing is possible). The STL makes no grammar changes until it receives sentences such as those in table 1. Some of the sentences in table 1 are quite complex, and perhaps not encountered in the earliest stages of acquisition. This might appear problematic in two ways. It might seem to predict delayed acquisition of word order by a conservative learning system. And it might seem to leave learners with no grammars to use at all until definitive triggers are encountered. Both of these concerns can be put to rest.

First, not all unambiguous sentences are as complex as the examples in table 1. A word order trigger could not be less than two words long; for the complement-head parameter it must be at least three words long. The triggers in table 1 range from three to five words. And there do exist unambiguous two-word triggers such as *V S* and three-word triggers such as *S O V*. However, some complexities are unavoidable. Generally, and for obvious reasons, it takes longer sentences to establish the order of verb and object than the order of verb and subject, longer sentences to establish -V2 in SV languages than in VS languages, and longer sentences to establish +V2 than -V2 (since +V2 is evidenced by sequences that could arise only by movement, such as separation of object or Aux from the main verb). But these, of course, are facts about languages, not about a particular learning system. A nondeterministic learner would in large

part be influenced by the same triggers as a deterministic learner; the only difference is that the nondeterministic learner also acts in response to nontriggers that intervene between the unambiguous triggers, whereas the deterministic learner sits quietly and lets them go past.¹⁸ If unambiguous triggers are complex or infrequent, both types of learner would be affected.

The second concern is what grammar a conservative learner could use while waiting to encounter definitive triggers. This is an empirical question, and a deterministic learning model is not committed to any particular answer. There are several possibilities. One is that some or all parameters are in a specific “unset” state prior to unambiguous triggering. This would mean that either value of the parameter could be used freely by the child for purposes of language production and comprehension. For word order this would be equivalent to a policy of free word order prior to parameter setting. The learner’s language would be a superset of the target language, but the usual subset/superset problem would not arise as long as parameter setting was mandatory on encountering an unambiguous trigger. A learner who had observed object NPs only in *SVO* sentences, which do not disambiguate between the VO and OV values of the complement-head parameter (unless –V2 has been established), could utter sentences with either VO or OV orders, but only until hearing an unambiguous trigger such as *Adv SVO*; from then on, only VO orders would be produced. Perhaps this course of events does not comport with empirical reports of word order acquisition. However, it is not out of the question that learners normally hear unambiguous triggers for setting a parameter before starting to utter sentences that make use of it; in that case no production errors would occur. Another possibility, noted above, is that parameters have default values that are retained until the input provides decisive evidence to the contrary.¹⁹

In sum: At least for GW’s word order domain, once ambiguous inputs are screened out by the supergrammar parse test, the learnability problem disappears. All risks are eliminated if learners wait for decisive evidence. Moreover, this “delay strategy” for dealing with ambiguity (see section 1.1) has no obvious disadvantages. There is no reason to suppose that it slows down or prevents attainment of the target grammar. Though the number of trigger types is reduced, their quality is improved, so less effort is wasted making wrong grammar changes and correcting them.

¹⁸ The TLA, on the other hand, lets many *good* triggers go past. The TLA requires far more inputs for convergence than the STL does, because of the high rate of TLA parse test failure: very often the parameter value randomly selected for testing does not license the input sentence. In that case, even though the input may be a fully unambiguous trigger, no grammar change occurs. Many tokens of the same sentence type must typically be received before the grammar is changed. (See Fodor, in press a, for discussion.) Interestingly, this failure rate is higher the more optimal the language domain is—that is, the lower the incidence of parametric ambiguity. (Also, the less ambiguity there is, the greater the chance of local maxima.)

¹⁹ I note in the appendix that default values are potentially dangerous. A default value is an “error” from the point of view of at least some target grammars, and errors can create more errors (even superset errors) under the influence of certain local triggers. Also, as GW note, a default is not stable in a system that guesses. This is why their proposed –V2 default to solve the local maximum problem is ineffective without the additional assumption that the marked +V2 value is inaccessible to children until a certain stage of maturation. (This would stabilize the default but would avoid local maxima only if the maturational change occurred late enough that learners of –V2 languages were certain to have all the other parameters correctly set by then, but not so late that learners of +V2 languages would be significantly delayed in acquisition; see Bertolo 1995 for discussion.) But defaults have neither of these drawbacks in the STL model. Defaults are safe (because the supergrammar test reveals when the nondefault value might be correct and thus prevents grammar changes based on false presumptions) and stable (because a default is retained until the input unambiguously signals the opposite value).

(See also footnote 18.) And although no general proof is possible, we have seen that a language domain that contains parametrically ambiguous sentences can still provide ample unambiguous information for setting all parameters. This is encouraging, since it confirms that the source of the learning problem that GW discovered was the learning algorithm, not natural languages themselves. Despite ambiguity, the information learners need is not actually missing; it merely seems to be so when it is used unselectively. To benefit from it, a learner need only refrain from guessing when the information is not decisive.

Appendix: Aspects of the Triggering Learning Algorithm

A. Local and Global Triggers

Unlike the STL, the TLA does not know, when it sets a parameter, whether or not it was justified in doing so. This has some undesirable consequences. One is that the TLA cannot dismiss the opposite value of the parameter from further consideration; indeed, it can never dismiss any parameter values. This is why it must keep searching through all parameter value combinations. Another consequence is that the TLA cannot use certainty about one parameter to disambiguate evidence concerning another one. The latter point arises in connection with GW's definition of local triggers in (2b). The definition appears to be intended to characterize only unambiguous local triggers (in contrast to the triggers actually employed by the TLA), but in fact the triggers so defined would be ambiguous from the point of view of any learner that was uncertain of the values of other parameters. Therefore, even if the TLA could somehow be given the ability to restrict itself to the triggers defined by (2), it would still be error-prone.

This problem arises only for local triggers, not for global triggers. However, it cannot be sidestepped by assuming that learners could rely exclusively on global triggers. By (2a), a global trigger for $P_i(v)$ is present in every language whose grammar has $P_i(v)$; by (2b), a local trigger for $P_i(v)$ is present only in some languages whose grammars have $P_i(v)$, depending on what the values of other parameters are.²⁰ GW have demonstrated that a parameter may have no global triggers in the sense of (2a). None of the three word order parameters they consider has global triggers under (2). (They all have global structural triggers, though; see section 2.1 above.) Hence, most or all of the burden of parameter setting *must* be carried by local triggers, under GW's assumptions. I explain here how local triggers can simulate ambiguity for learners. I then show that only a certain subclass of local triggers does so. Definition (2b) can be reformulated to exclude that subclass.

²⁰ Every global trigger qualifies also as a local trigger (GW, p. 409), but for present purposes it is convenient to think of global and local triggers as disjoint; no matters of substance are affected. The difference in reliability between local and global triggers may be masked by the more salient difference that a local trigger does less work than a global trigger (GW, p. 417). A given local trigger for $P_i(v)$ is not present in all $P_i(v)$ languages, so there must be other triggers for setting $P_i(v)$ that do occur in those languages. Thus, it may take several local triggers to do as much parameter setting as one global trigger. But this is not a serious defect (in a model in which triggers are not mentally stored; if they were, local triggers would escalate storage costs). For a learner with an on-line test for triggers, reliance on local triggers does not increase the workload of language learning in any way.

Local triggers are not unambiguous for learners despite the “only if” clause in (2b), which appears to demand that $P_i(v)$ be the *only* way sentence S can be licensed (given values for the other parameters). By (2b), whether a sentence is a local trigger for $P_i(v)$ depends on how the other parameters are set. But during the course of learning the learner inevitably has some parameters set wrong. Therefore, the dependence relations for local triggers may be miscalculated. Suppose a sentence S is a local trigger for $P_i(v)$ when the other parameters have a certain array of values W , but not when they have other arrays of values, which I will designate collectively as $\sim W$. Now suppose that a learner’s current grammar has the values $P_i(v')$ and W , while the target grammar has $P_i(v)$ and some collection of values in $\sim W$. The learner’s current belief is that the parametric context is W . On encountering S , the learner could therefore switch the value of P_i from v' to v . But in fact it was W that was wrong; P_i was correctly set before and resetting it is an error, a retrograde move.

The word order domain provides an illustration. Suppose that the current grammar has SV, OV, and $-V2$, that the target grammar has SV, OV, and $+V2$, and that the input is an SVO sentence. In the context of $-V2$ and SV, SVO is an unambiguous local trigger for VO. In the terms of (2b): given the values $-V2$ and SV, SVO is a sentence of the target language that is grammatical if and only if the value for the complement-head parameter is VO. Thus, even the most conservative learner would seem to have grounds for switching OV to VO. But the resulting grammar would be SV, VO, and $-V2$, which is further from the target than before.

Global triggers are context-free triggers, not dependent on the values of other parameters. Local triggers are context-sensitive, and an error about what the parametric context is can cause errors on other parameters, as we have just seen. If even one error is present (i.e., at all times until language acquisition is over), that error can be parlayed into more errors (even incurable superset errors) by local triggers. This offers a clear illustration of how garden paths can grow once an error has occurred. For a nondeterministic system that typically makes many errors en route to the correct grammar, local triggers could considerably magnify the inaccuracies due to other causes. And even if a learner could avoid other mis-triggerings, local triggers could create errors on the basis of default values. If, for instance, the $V2$ parameter were initially set at $-V2$ by default, then an SVO input could mis-set the complement-head parameter for an OV, $+V2$ target language as just illustrated.

In short: local triggers must be employed, but local triggers breed errors. It might seem fruitless, therefore, even to try to eradicate learning errors; nondeterministic learning would appear to be the only choice. But that would be an unduly pessimistic conclusion. Not all local triggers have ill effects. If the “good” local triggers can be identified, and learners can be limited to them, errors would not multiply.

All local triggers are context-sensitive, but there are two kinds of context-sensitivity and only one of them creates errors. We are assuming S to be a local trigger for $P_i(v)$ in the context W but not in contexts $\sim W$. The question is, what happens in $\sim W$? If S does not occur at all in these contexts, then S will never cause a problem; whenever it does occur, it is a completely reliable cue for setting P_i to value v . For example, an SVO sentence is a local trigger for the parameter value OV in the context of the values SV and $-V2$, but in other parametric contexts

(i.e., if VS and/or +V2) *S O V* sentences do not occur. Therefore, any learner encountering an *S O V* sentence could confidently shift to (or retain) the setting OV of the complement-head parameter. This local trigger (unlike the local trigger *S V O* in the example above) is just as reliable as an (unambiguous) global trigger is, even for a learner that has wrong settings of other parameters. I will refer to local triggers of this kind as *reliable local triggers*. They are only local, not global, because their existence is context-restricted, but their cue value is not context-dependent. Clitic climbing is a local trigger in this sense for the positive value of the null subject parameter, assuming that clitic climbing occurs only in null subject languages (Kayne 1989). It is not a global trigger for the value [+null subjects] because it is not available in null subject languages that have no clitics; but when it does occur, the value [+null subjects] is certain.

Now consider the case where *S*, which is a local trigger for $P_i(v)$ in the parametric context *W*, occurs also in languages with parameter value combinations in $\sim W$. If even one such language has $P_i(v')$, not $P_i(v)$, then *S* would not be a reliable cue for $P_i(v)$. It would not be possible for a learner to tell, from an encounter with *S*, whether or not P_i has the value *v* (unless it were known whether the target grammar has *W* or $\sim W$; but this cannot be assumed). Nevertheless, *S* counts as a local trigger for $P_i(v)$ under GW's definition. For example, given the parametric context -V2, an *S V O* sentence qualifies as a local trigger for the parameter value SV. Yet *S V O* also occurs in +V2 languages, including +V2 languages that have VS, so the occurrence of *S V O* is not a reliable indicator of an SV grammar. *S V O* is a local trigger but not an unambiguous trigger. Clearly, it is this kind of local trigger that is the source of errors. A learner whose current grammar has -V2 could interpret an encounter with *S V O* as evidence for the setting SV; but if the target is a VS, +V2 grammar, this would be wrong. I will call local triggers of this type *unreliable local triggers*. They are dangerous for learners because their validity as evidence of parameter settings is context-dependent.

GW's division of triggers into global and local is illuminating. The present discussion has added to it a further division within the local triggers. Let us restrict attention to triggers that are unambiguous in the sense that the parameter value triggered is necessary for licensing the trigger sentence (either globally or in a particular context). Within these unambiguous triggers, there is now a three-way classification into global triggers (globally available and globally valid), reliable local triggers (locally available, valid when they occur), and unreliable local triggers (locally or globally present, but only locally valid). An encounter with a global trigger or a reliable local trigger guarantees (if the input is veridical) that the language has the parameter value in question. This is not so for the unreliable local triggers. Whether or not the latter should be employed as triggers for setting parameters depends on the goals of the learning model. For avoiding the local maxima problem (i.e., the threat of missing triggers for some grammars), the primary concern is trigger availability; so including the unreliable local triggers could be helpful. For avoiding errors, on the other hand, the paramount concern is trigger reliability; so the learner would be better off using just unambiguous global triggers (if any) and the reliable local triggers. (However, unreliable local triggers could be used safely in some circumstances by a learner like the STL, which does sometimes know, at least with a reasonable degree of certainty, which other parameters are correctly set. See footnote 15.)

The formal definition of local triggers can be made to distinguish reliable from unreliable local triggers. (2b) can be amended as in (8).

- (8) Given an array W of values for all the parameters except one, parameter P_i , such that a sentence S from the target language L is grammatical only if the other parameters have the values in W , S is a *reliable local trigger* for $P_i(v)$ just in case S is grammatical if and only if the value for P_i is v .

The combination of (2a) and (8) now captures all the fully trustworthy triggers, global and local. A sleeker version of these definitions is given in (9).

- (9) a. A *global trigger* for $P_i(v)$ is a sentence S that is grammatical in a language $L(G)$ if and only if G has P_i set to v .
 b. A *reliable local trigger* for $P_i(v)$ is a sentence S that is grammatical in at least one language, and is grammatical in $L(G)$ only if G has P_i set to v .

A learner that encounters a trigger for $P_i(v)$ in the sense of (9) is guaranteed (input errors aside) that $P_i(v)$ is correct.

Though the trustworthy triggers have now been defined, it is clear that the TLA is no more able to identify and use them than it was before. The TLA adopts $P_i(v)$ in response to any sentence that can be parsed with $P_i(v)$ and not without it, given a certain set of values for the other parameters. That is, it adopts $P_i(v)$ when it has established that $P_i(v)$ is a *sufficient* condition for the grammaticality of S , in the current parametric context. Because it does not try out other parameter combinations on S , it cannot establish that $P_i(v)$ is a *necessary* condition for the grammaticality of S , in that parametric context or any other. Thus, the general conclusion about TLA triggers is unchanged by the new definition: the TLA cannot avoid mis-setting parameters, because it cannot distinguish trustworthy from misleading input, because its parsability test for triggerhood is not powerful enough. (9) confirms that the TLA's trigger identification problem is *merely* a practical matter of test power. There is no in-principle problem about identifying perfect triggers, given suitable resources.

B. Greediness and the Single Value Constraint

I have argued that the lack of a practical parsing test for parametric ambiguity is the source of the learnability problem that GW have identified. It is the reason why the TLA could not be required to use only the unambiguous triggers characterized by GW's definitions. The intrusion of ambiguous triggers creates uncertainty that precludes decisive parameter setting. The learner must therefore resort to a semirandom walk through the space of possible grammars. The attempt to constrain this by means of the Greediness Constraint and the Single Value Constraint creates local maxima: the search may reach a dead end at an incorrect grammar that affords no route to the correct one. I show now that the weakness of the TLA's parse test is also the reason why Greediness and the Single Value Constraint cause local maxima. These constraints do not discriminate between situations where limiting the learner's search path is helpful and situations where it only makes good grammars inaccessible. However, if the parse test were more powerful, the

constraints could be reformulated so as to respect this distinction, and then they could do good without doing harm. The local maximum problem would disappear. Even within a TLA-style random walk model, therefore, upgrading the TLA's parse test would be beneficial.

GW consider refinements of both constraints but judge them impracticable. The problem is that implementing more optimal versions would require that the learner be able to compute more than one parse of the input at a time, and that it be able to tell which parameter value(s) were responsible for a successful parse. These are exactly the properties that we established (section 1.3) to be necessary for deterministic learning in the face of ambiguous input, but that the TLA's parse test cannot deliver. The fact that they also impede GW's proposed constraints strongly suggests that this parsing test is a bottleneck that will hinder any attempt to make the TLA less error-prone.

The Greediness and Single Value Constraints follow from the TLA (though GW also state them separately). The TLA incorporates three significant constraints on grammar change. It entails error-driven learning: the learner never gives up a superficially successful grammar but only grammars that must be wrong because they fail to license a sentence of the target language (the current input). This seems a desirable constraint to impose on any learning device. The TLA also entails that the learner shifts to a new grammar only if it provides a successful parse of the current input. This is the Greediness Constraint. This also seems helpful for any learning system (deterministic or not), despite the adverse effect on learnability that GW's investigations have uncovered. It is not guaranteed that the currently successful grammar is the right grammar. Nevertheless, together with error-driven learning, Greediness ensures that all grammar changes represent at least apparent progress: they are all shifts from certainly wrong grammars to possibly right ones. In fact, the Greediness Constraint does no great harm (but see Berwick and Niyogi 1996) except in combination with the Single Value Constraint.

The Single Value Constraint permits only one parameter to be reset per learning event (i.e., on each encounter with an input sentence). This follows from the fact that the TLA tries out only one novel parameter value per learning event. The effect of the constraint is that the learner will tend to explore one local neighborhood within the parameter space before venturing further afield; the learner adopts grammars close to those that were previously (apparently) successful, and it can crawl only slowly away toward more dissimilar grammars. In a deterministic system this constraint is not needed. As long as the evidence is decisive, there can be no harm in resetting several parameters at the same time. For the STL, for instance, the Single Value Constraint would be a pointless encumbrance. How essential the Single Value Constraint is to a trial-and-error system is not entirely clear. GW maintain that it is beneficial, but the reasons are not spelled out in detail. What savings it brings in terms of the speed or accuracy of learning is not documented. (Berwick and Niyogi (1996) claim that it slows learning down even when there are no local maxima.) Informally, it seems that it would be most advantageous late in the learning process. If a learner is only one parameter value away from the correct grammar, resetting two parameters at once is bound to involve changing at least one of them from the right value to the wrong one; the result would be either one or two wrong values. Thus, the change could not in principle be an overall improvement: the new grammar might be worse than before, but could not be better.

The risk of doing more harm than good is limited in part by the Greediness Constraint, which demands at least superficial success, but it could be minimized by resetting no more parameters than are currently mis-set. The learner cannot know how many are currently mis-set; it knows only (from failure of the parse with the current grammar) that at least one is. So the safest course would be to reset just one.²¹

At other stages the Single Value Constraint appears to be overly stringent. It makes for slow learning early on when there are many parameters still to be set. Suppose that the target is English (SV, VO, and -V2), that the learner's current grammar has VS, VO, and +V2, and that the input is of the form *Adv S V O* (e.g., *Probably Mary saw me*), which clearly demands -V2 and SV. Under the Single Value Constraint, the learner cannot use this input to reset these two parameters. It has wandered too far afield from the target and is not permitted to snap back to the correct grammar even though this is the *only* grammar that could possibly license the current input. The troublesome interaction of the Greediness Constraint with the Single Value Constraint can also be seen here. To change +V2 to -V2 and VS to SV simultaneously is prohibited by the Single Value Constraint, but to change them one at a time is prohibited by the Greediness Constraint. This is because neither resetting +V2 to -V2 nor resetting VS to SV would be sufficient by itself to license the input sentence. In general, whenever one input sentence signals the need for more than one parameter change, the two constraints conspire to paralyze the parameter-setting mechanism completely. If all the triggers for a target parameter value have this property, the learner may be trapped permanently on a garden path. Escape is possible only if the two parameter changes can be decoupled, that is, if the language contains "intermediary" sentences that exhibit one parameter value but not the other. Such sentences function as stepping stones, permitting the target grammar to be attained one parameter value at a time. The major result in GW's article is that languages (at least in this limited domain of word order variation) do not always provide these intermediary triggers. GW prove that for 6 of the 56 possible transitions from one of the eight grammars to another, the necessary triggers do not exist. The grammars from which the shifts are impossible are the local maxima. Four of the eight grammars are local maxima relative to at least one potential target grammar. These facts are testimony to the combined power of Greediness and the Single Value Constraint.

The Single Value Constraint also threatens the initial step of parameter setting in GW's system. At the onset of learning it happens that *every* input sentence requires at least two parameters to be set. In languages without null subjects, as in the example domain, the simplest possible sentence type is *S V* or *V S*; and generation of either of these requires both the specifier-head parameter and the V2 parameter to be set. To allow learning to proceed at all, GW are therefore forced to assume that at least one of these two parameters has an innate default. As it happens,

²¹ The Single Value Constraint also provides a trial-and-error learner with a relevance test for parameters. If changing the value of just one parameter *P* turns a parsing failure into a parsing success, then *P* must be relevant to that input. For unambiguous input it follows that the new value of *P* is correct. If the input may be ambiguous, this is not guaranteed; but still, limiting resetting to relevant parameters does prevent many pointless changes, some of which might mis-set a parameter that was correctly set before. If more than one parameter were reset at the same time (except under the revised constraint discussed below), success in the parse test would show only that at least one of them was relevant; it would not reveal which. See Fodor, in press b, for discussion.

this is not unwelcome, since the assumption of a $-V2$ default for the $V2$ parameter offers a possible solution to the problem of local maxima in GW's system. However, GW also observe (p. 442) that a different way out of this problem would be to make the Single Value Constraint more flexible. It could permit just one parameter to be reset if that is sufficient, or two if it is not, or three if three are necessary, and so on. In other words: the learner would be allowed to reset just the *minimum* number of parameters necessary to license an input.

However, this "smart" version of the Single Value Constraint cannot be implemented in GW's model because it would require too many parses of some input sentences. With the TLA's Single Value Constraint unmodified, the learner would parse each input just once. The same would be true if there were no Single Value Constraint at all; the only difference is that the one test-parse could be used to try out any number of new parameter values at a time, rather than just one. The computational load would be the same either way, since parsing a sentence with 2 (or 3 or 17) parameters reset takes no more effort than parsing it with 1 reset. However, test-parsing it with more than one *grammar* does consume resources, and this is what the flexible Single Value Constraint would require: the learner would have to check all grammars that differ by one value from the current grammar, to be sure they all fail, before trying out a grammar at a remove of two values, and it would have to check all of these before considering grammars three values away, and so on. All this must be done on the same input. It cannot be spread over successive inputs, because of the assumption that the learning device has no memory for prior learning events. If it tested only one grammar per input, it would never accumulate enough information to support a universal quantification over all the one-parameter changes. Hence, it would never be justified in setting more than one parameter; de facto the old Single Value Constraint would still apply and nothing would have been gained. The STL does not have this limitation, because it can test multiple grammars at one time. It might compare parses and pick the least novel one. Alternatively, there might be a penalty for using trigger structures not yet adopted into the grammar, so that the parser would naturally favor the least novel analysis. In some such way the STL could respect the more flexible Single Value Constraint if there were reason for it to do so. GW contemplated the idea that the TLA should be allowed to try out multiple grammars on the same input during a single learning event, but they judged that this would require "too much processing to be realistic" (p. 442). In short: a useful revision of the Single Value Constraint is impracticable for the TLA because of the parse test bottleneck.

The same is true for Greediness: a more flexible variant of it would eliminate the missing-triggers problem, but the parsing test is too weak to implement it. The Greediness Constraint as it stands requires that the *whole* of an input sentence be parsable under a new parameter setting. This complete-parse requirement is a close relation of the complete-sentence requirement in GW's definition of triggers (section 2.1 above). Both have the disadvantage that they can impede grammar change even where the evidence for the change is excellent. As we saw above, the shift from $+V2$ to $-V2$ in response to *Adv S V O* is prohibited unless the other parameters are set in such a way that every aspect of the input string is accounted for on the new parse. GW observed that the Greediness Constraint might be relaxed to allow a parameter value to be adopted if it affords a *partial* analysis of the input. They considered several variants of this approach (e.g., successful

analysis of the beginning of the sentence, success on the initial and final constituents) but found none of them satisfactory. They were looking for a metric for ranking grammars so that the Greediness Constraint could allow the learner to shift from a grammar that is less successful at parsing the input to a grammar that is more successful at parsing it. They concluded (p. 449), “[W]e have not been able to identify such a metric that is simple, effective, and psychologically plausible. An important open question is whether there is any such metric that avoids local maxima for all possible linguistic parameter systems.” I suggest that if there is one, the likeliest candidate is one that is tailored to the particular parameter being considered: it would allow a parameter value $P_i(v)$ to be adopted just in case the new grammar affords an analysis of that part of the input sentence to which $P_i(v)$ is relevant (i.e., to the analysis of which the value of P_i could in principle make a difference).

This “smart” Greediness Constraint would allow a learner to ignore unfamiliar parts of sentences such as tag questions or adverbial clauses or any other constituents not yet mastered, while learning from the rest of the sentence. It would also allow a learner recovering from a garden path to get back onto the right track one parameter value at a time. For instance, it would allow the complement-head parameter to be reset if resetting provided a partial parse that accounted successfully for the order of verb and object, even though other facts such as the position of the subject remained a mystery. (For example, if the target were SV, VO, – V2, and the current grammar were VS, OV, – V2, the sequence *S Aux V O* could switch OV to VO even though that would leave the specifier-head parameter wrongly set at VS.) This version of Greediness would also permit the previously prohibited shift from + V2 to – V2 in response to *Adv S V O*, regardless of how the specifier-head and complement-head parameters were set.

The flexible Greediness Constraint is feasible in the structural triggers model, in which the parser can detect that a parameter value is essential for parsing one portion of a sentence even if the structure of other parts is unclear. But it is not an option in GW’s model. This is because it requires knowledge of relevance relations between parameter values and aspects of sentence structure. But as GW note (p. 410), “[U]nder the TLA, the learner has no knowledge of which parameters are relevant to a given input. . .” GW reject the idea that information about how parameter values relate to sentences is innately listed or can be deduced from UG by the learner (for reasons outlined in section 1.3). Yet these mechanisms provide information about relevance that is not provided by the successful-parse criterion that GW adopt instead. The TLA’s parse test treats parameter values as contentless switches and merely waits to see what effect they will have when they are turned on. Furthermore, it registers these effects by a simple binary distinction: either the parse succeeds or it fails. No account is taken of what aspects of the syntactic structure a given parameter value makes available. Thus, the learning device has no insight into where and how a parse succeeded or failed. A useful refinement of the Greediness Constraint is thus ruled out by the uninformative nature of the parse test.

Conclusion: Even if the goal is to make GW’s nondeterministic learning model more efficient, rather than to design a deterministic alternative, strengthening the on-line parsing test for triggerhood is the most effective way of improving the performance of the learning device.

References

- Bertolo, Stefano. 1995. Maturation and learnability in parametric systems. *Language Acquisition* 4:277–318.
- Berwick, Robert C. 1985. *The acquisition of syntactic knowledge*. Cambridge, Mass.: MIT Press.
- Berwick, Robert C., and Partha Niyogi. 1996. Learning from triggers. *Linguistic Inquiry* 27:605–622.
- Chomsky, Noam. 1986. *Knowledge of language: Its nature, origin, and use*. New York: Praeger.
- Chomsky, Noam. 1993. A minimalist program for linguistic theory. In *The view from Building 20: Essays in linguistics in honor of Sylvain Bromberger*, ed. Kenneth Hale and Samuel Jay Keyser, 1–52. Cambridge, Mass.: MIT Press. [Reprinted in Noam Chomsky, *The Minimalist Program*, 167–217. Cambridge, Mass.: MIT Press (1995).]
- Clark, Robin. 1989. On the relationship between the input data and parameter setting. In *NELS 19*, 48–62. GLSA, University of Massachusetts, Amherst.
- Clark, Robin, and Ian Roberts. 1993. A computational model of language learnability and language change. *Linguistic Inquiry* 24:299–345.
- Earley, J. 1970. An efficient context-free parsing algorithm. *Communications of the ACM* 13:94–102.
- Fodor, Janet Dean. 1989. Principle-based learning. In *CUNYForum 14*, 59–67. Ph.D. Program in Linguistics, Graduate Center, CUNY, New York.
- Fodor, Janet Dean. 1992a. Designated triggers versus the Subset Principle. Ms., Graduate Center, CUNY, New York.
- Fodor, Janet Dean. 1992b. How to obey the Subset Principle: Case assignment. In *CUNYForum 17*, 55–84. Ph.D. Program in Linguistics, Graduate Center, CUNY, New York.
- Fodor, Janet Dean. 1994. How to obey the Subset Principle: Binding and locality. In Lust, Hermon, and Kornfilt 1994, 429–451.
- Fodor, Janet Dean. 1995. Fewer but better triggers. In *CUNYForum 19*, 39–64. Ph.D. Program in Linguistics, Graduate Center, CUNY, New York.
- Fodor, Janet Dean. In press a. Learnability theory: Decoding trigger sentences. In *Linguistics, cognitive science, and childhood language disorders*, ed. Richard C. Schwartz. Hillsdale, N.J.: Lawrence Erlbaum.
- Fodor, Janet Dean. In press b. Learnability theory: Triggers for parsing with. In *The development of second language grammars: A generative approach*, ed. Elaine C. Klein and Gita Martohardjono. Amsterdam: John Benjamins.
- Gibson, Edward. 1991. A computational theory of human linguistic processing: Memory limitations and processing breakdown. Doctoral dissertation, Carnegie Mellon University, Pittsburgh, Pa.
- Gibson, Edward, and Kenneth Wexler. 1994. Triggers. *Linguistic Inquiry* 25:407–454.
- Gorrell, Paul G. 1989. Establishing the loci of serial and parallel effects in syntactic processing. *Journal of Psycholinguistic Research* 18:61–73.
- Grodzinsky, Yosef. 1989. The language learner: A trigger-happy kid? *Behavioral and Brain Sciences* 12: 342–343.
- Holmberg, Anders, and Christer Platzack. 1991. On the role of inflection in Scandinavian syntax. In *Issues in Germanic syntax*, ed. Werner Abraham, Wim Kosmeijer, and Eric Reuland, 93–118. Berlin: Mouton.
- Inoue, Atsu, and Janet Dean Fodor. 1995. Information-paced parsing of Japanese. In *Japanese sentence processing*, ed. Reiko Mazuka and Noriko Nagai, 9–63. Hillsdale, N.J.: Lawrence Erlbaum.
- Kapur, Shyam. 1994. Some applications of formal learning theory results to natural language acquisition. In Lust, Hermon, and Kornfilt 1994, 491–508.
- Kayne, Richard S. 1989. Null subjects and clitic climbing. In *The null subject parameter*, ed. Osvaldo Jaeggli and Ken Safir, 239–261. Dordrecht: Kluwer.
- Lust, Barbara, Gabriella Hermon, and Jaklin Kornfilt, eds. 1994. *Syntactic theory and first language acquisition: Cross-linguistic perspectives*. Vol. 2, *Binding, dependencies, and learnability*. Hillsdale, N.J.: Lawrence Erlbaum.

- Marcus, Mitchell P. 1977. *A theory of syntactic recognition for natural language*. Cambridge, Mass.: MIT Press.
- Morgan, James L., Richard P. Meier, and Elissa L. Newport. 1987. Structural packaging in the input to language learning: Contributions of prosodic and morphological marking of phrases in the acquisition of language. *Cognitive Psychology* 19:498–550.
- Valian, Virginia. 1990a. Logical and psychological constraints on the acquisition of syntax. In *Language processing and language acquisition*, ed. Lyn Frazier and Jill de Villiers, 119–145. Dordrecht: Kluwer.
- Valian, Virginia. 1990b. Null subjects: A problem for parameter-setting models of language acquisition. *Cognition* 35:105–122.
- Valian, Virginia. 1993. Parser failure and grammar change. *Cognition* 46:195–202.

Ph.D. Program in Linguistics
Graduate Center
CUNY
33 West 42nd Street
New York, New York 10036
jfodor@email.gc.cuny.edu

Copyright of Linguistic Inquiry is the property of MIT Press and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.