Decoding Syntactic Parameters: The Superparser as Oracle

Janet Dean Fodor (jfodor@gc.cuny.edu)
Ph.D. Program in Linguistics; CUNY Graduate Center
365 Fifth Avenue, New York, NY 10016 USA

Virginia Teller (teller@cs.hunter.cuny.edu)
Department of Computer Science; Hunter College CUNY
695 Park Avenue, New York, NY 10021 USA

Abstract

Syntactic parameter setting has proven extremely difficult to model. The original 'switch-setting' metaphor failed because parametrically relevant properties of a natural language sentence cannot be recognized without considerable structural analysis. The result has been a move to trial-and-error learners which attempt to guess a grammar that can analyze (parse) the current input sentence. But standard variants of grammar guessing are wasteful of the parametric information in input sentences because they use it only as feedback after a candidate grammar has been chosen. We show here that performance is significantly improved by a 'superparsing' routine which constructs a candidate grammar on-line in response to the properties of the input sentence. No sentences then need to be discarded for lack of a grammar to parse them. The gain in learning speed can be quantified in terms of the average number of sentences required for convergence. Superparsing can be achieved by the normal sentence parsing routines, applying a grammar that incorporates all possible parameter values. The superparsing learner is robust and imposes no special demands on the input.

Natural Language Acquisition

Children exposed to a sample of sentences from a natural language acquire its grammar in a few years. There is as yet no computational model of the acquisition process that is both effective and psychologically realistic. The conception of the learner's task was greatly simplified with the advent of parameter theory (Chomsky, 1981, 1995) under which a natural language grammar consists of an innate component (Universal Grammar, UG) and a selection from among a finite set of properties by which languages can differ (the parameters). Depending on the particular linguistic theory assumed, a parameter might be a choice between grammar rules, or between the presence or absence of a rule in the grammar. But as developed in the Chomskyan framework a parameter specifies a more abstract property of grammatical derivations, such as the direction of case assignment by a verb, or the derivational level at which a universal constraint applies, or, in more recent versions, the 'strength' of a syntactic feature on a tree node. The learner's task is to select the correct setting for each relevant parameter, i.e., each parameter whose value contributes to the derivation of at least one sentence of the target language. Though the concept is simple, the task is more challenging than was originally appreciated.

The earliest idea was that some property of an input sentence would trigger the correct setting of each parametric switch. But the needed property detectors could not be devised, because the characteristic properties of sentences derived by means of some parameter value *v* (inter alia) are not sufficiently uniform and superficially identifiable (Clark, 1994; Gibson & Wexler, 1994). To find the abstract properties that identify *v*, the derivation of the sentence must be computed, i.e., the sentence must be parsed. But parsing is work, and the computational workload of a learner must be kept within psychologically plausible limits. The problem is that when the parser lacks the correct grammar, as it does by definition in a learning event, it must apparently try out multiple grammars until it finds a successful one. Even though the number of parameters is limited, there are too many possible grammars for it to be feasible for a learner to try them all on a single sentence, either serially or in parallel.

This is where current learning models diverge. Some test one grammar per sentence (e.g. Gibson & Wexler, 1994, henceforth GW94) and are very slow to converge on the correct grammar. Others test batches of grammars at a time (e.g. Clark, 1992; Nyberg, 1992) and thereby go beyond what it is plausible to suppose a child is capable of. In this paper we discuss a novel way to use the parser to decode the parameter values that license an input sentence, without undue use of resources (Fodor, 1998a). We provide here a quantitative assessment of the substantial increase in learning speed that this model permits compared with traditional grammar guessing models.

A Simple Model of Grammar Selection

A learning event begins with the learner receiving a novel input sentence *s*, a sentence of the target language not licensed by the learner's current grammar G_c. G_c may have some parameters set to correct values, but it also has one or more set to incorrect values or not set at all. The learner first attempts to parse *s* with G_c. If the parse were successful, no change would be made to G_c; this is error-driven learning (Gold, 1967). Since by hypothesis G_c does not license *s*, this parse attempt fails, and the learning device seeks an alternative grammar. To preserve psychological plausibility we will make the strong assumption here that only one more parsing attempt may be made on this same input. Hence, the task of finding a new grammar that does license *s* must be achieved by means of just one parse. If it is not, *s* must be discarded and learning must await further input. Each such discard increases the total number of inputs needed before the target grammar is attained, and hence decreases the speed of learning and increases the effort expended. It is important, therefore, for the learner to make good use of the one parse test of a new grammar that it is able to conduct.

However, standard grammar guessing procedures extract only minimal information from their interrogation of the input. Because of the single-parse restriction, just one

grammar must be selected to undergo the parse test, and this selection must of necessity be made *prior* to parsing. The success or failure of this parse attempt with the hypothesized grammar, G_h, provides feedback on the basis of which the learner decides whether or not to adopt Gh.[1] If the parse is successful the learner will adopt G_h; if the parse fails the learner is assumed to retain G_c. This policy of shifting to a new grammar only if it licenses the current input is the **Greediness** constraint of GW 94 and others. If G_h happens to be the target grammar, it will be retained permanently and learning is complete. A grammar is correct for the target language if it has the target value for each parameter relevant to the language. If some parameters are irrelevant to the target language there will be an equivalence class of correct parameters. For convenience in what follows we will refer to these grammars collectively as 'the target grammar', G_t.

Assessing Grammar Selection Efficiency

A simple random choice learning model such as this will demonstrably converge on the target grammar (Berwick & Niyogi, 1996). However, learning is slow largely because the learning component bases its actions on the mere fact that some randomly chosen G_h does, or does not, license *s*. We will show that learning could be substantially faster if instead the parser could reliably identify for the learner a grammar that licenses *s*. For the present let us suppose that this information is provided to the learner by an oracle. Later we will show how this oracle can be implemented.

Given $G_c \neq G_t$, what is the probability that the learner will shift to G_t as a result of an encounter with an arbitrary input sentence *s*? At the point where G_c is rejected, the random choice learner (without oracle) picks an alternative from among the set of all possible grammars, of which there are 2^n for *n* binary parameters. (For simplicity we treat G_c as a candidate grammar even though it has just failed.) Of these, 2^i are correct (are in the equivalence class G_t) where i = the number of parameters irrelevant to the target language. Thus the probability that the learner's selected G_h is $G_t = 2^i/2^n$. Observe that this is not sensitive if *s* uniquely determines every parameter value in the target grammar, the learner has no more chance of guessing correctly than if *s* is fully ambiguous. This is because, as noted, this learner must make its selection *before* testing out the selected grammar on s, and so it cannot restrict its guesses to grammars which license *s*.

Imagine now that this learning device is equipped with an oracle which offers the learner a grammar that licenses *s* (any one of the grammars in the domain that do so). Then the learner could take this grammar to be G_h, and avoid wasting attention on any grammar that does not license *s*. Let us say that a learning device which considers only grammars that license the current input meets the **Licensing** condition.[2] For a learner that satisfies Licensing, the chance of hypothesizing G_t would be 1/A, where A is the degree of ambiguity of *s*, measured as the number of grammars in the domain that license *s* divided by 2^i (the irrelevance factor). Clearly this is responsive to how informative the language sample is. For extremely ambiguous input (A approaching 2^{n-i}), the success rate is hardly better than without the oracle. But if *s* is unambiguous with respect to even one relevant parameter, the probability of a successful guess is increased.

This shows up in the speed of learning, estimated in terms of the number of inputs required, on average, to arrive at G_t. This is the reciprocal of the probability of a successful guess.[3] For the oracle learner this is A; for the random guess learner, it is 2^{n-i}. Table 1 shows the differences in average number of inputs consumed for various values of A and numbers of relevant parameters (= 2^{n-i}-A). On average, performance is improved by a factor of $(2^{n-i}\text{-A})/2^{n-i}$. The oracle learner, unlike the random choice learner, benefits to the extent that the input constrains the set of candidate grammars.

Table 1
Reduction due to oracle in average inputs to convergence

	Number of parameters relevant to G_t (=*n-i*)				
Avg. A	10	15	20	25	30
1	1023	32,767	1,048,575	33,554,431	1,073,741,823
10	1014	32,758	1,048,566	33,554,332	1,073,741,822
100	924	32,668	1,048,476	33,554,422	1,073,741,724
1000	24	31,768	1,047,576	33,553,432	1,073,740,824
1 million	-	-	48,576	32,554,432	1,072,741,824
1 billion	-	-	-	-	73,741,824

The values of A range from 1 to 2^a, where *a* is the number of parameters relevant to G_t whose values *s* does not determine. For a simple example: Assume that 30 parameters are relevant to G_t and $a = 25$. Such a sentence might be licensed by exactly 2 grammars, with opposite values for each of those 25 parameters. Or it might be licensed by grammars with all possible combinations of values for those parameters, of which there are $2^{25} = 33,554,432$. The former situation we will term **sparse ambiguity**, and the latter **dense ambiguity**; clearly, all situations in between are possible also.

It seems likely that the parametric ambiguity of natural

[1] Licensing is related to Greediness but the difference between them is important. Licensing applies in the selection of G_h, while Greediness governs only the grammar adoption stage at the end of each learning event. A learner that respects Licensing can also respect Greediness. The simple learning model discussed above shows that it is possible to obey Greediness but not Licensing.

[2] Licensing is related to Greediness but the difference between them is important. Licensing applies in the selection of G_h, while Greediness governs only the grammar adoption stage at the end of each learning event. A learner that respects Licensing can also respect Greediness. The simple learning model discussed above shows that it is possible to obey Greediness but not Licensing.

[3] Homogeneity is assumed in these calculations; no grammar is antecedently more likely than any other to license *s* or to be G_t.

languages is quite sparse (A much less than 2^a). In the miniature natural language domain defined by 3 parameters presented in GW94, ambiguity is less than fully dense in every one of the 11 sentence types in which two or more parameters are ambiguously expressed. It remains to be seen how this scales up in a domain of more realistic size. But the principle is clear. With maximally sparse ambiguity, a sentence could be ambiguous with respect to every parameter and nevertheless offer the oracle learner a 50% chance of guessing the target grammar. In general, for a constant degree of parametric ambiguity in terms of *a*, sparse ambiguity is more informative for a learning system capable of making use of it, i.e., a learning system that has knowledge of which grammars do and do not license the current input.

In a simple random choice system this information is unobtainable. It could be established only by testing every possible grammar on *s*, which clearly violates the limit of one parse per sentence (plus the original parse with G_c). Of course, this one-parse limit is just one instantiation of a practical ban on excessive processing, and the limit might be raised to two or three parses per sentence. But this will make little difference. In order to significantly reduce the amount of input needed for convergence, it would be necessary to permit testing of each sentence with as many grammars as required to find one that licenses it.[4]

However, there are other ways of improving the quality of grammar selection which do not presuppose an ability to sort grammars into those that do and do not license *s*. We review these in the next section. Their effects are less easy to quantify, but it is highly doubtful that either singly or jointly they could substitute for the usefulness of a Licensing oracle.

Criteria for Grammar Selection

Given a particular input sentence *s* from the target language, which grammar in the domain is it optimal for a learner to hypothesize? The grounds for selecting a grammar may be of several kinds, differing with respect to how much information they draw from the current learning situation. The preference for one grammar over another may be (a) independent of the current situation, or it may (b) reference the current grammar, and/or it may (c) reference properties of the current input. Some criteria of this latter kind may (d) require parsing of *s* with more than one new grammar, and thus exceed the limit on feasible processing for a learner without an oracle, which must select candidate grammars before knowing how they relate to the input.

(a) Orderings on the Class of Grammars

Grammar orderings may be imposed by linguistic principles of markedness. One value of a parameter may be less marked (more favored) than its other value; e.g., local binding of anaphors may be less marked than long-distance anaphors (Manzini & Wexler, 1987). Or parameters may be ordered with respect to each other: the marked value of one parameter may be less marked than the marked value of another parameter (Clark, 1989). Linguists have mostly been cautious about embracing markedness theory, but many markedness-type rankings are nevertheless implicitly assumed in linguistic descriptions (Wacholder, 1995). Also under type (a): grammars could be prioritized by linguistic maturation if, as has been proposed, some aspects of UG develop later than others (Wexler, 1999).

Criteria of type (a) may be helpful in resolving parametric ambiguities. To the extent that linguistic markedness has an impact on the frequency of grammar adoption by language communities (though this is a fraught topic), type (a) criteria can reflect the antecedent likelihood that any given grammar is the target. They may also reduce effort by holding learners to simpler or linguistically more natural grammars as long as the evidence permits.

(b) Rankings Relating to G_c

One G_c-related criterion is the Single Value Constraint (SVC) of GW94, which requires the learner to select grammars that differ from Gc in the value of just one parameter. Another is the assumption of 'indelible' or 'deterministic' learning, which requires that G_h include G_c. For parameter theory thi is taken to mean that once a parameter has been set, it may never be switched to its other value (Clahsen, 1990/91).

Type (b) criteria reflect what has already been gained by experience of the target language, insofar as this is compressed into the grammar G_c that the learner has been led to so far. (A learner is standardly assumed to have no memory of past inputs or past grammar hypotheses, other than their legacy in determining the current grammar.) Because of Greediness, a grammar that has been adopted by the learner may be assumed to be more likely to have some parameters correctly set than an arbitrary grammar in the domain; and a grammar similar to such a grammar may be presumed to share its virtues. The worth of these considerations has been disputed, but we need not enter the debate here; see Berwick & Niyogi (1996) and Sakas & Fodor (in press) for discussion. Clearly it is desirable for a learning device to have some way to hold onto past gains. To adopt a completely fresh hypothesis at each step, as permitted in an unconstrained guessing model, does nothing to improve the probability of success as learning proceeds.

(c) Rankings Based on Properties of *s* Identifiable by Parsing *s* with at Most One New Grammar

Type (c) criteria are sensitive to the current input but compatible with the ban on excessive processing even for a learner that first selects a grammar and then tests it. Input-sensitive criteria can deliver hard information. They constitute the learner's contact with the facts of the language and so should be a particularly helpful guide to the correct grammar.

Greediness and error-driven learning fall under type (c) as well as (b), since they refer to *s* as well as G_c. Greediness ranks all grammars that do not license *s* lower than G_c. The requirement of error-driven learning ranks G_c above all other grammars if G_c licenses *s*. These two input-sensitive criteria can be incorporated into a simple grammar guessing procedure without a Licensing oracle, because each can be checked with limited resources: a parsing attempt with G_c

[4] Parse-testing two grammars on *s* would double the chance of guessing G_t. This would as helpful as if *s* were unambiguous with respect to one parameter. However, testing *m* grammars on each sentence would increase the chance of success only linearly in *m*, not exponentially.

for error-driven learning, and then with one new grammar for Greediness.

By contrast, some input-sensitive selection criteria do not qualify as type (c) conditions because they require (or may do so) the checking of two or more new candidate grammars. This threatens to violate the ban on excessive processing for a learner without oracle. Licensing (as opposed to Greediness) is one casualty already noted: it imposes the tough requirement that a grammar must be known to be capable of parsing *s* in order to be selected for parsing *s*. Also not possible under type (c) is comparison of the derivations assigned to *s* by different candidate grammars, as would be necessary for application of a structural simplicity metric.

Even the grammar-similarity constraints of type (b) are affected by the limitation on processing. The SVC has been demonstrated by GW94 to be too stringent in that, in conjunction with Greediness, it can trap the learner at a local maximum where there is no grammar that both licenses an input sentence and differs from G_c in only one parameter value. If a range of alternative grammars could be evaluated, a more general **Closeness** criterion could be applied instead. The learner would adopt the grammar most similar to G_c among those that license *s* (with dead heats resolved by random choice or other criteria). The adopted grammar would differ from G_c by only one parameter value in many cases, but could differ by two or more if necessary. This would maintain the fruits of past learning while eliminating all local maxima. (Note that Closeness can be seen as a generalization of error-driven learning: G_c is to be changed only to the extent that is necessary in order to license the input.) But for this we must move up to type (d) criteria, which are not feasible for a standard grammar guessing learner.

(d) Rankings Based on Properties of *s* Identifiable Only by Evaluation of Multiple New Grammars

Closeness is an ideal similarity metric but (unlike the less flexible SVC) it is a comparative criterion which demands knowledge of all the grammars that license *s*, so that the one most similar to G_c can be selected. This puts it beyond the scope of any resource-limited pre-parse grammar selection process. Also falling under type (d) would be a simplicity measure which favors grammars that assign the smallest syntactic tree, or the shortest transformational derivation, compatible with the word string. This selection criterion seems very plausible both linguistically and psychologically, but is not easy to impose. How could a resource-limited learner set about discovering which of a million or a billion grammars assigns the simplest structure to *s*?

In general: Adding suitable grammar selection principles to a random choice learner can improve performance, compensating in part for inefficiency due to inability to discriminate between grammars that do and do not license *s* before committing resources to those that do not. However, the present analysis of grammar selection strategies makes clear that the most potent selection principles are also beyond the reach of such a system, and for much the same reason. We next show that both weaknesses can be remedied by the same means. With one change in how the parse test is conducted, the guessing learner can gain both a Licensing oracle which eliminates useless grammar guesses, and also the powerful type (d) criteria which improve the quality of guesses in case more than one grammar meets the Licensing condition, i.e., in case of parametric ambiguity.

Superparsing: A Constructive Process of G_h Selection

Inefficiency results from formulating a grammar hypothesis in advance of parsing the input string. This was assumed to be unavoidable, given the patent unfeasibility of first analyzing the string with all grammars as a basis for selecting one from among them. But if an optimal grammar choice cannot be made *before* parsing *s*, or *after* parsing *s*, perhaps it can be made *in the course* of parsing *s*. The solution we will outline is to let the ongoing parse shape the formulation of G_h. Total parametric decoding cannot be achieved by this means, for reasons we will explain, but most of the desirable learning characteristics we have been seeking do follow. By the end of the parse, the learner will know of one grammar that licenses *s*. Hence there will be no wastage of input due to lack of a grammar to parse it. The grammar the parser finds will always be drawn from among the A grammars that license *s*, rather than from the total set of 2^n grammars (or 2^{n-i} relevant grammars), so the learner will be taking full advantage of parametric disambiguation provided by the input. Where disambiguation is not total, Closeness and a structural simplicity metric can be applied to choose a good candidate, as indicated below.

Selecting a grammar that licenses *s*, during the course of parsing *s*, is feasible. Fodor (1998a) suggested the following procedure. The parsing routines set about parsing *s* with the current grammar G_c. If *s* is not licensed by G_c this parse attempt will break down at some place in the sentence. When it does, the parser should not stop and merely report back its failure, as we assumed earlier. Instead, it should supplement G_c with all possible parameter values and continue processing *s* with this 'supergrammar' **SG**. **SG** must afford at least one parse for *s* (as long as the sentence contains no unknown lexical items, and does not cause a severe 'garden path' beyond the capacity of the parser to recover from; see Fodor, 1998b). Where there is a choice of analysis for *s*, priority is given to the parameter values in G_c; this incorporates the error-driven learning condition. But new parameter values can be made use of as needed. Any new parameter value that is found to be necessary for parsing *s* is adopted by the learner. Thus, the superparser shuttles through the sentence flipping parameter settings as it goes, in response to the demands of the input sentence. Its output consists of (i) a complete parse tree, and (ii) a grammar that satisfies Licensing.

If *s* is fully unambiguous with respect to all the parameter values it expresses, the superparser has no choices to make (above the usual within-grammar ambiguity resolution choices of normal sentence processing). If *s* is ambiguous with respect to *a* parameters, **SG** assigns it up to 2^a distinct parse trees. In principle the parser might identify them all. In practice it could not, since this would require massive parallel parsing which would violate the general ban against excessive processing (even though it doesn't strictly violate the one-parse-per-sentence constraint imposed above). More reasonable is to suppose that the parser employed by the

learner for superparsing is the same parser that will be used throughout life for sentence comprehension. A standard assumption is that this is a serial device which, when it hits a point of ambiguity, selects one structural analysis to pursue for the sentence.[5] (Parallel parsing models have been proposed, but to conserve resources their parallelism is strictly limited, and their consequences for superparsing do not differ significantly from those of serial parsing.) Thus the superparser may be faced with choices to make between alternative ways of resetting parameters to assign an analysis to *s*. It can output only one of the A grammars that would satisfy Licensing. The choice between them might be random, or other selection criteria must be invoked.

Markedness and conservatism criteria (types (a) and (b)) could be employed, as well as input-sensitive type (c) criteria. The more powerful type (d) criteria such as Closeness and a minimal structure constraint are also available in this system. In fact, both of the latter are more or less automatic consequences of superparsing given that the human parser is a least-effort device (Inoue & Fodor, 1995). For instance, the Minimal Attachment parsing strategy entails that superparsing will prefer simple, compact trees over more complex ones; the learning device inherits this and so favor grammars that assign simpler structures. A conservative policy of staying close to the previous grammar will result if, as is natural, the parser makes the effort of changing parameter settings in G_c only when it is forced to do so to avoid parse failure. Again, parser preference translates into learner preference. In much the same way, frequency sensitivity in parsing could lead to frequency-sensitive learning (e.g. Charniak, 1993).

The exact mix of these various criteria remains to be established (e.g., Minimal Attachment versus minimal resetting of parameters). The supergrammar model allows various policies for resolving conflicts; which of these is adopted by human learners is an empirical question. To the extent that these criteria help the learner select an optimal grammar from among those that license *s*, the fact that they can be applied by a superparsing learner means that its efficiency gain compared with pre-parse selection criteria is even greater than was calculated above (Table 1). However, exact benefits are not easily quantified. The effects of Closeness and other such rankings are complex, and are best assessed by simulation studies. This awaits future research.

Limitations of Superparsing

Does superparsing as a means of parameter setting carry significant costs to offset these advantages, so that no net gain in efficiency results? This appears not to be so.

As noted, the parsing routines need not be unusually powerful. The mechanism can be the normal human sentence parsing device, which clearly must be present in children for comprehension of sentences already licensed by the current grammar. Thus, all that is special about the superparser is that it applies the supergrammar, augmented with all possible parameter values. This could exact a heavy cost in on-line processing due to the massive ambiguity of sentences in relation to the supergrammar, far greater in many cases than ambiguity levels relative to a settled adult grammar. However, the added cost of ambiguity is negligible as long as no attempt is made to compute all analyses of a sentence. For a serial parser, alternative parses are evaluated only momentarily as each new word is encountered and attached into the parse tree. They are not pursued through the sentence, and are not multiplicative. As soon as one of the alternatives has been chosen, the others can be forgotten. And arguably, even the selection process is cost-free in a least-effort system, since it consists of adopting the first (simplest) attachment option that is computed (Frazier & Fodor, 1978; Lewis, 1999). The only difficult analyses will be (i) those the human sentence parser has trouble with even when the grammar is settled, e.g., center-embedded constructions; and (ii) analyses which are not complex in themselves but are systematically masked by more attractive analyses allowed by the supergrammar. Grammar guessing models without superparsing would suffer from (i), but not from (ii). The incidence of such cases is not known. They could lead to false negative reports from the parser to the learner, indicating wrongly that *s* is not licensed by the grammar being tested. For examples and discussion see Fodor (1998b).

A requirement for smooth functioning of the superparsing routine is that the parameter values defined by UG are such that they can be added into a natural language grammar without altering its basic character. The competing values of one parameter must be able to co-exist in the same grammar without internal contradiction. And the parameter values temporarily added into G_c to create the supergrammar should be no harder to access and use on-line than other elements of natural language grammars. This may preclude any kind of precompiling process by which the combination of G_c and the added parameter values is reformulated for convenience in parsing, since the computational costs of repeated compiling would be added into the workload of the superparser. For some kinds of parameters (e.g., Subjacency applies at Surface Structure or at Logical Form; Huang, 1981/82) these conditions are hard to meet. But a variety of current linguistic theories conceive of parameter values as fragments of tree structure (see Fodor, 1998c), and these 'treelets' do meet the needs of superparsing. They can be directly added into a normal grammar to create another perfectly normal grammar, only slightly more complex than the original, and yet incorporating all the structural options that UG permits.

The one limitation of superparsing that is unavoidable is that it delivers only one structural analysis for each sentence. Because of the ban on excessive processing, it is impossible for the parser to present the learning component with all analyses for *s*, to compare and evaluate in order to make the best possible guess. The process of selecting one of the licensing grammars is piecemeal and order-dependent as each ambiguity must be resolved as it arises on-line. Interestingly, this appears to do relatively little damage, because there seems to be an excellent fit between the choices made on-line by the human parser and the choices that a well-designed learning device would be expected to favor: the minimization of derivational complexity, and the minimization of grammar

[5] In a serial parse, the selected resolution of an ambiguity may prove to be incorrect, by failing on subsequent words of the sentence. In a garden path situation such as this the superparser would engage in reanalysis procedures just as the human parser normally does in the case of a garden path.

revision. Whether this is merely a coincidence is not clear, but at any rate it is fortunate for superparsing as a method for parametric decoding.

Summary and Conclusions

A start has been made here on quantifying the efficiency advantage of a learning device which has the ability to read off a set of parameter values for licensing a sentence, in the course of parsing that sentence in the normal way for comprehension. The superparsing approach was developed originally for a different purpose. It was designed to provide a feasible ambiguity detection system, so that all parametrically ambiguous input could be discarded. This permitted development of a non-guessing learning routine, capable of error-free parameter setting based exclusively on unambiguous items in the input sample (Fodor, 1998a). Whether this is the best research goal, either for modelling human learning or for engineering applications, remains to be seen. The error-free learner has the advantage that it never has to re-set a parameter. Also, once it has set a parameter it can ignore the alternative value of that parameter thereafter, so the size of the domain to be searched for G_t shrinks as learning proceeds. It is an empirical issue whether these benefits balance the need to discard all ambiguous sentences in the input sample. It is therefore of interest, as we have shown here, that superparsing can also make a useful contribution to a grammar guessing routine.

To the extent that the input sample does carry parametric information, superparsing allows the learner to exploit it. Despite its modest consumption of resources, and despite its practical inability to list all parametric analyses of a sentence, the superparser nevertheless extracts from a sentence *all* the definitive parametric information it contains. If a sentence is compatible with only one grammar in the whole domain, the superparser will identify that grammar and the learning component will adopt it. If a sentence is less informative, e.g., is compatible with a thousand grammars, the superparser will identify one of the thousand. (Which one of the thousand it identifies depends on which ambiguity resolution criteria it applies on-line.) All parameter values expressed unambiguously by a sentence will be set correctly by the time it has been parsed. The values of the other parameters can only be guessed. They will be left unchanged from previous learning where possible; otherwise they will be changed to some combination of values which licenses *s*.

The superparsing learner is quite undemanding about the nature of its input. For example, it does not require a language to contain unambiguous triggers for all of its parameter values. A simple random grammar guessing learner also needs no unambiguous triggers, but that is because, as noted above, it gains hardly more from unambiguous than from ambiguous input. By contrast, a non-guessing error-free learner is very choosy; it needs an unambiguous trigger for each parameter, and moreover it needs some of these to be fully unambiguous (i.e., unambiguous with respect to all parameters they express). The superparsing learner has the dual virtues that it can use the information in fully unambiguous triggers when they are present, but it can also make progress when input sentences are ambiguous with respect to many (or even all) of the parameters they express. Thus it is robust as well as efficient.

Acknowledgements

This research was supported in part by PSC-CUNY Grants 61595-00-30 to Fodor and 61403-00-30 to Teller.

References

Berwick, R. C. and Niyogi, P. (1996) Learning from triggers. *Linguistic Inquiry* 27.4, 605-622.

Charniak, E. (1993) *Statistical Language Learning*, MIT Press, Cambridge, MA.

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

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

Clahsen, H. (1990/91) Constraints on parameter setting: A grammatical analysis of some acquisition stages. *Language Acquisition* 1.4, 361-391.

Clark, R. (1989) On the relationship between the input data and parameter setting. *NELS* 19, 48-62. GLSA, UMass.

Clark, R. (1992) The selection of syntactic knowledge. *Language Acquisition* 2.2, 83-149.

Clark, R. (1994) Finitude, boundedness and complexity. In B. Lust, G. Hermon and J. Kornfilt (eds.) *Syntactic Theory and First Language Acquisition: Cross-linguistic Perspectives.* Lawrence Erlbaum, Hillsdale, NJ.

Fodor, J.D. (1998a) Unambiguous triggers. *Linguistic Inquiry* 29.1, 1-36.

Fodor, J.D. (1998b) Parsing to Learn. *Journal of Psycholinguistic Research* 27.3, 339-374.

Fodor, J.D. (1998c) What is a parameter? Presidential Address to the Linguistic Society of America.

Frazier, L. and Fodor, J.D. (1978) The sausage machine: A new two-stage parsing model. *Cognition* 6, 291-325.

Gibson, E. and Wexler, K. (1994) Triggers. *Linguistic Inquiry* 25.3, 407-454.

Gold, E.M. (1967) Language identification in the limit. *Information and Control* 10, 447-474.

Huang, C.-T. J. (1981/82) Move *Wh* in a language without *Wh* movement. *The Linguistic Review* 1, 369-416.

Inoue, A. and Fodor, J.D. (1995) Information-paced parsing of Japanese. In R. Mazuka and N. Nagai (eds.) *Japanese Sentence Processing*, Lawrence Erlbaum, Hillsdale, NJ.

Lewis, R. (1999) Attachment without competition: A computational model of race-based parsing. 12th Annual CUNY Conference on Human Sentence Processing.

Manzini, R. and Wexler, K. (1987) Parameters, binding theory and learnability. *Linguistic Inquiry* 18.3, 413-444.

Nyberg, E. (1992) *A Non-deterministic Success-driven Model of Parameter Setting in Language Acquisition.* Unpublished Ph.D. Dissertation, Carnegie Mellon University.

Sakas, W. and Fodor, J.D. (in press) The structural triggers learner. To appear in S. Bertolo (ed.) *Parametric Linguistics and Learnability: A Self-contained Tutorial for Linguists*, Cambridge University Press, Cambridge, UK.

Wacholder, N. (1995) *Acquiring Syntactic Generalizations from Positive Evidence: An HPSG Model.* Unpublished Ph.D. Dissertation, City University of New York.

Wexler, K. (1999) Maturation and growth of grammar. In W.C. Ritchie & T.K. Bhatia (eds.) *Handbook of Language Aquisition*, Academic Press, San Diego.