Setting the First Few Syntactic Parameters - A Computational Analysis

William G. Sakas (sakas@roz.hunter.cuny.edu)
Ph.D. program in Computer Science; CUNY Graduate Center;
33 West 42 Street, New York, NY 10036 USA

Janet Dean Fodor (jfodor@email.gc.cuny.edu)
Ph.D. program in Linguistics; CUNY Graduate Center;
33 West 42 Street, New York, NY 10036 USA

Abstract

We consider the process by which the syntactic parameters of human language are set. Previous work has shown that for natural languages there can be no instant "automatic" triggering of parameters because the trigger properties in natural languages are often deep properties, not recognizable without parsing the input sentence. There are parametric algorithms that learn by parsing, but they are inefficient because they do not respect the *Parametric Principle*: they evaluate millions of grammars, rather than establishing the values of a few dozen parameters. They do so because they cannot tell in advance which input sentences are pertinent to which parameters, and because they have no protection against mislearning due to parametric ambiguity of the input. There is one model that does implement the Parametric Principle. This is the *Structural Triggers Learner* (STL). For an STL, a parameter value and its trigger are one and the same thing; they are what we call a *structural trigger* or *treelet* (a subtree or in the limiting case a single feature). These structural triggers are made available by UG and adopted into the learner's grammar just in case they prove essential for parsing input sentences. This permits efficient recognition of the parameter values entailed by input sentences and allows the learner to avoid errors by discarding ambiguous input. However, the high degree of ambiguity inherent in natural language impedes learning even for this efficient system. An STL must wait a long time between unambiguous inputs. As we explain, this problem is particularly acute in the early stages of learning. In this paper we give a computational analysis of the performance of an STL. We then identify an important factor – the *parametric expression rate* - that holds promise of a solution to this early learning problem.

Setting Syntactic Parameters is Hard Work

The Parametric Principle

Chomsky (1981 and elsewhere) has proposed that all natural languages share the same innate universal principles (Universal Grammar - UG) and differ only with respect to the settings of a finite number of parameters. For example, all languages have subjects of some sort, but whether a language's grammar dictates that the subject must be overt is determined by the setting of the "null subject" parameter. The null subject parameter is set "off" in English and "on" in Spanish and Italian.

These syntactic parameters are standardly taken to be binary and their two values to be mutually exclusive. It is not yet known how many syntactic parameters there are (see Roberts, in press, for discussion). For convenience here, suppose humans are equipped with 30. Then there are 2^{30} = 1,073,741,824 possible languages. Importantly, the number of languages grows exponentially with the number of parameters. The insight that underlies Chomsky's parametric approach to language is that for a learner that sets parameters, the complexity of the learning process need be no more than linear in the number of ways grammars can differ from each other (i.e. in the number of parameters). For this, the learner must establish the value of each parameter independently of the values of all others. We call this the *Parametric Principle*.

Setting parameters in accordance with the Parametric Principle permits a very rapid reduction of the pool of possible grammars. Each time a parameter is set, one parameter value is eliminated; and since half of all grammars have that value, that eliminates from consideration half of the candidate grammars remaining. In a domain of 30 parameters, setting one parameter rules out roughly 500 million grammars; setting the next one excludes another 250 million; setting five reduces the pool to roughly 3% of its original size.

Very few existing parameter-based learning models abide by the Parametric Principle. Rather, the learner evaluates complete collections of parameter values, i.e. whole grammars. Typically, no one parameter value is finally established until the learner discovers the full target grammar. Statistical weighting systems, such as those proposed by Valian (1990) and Kapur (1994), do settle on a value for each parameter independently, although along the way they postpone setting a parameter for some time while evaluating the evidence. It seems surprising that other models do not take advantage of the powerful reduction of the acquisition problem that the Parametric Principle makes possible. There are no compensating advantages to be gained by searching through the space of grammars. At best, clever search strategies may make it less punishing (see Nyberg, 1992). It appears that the sole reason for violating the Parametric Principle is that obeying it has been judged to be too difficult. The literature on language learnability has not emphasized this. Only Clark (1994) has addressed it explicitly, and he considered the

computational costs of respecting the Parametric Principle to be "too great to be acceptable." We believe this pessimistic conclusion to be premature. The learning model we present here obeys the Parametric Principle at minimal cost, and has other advantages besides.

Parametric Ambiguity

A sentence is *parametrically ambiguous* if it is licensed by two or more distinct combinations of parameter values. Parametric ambiguity is rampant in natural language. For example, an input string of the form Subject-Verb-Object (SVO) is parametrically ambiguous between underlying SVO order as in English, and "verb second" order as in German. Although SVO sentences can be parsed by either kind of grammar, the derivations will be different due to the different parameter settings.

A "verb second" language has a positive value of the V2 parameter (i.e. +V2). This entails that the finite verb is transformationally fronted and that a topic phrase is moved into first position before it. If the topic is the subject, this gives an SV(O) sentence. SVO order can be licensed by +V2 with any values for the parameters that control the underlying order of subject, verb and object (in German the underlying order is SOV). SVO order can also be licensed, without movement, by the parameter values for underlying subject-before-verb and verb-before-object order with a negative value of the V2 parameter, as in English. By contrast, a VOS sentence is not parametrically ambiguous (at least with respect to its word order). It can be licensed only by the -V2 value with underlying verb-before-object and verb-before-subject.

In order to abide by the Parametric Principle, a learner must be able to establish a parameter value with sufficient confidence to be prepared to rule out all grammars in which that parameter takes the opposite value. Otherwise, the number of candidate grammars could never be reduced. This would effectively nullify the benefit of the Parametric Principle: the halving of available grammars with each parameter that is set. Parametric ambiguity is potentially very damaging since it robs the learner of confidence. To be certain that its setting of a parameter was correct, the learner would need to run an exhaustive check of all possible parses of the input sentence. If the parameter value in question were present in every grammar that could parse the sentence, then it could be adopted with full confidence. Anything short of exhaustive grammar testing would leave it uncertain whether the parameter value was indeed necessary. Grammars can be applied to sentences by parsers; however, an exhaustive search through a billion grammars is hopelessly impractical. The consensus in sentence processing research is that even adults are capable of only limited parallel parsing if any (see Gibson, 1991), even when the alternative analyses all involve the same grammar. It does not seem plausible to suppose that a two-year old can apply a billion grammars to each sentence.

It might seem that a less demanding alternative would be for the learner simply to establish which inputs were parametrically ambiguous and refrain from setting parameters in response to them. Learning would be based solely on unambiguous triggers. However, this also demands parsing with multiple grammars. Parametric ambiguity can be established by parsing with just enough grammars to find two that parse the input, but **non**-ambiguity can be established only by parsing with all possible grammars and finding no more than one that parses the input. Fortunately, the STL model proposed by Fodor (1998) and described below manages to curtail the excessive parallel parsing needed to test multiple grammars.

In summary: the complexity of the learning problem remains exponential unless the Parametric Principle can be implemented. But, this must not require massively parallel parsing of input sentences, which appears to be far beyond the capacity of a human learner. What is needed is a system that can **test grammars** in parallel without having to engage in full **parallel parsing** of input sentences. This is possible within the STL model to which we now turn.

The Structural Triggers Learner

The Structural Triggers Learner (STL) has made parallel grammar testing possible without the need for full parallel parsing of input sentences. It achieves this because it takes triggers and parameter values to be the kinds of things that are both ingredients of grammars and ingredients of trees. What fits these specifications is a subtree consisting of just a few nodes and/or feature specifications. A trigger and the parameter value it triggers are then identical, so that only one innate specification is needed, rather than linked specifications of parameter values and their triggers. UG provides a pool of these schematic *treelets*, one for each parameter value, and each natural language chooses to employ some subset of them. The UG treelets can be folded into the learner's current grammar, when the current grammar alone is insufficient to parse an input. The resulting grammar (termed a "*supergrammar*") can be applied by the parser to the input in exactly the same way as any natural language grammar would be applied. No unusual parsing activity is needed, yet all parameter values are evaluated simultaneously. Parametric treelets necessary to that sentence can be detected in the output of the parser. The learning device can "see" which values contributed to the parsing of an input sentence, and thus know which values to adopt. Once a parameter value is adopted into the learner's grammar, it is available for licensing new sentences.

Fodor discusses various learning strategies which could utilize these structural triggers / parameter values. The one that is of interest here is what has been called the *weak STL* (WSTL), which is most appropriate as a model of human learners. The WSTL employs what is essentially a serial parser that parses sentences with the supergrammar (i.e. the current grammar augmented with parametric treelets). When the parser notes a choice point in assigning structure to a sentence, it selects one analysis to pursue for purposes of comprehension and it ignores all other analyses. But it reports the presence of ambiguity to the learning mechanism, and the learner thereafter adopts no new parameter values on the basis of that sentence. Since it cannot know what parameter values might have been

involved in the other parses, had it pursued them, it cannot be certain which values, if any, would be common to all analyses of the string, and so it cannot safely acquire any of them. However, if during the parse, **no** choice points were encountered, then the parser reports that the sentence is unambiguous and consequently the learner adopts all of the novel treelets that were used by the parser in constructing the parse tree.

The WSTL thus learns only from fully unambiguous sentences, so it does not make errors. The drawback is that, due to the high degree of parametric ambiguity in natural language, fully unambiguous input is likely to be scarce. So the WSTL, though efficient in other ways, faces the problem that it must discard a high proportion of sentences and must often wait a long time for usable (unambiguous) input. This problem is particularly acute at the earliest stages of parameter setting. In later stages, the input has become less ambiguous because each parameter that is set disambiguates some previously ambiguous sentences. But, before progressive disambiguation gets underway, it would seem that the learner could find very few unambiguous input sentences to learn from. We now formalize the WSTL model to illustrate the scale of this problem and in following sections we offer a solution.

Analysis of a WSTL

The WSTL-minus

The WSTL outlined above is the one we believe most closely models human learners. However, the complete mathematics that describes its performance is intricate (due to the dynamic nature of the progressive disambiguation; for details of this, see Sakas, in prep). Moreover, the full mathematics is not necessary for the present purpose of analyzing early parameter setting. We can consider here a simpler version of the WSTL that we will call the *WSTL-minus* (WSTL-). This does no learning unless a sentence is completely unambiguous regardless of whether any parameters have already been set, i.e. it behaves as if there were no progressive disambiguation. This can be implemented by assuming that the WSTL- has access, while parsing an input sentence, to both treelets associated with each parameter (whereas for the "true" WSTL, the parser's pool of parametric values does not include the rejected value of a parameter that has already been set). Due to the lack of progressive disambiguation, the WSTL- does not do justice to the efficiency of the full WSTL: it requires substantially more input sentences to acquire the target grammar than the WSTL does. Yet, at the outset of learning, when few parameters have been set, the performance of the two versions differs relatively little, and so the WSTL- is adequate as an approximation to the true WSTL.

WSTL- Algorithm:

1) Receive an input sentence ***s*** from the linguistic environment
2) Attempt to parse *s* with the current grammar; if successful, go to 1
3) Otherwise, begin to parse *s* with the supergrammar (= current grammar + both parametric treelets for every parameter)
4) If at some point in the parse, there is a choice of treelets, disregard *s* for learning
5) Otherwise, *s* is unambiguous, so adopt all novel parametric treelets that have been employed in parsing *s*
6) If not all parameters have been set, go to 1

The number of input sentences consumed by the WSTL- before convergence on the target grammar can be derived from the probability that the WSTL- will adopt some number of new parameter values, ***w***, on the basis of a single sentence. There are several factors (described in detail below) that determine this probability:

- the number of *relevant* parameters (***r***)
- the *expression rate* (***e***)
- the *"effective" expression rate* (***e'***)

Not all parameters are relevant parameters. Irrelevant parameters control properties of phenomena not present in the target language, such as clitic order in a language without clitics. For our purposes, the number of relevant parameters, ***r***, is the total number of parameters that need to be set in order to license all and only the sentences of the target language.

Of the parameters relevant to the target language as a whole, only some will be relevant to any given sentence. A sentence *expresses* those parameters for which a specific value is required in order to build a parse tree, i.e. those parameters that are essential to its structural description. For instance, if a sentence does not have a relative clause, it will not express parameters that concern only relative clauses; if it is a declarative sentence, it won't express the properties peculiar to questions; and so on. The expression rate, ***e***, for a language, is the average number of parameters expressed by its input sentences. For simplicity, we will assume here (not realistically) that e is the same for all target language sentences.

As a measure of ambiguity, consider that each sentence, on average, is ambiguous with respect to ***a*** of the parameters it expresses. The effective rate of expression, ***e'***, is the average number of expressed parameters that are expressed unambiguously (i.e. $e' = e - a$). (Note: Readers not interested in further mathematical details may skip the following sub-section.)

Derivation of a Transition Probability Function

We can now present a derivation of the number of inputs the WSTL- can be expected to consume before converging on the target grammar. We make the following background assumptions throughout this section.

- The sample of the target language that a learner is exposed to entails the value of every parameter relevant to the language.
- The sample is uniformly distributed (i.e. no particular sentence type within it is systematically withheld or delayed).
- All sentences within the target language are ambiguous with respect to the same number of parameters. (This is for mathematical convenience only.)

Our approach is to first derive the probability that the learner is exposed to a sentence containing one or more parameters (expressed ambiguously or unambiguously) that have not yet been set. We then derive the probability that the WSTL- does not discard that sentence because it contains an ambiguity. Combining these probabilities, as shown below, we will arrive at a formula that yields the probability that the learner will adopt w new parameter values ($0 \leq w \leq e$) on the basis of a given input sentence, given that t parameters had already been set.

To begin this computation, let us set ambiguity aside for a moment. In order to set all r parameters, the WSTL- has to encounter enough batches of e parameter values possibly (in fact, probably) overlapping with each other, to make up the full set of r parameter values that have to be established. Let $P(w|t,r,e)$ be the probability that an arbitrary input sentence, s, expresses w new (i.e. as yet unset) parameters, given that the learner has already set t parameters (correctly), for some r and e as defined above.

$P(w|t,r,e)$ is simply the number of ways s can express w unset parameters drawn from the current total pool of unset parameters (the size of which is $r - t$), times the number of ways s can express e-w previously set parameters from the total collection of t set parameters, divided by the total number of ways s can express e parameters out of all r relevant ones. This is displayed in the following equation:

$$P(w \mid t,r,e) = \frac{\binom{r-t}{w}\binom{t}{e-w}}{\binom{r}{e}}$$

Now, to deal with ambiguity, we bring the effective rate of expression, e', into play in order to calculate the probability that any single parameter is expressed unambiguously. This is e'/e. The probability that all e parameters expressed by a sentence are expressed unambiguously is $(e'/e)^e$. This is the probability, u, that an input is unambiguous and hence usable for learning.

$$u = \left(\frac{e'}{e}\right)^e$$

We are now in a position to give the formula for the probability, $P'(w|t,r,e,e')$, that the WSTL- will set w additional parameters after encountering an arbitrary input sentence.

$$P'(w \mid t,r,e,e') = \begin{cases} (1-u) + uP(w \mid t,r,e), & \text{if } w = 0 \\ uP(w \mid t,r,e), & \text{otherwise} \end{cases}$$

For values of w other than 0, the probability of setting w new parameters is simply the probability that the sentence is usable for learning (i.e., all e parameters are unambiguously expressed (= u)) times the probability that w of those e parameters were previously unset (= $P(w|t,r,e)$). The probability of setting 0 parameters (i.e., $w = 0$) is the probability that not all e parameters are unambiguously expressed (= 1- u) plus the probability that even if the e parameters are unambiguously expressed (= u) all of them had already been set (= $P(w=0|t,r,e)$).

In order to analyze the performance of the WSTL- after several inputs have been encountered, we model the learner as an absorbing Markov system where each state depicts the number of parameters that have been set[1] The system starts in state S_o and on receiving an input, may stay in state S_o or move to any of the states S_1, S_2, S_3, ... S_e. In general, the transition probability that the system will change from an arbitrary state S_t to state S_{t+w} is given by:

$$P(S_t \rightarrow S_{t+w}) = P'(w \mid t,r,e,e'), \quad 0 \leq w \leq e$$

Given values for r, e and e' one can calculate all possible transitions of the system and present them in a *transition matrix*. Since the WSTL- is error-driven (if the input is licensed by the current grammar no learning occurs), once it has set all relevant parameters (i.e. once it achieves state S_r), it stays in state S_r. Thus the system is *absorbing*. A well-known result from Markov chain theory is that the fundamental matrix of an absorbing Markov system yields the expected waiting time until absorption (see Waner and Costenoble, 1996, for a readable presentation of Absorbing Markov Models). The fundamental matrix Q is defined as the inverse of the difference between the identity matrix and the sub-matrix, N, that gives the transition probabilities between the **non**-absorbing states. That is: $Q = (I - N)'$. The sum of the first (S_0) row of Q yields the average number of inputs required for the WSTL- to enter the absorbing state S_r starting in state S_0.

[1] What follows is a presentation of one method for arriving at the expected size of the input sample consumed by the WSTL-. This approach is related to discussions in the literature by Niyogi and Berwick (1996) and elsewhere. There is at least one other approach that can be used for establishing these results. It utilizes dynamic programming to compute the following recurrence relation: that the expected sample size required, on average, to set n parameters can be determined from the size required to set n-i, $0<i<e$ parameters, together with the probability of setting i additional parameters given that n-i have been set.

Numeric Results

Table 1, below, displays numerical results derived for different values of r, e, and e'. e' enters indirectly via an ambiguity factor. Rather than using a as defined above, we employ a percentage measure of ambiguity a' in order to make comparable assessments of performance across different situations. We calculate a' as: $a' = 100\ (e\text{-}e')/e$.

Table 1: Average number of inputs consumed by the WSTL- before convergence

e	a'(%)	R=15	r=20	r=25	r=30
1	20	62	90	119	150
	40	83	120	159	200
	60	124	180	238	300
	80	249	360	477	599
5	20	27	40	55	69
	40	115	171	230	292
	60	871	1,296	1,747	2,218
	80	27,885	41,472	55,895	70,983
10	20	34	54	76	98
	40	604	964	1,342	1,738
	60	34,848	55,578	77,397	100,193
	80	35,683,968	56,912,149	79,254,943	102,597,823
15	20	28	91	135	181
	40	2,127	6,794	10,136	13,545
	60	931,323	2,975,115	4,438,464	5,931,148
	80	over 30 billion ...		.....	almost 200 billion
20	20	-	87	256	366
	40	-	27,351	80,601	115,415
	60	-	90,949,470	268,017,383	383,783,455
	80	-		... in the trillions ...	

Adjusting the mathematics to take into account progressive disambiguation as parameters are set (Sakas, in prep), we find that the input consumption rates in Table 1 are reduced by a factor of anything from approximately 1 to 20. This is useful, but less of a reduction than one might expect. The reason is that the benefit is strongest at later stages of learning, where fewer and fewer parameters in a sentence need to be expressed unambiguously; only **new** parameters need to be expressed unambiguously in order for the sentence to be usable for learning. Thus, once learning gets well underway, it is quite efficient. But at the early stages, the wait for unambiguous input is crippling.

Fortunately, this early ambiguity problem is not equally severe throughout the learning domain. Notice that the number of parameters to be set (r) has relatively little effect on convergence time. What dominates learning speed is the degree of ambiguity and the expression rate. When both a' and e are high, unambiguous inputs are very scarce. This is to be expected. For instance: there is little chance of encountering a fully unambiguous input if every sentence expresses 20 parameters and the ambiguity rate is 99% (the probability would be $(1/100)^{20}$). As a result, for high rates of expression and ambiguity there are very few sentences that the learner can make use of. This is why the worst cases in Table1 occur for high a' and e. For low ambiguity and/or low expression rate, however, learning proceeds very much more rapidly.

We emphasis that the results presented in this table portray the weakest WSTL which does not benefit from progressive disambiguation. This serves as a useful baseline for what follows, where we will consider how its performance can be improved.

How to Set the First Parameters

To avoid the inefficiency due to making and correcting errors, a WSTL waits for fully unambiguous input to learn from. We have shown that this can result in very slow rates of learning. Fortunately, this problem is not equally severe across the board. The generally damaging effect of ambiguity is absent at lower expression rates. We see that humble sentences that reveal only a few parameter values are the most useful for a learner seeking reliable information. This is important because expression rate is the one factor that might plausibly be low in real life learning. That there is a high degree of parametric ambiguity in natural languages seems undeniable. And, though linguistic research might prove otherwise, there seems little hope that the number of syntactic parameters relevant to a language will be reduced to less than a dozen. So there is not much prospect of a breakthrough in learning efficiency due to a reduction of either ambiguity rate or total number of parameters to be set. But it does seem within the realms of possibility that the expression rate for natural languages is as low as half a dozen parameters per sentence, particularly at the early stages of learning where the degree of parametric ambiguity is at its greatest. It is encouraging, therefore, to find that in the Structural Triggers framework, a reduction in the expression rate has a beneficial effect on learning speed.

Whether the earliest encountered sentences exhibit low expression rates must be determined by empirical research on child-directed language. But it seems reasonable to suppose that most of the sentences that early learners encounter do not exhibit every syntactic phenomenon in the language packed into 4 or 5 words or so. For instance, there are early child-directed sentences that contain negation, or overt WH-movement, or a subordinate clause, but probably few that involve them all.

To summarize: We have observed here that accurate learning is possible without loss of efficiency if the learner can take advantage of low expression rates at early stages of learning. What is essential to this solution to the early learning problem is the WSTL's ability to distinguish ambiguity of parameter expression from irrelevance of a parameter to a sentence. The former is to be avoided; the latter is very welcome as it divides the learning task into manageable steps. By contrast, whole-grammar testing systems treat parametric irrelevance as a species of parametric ambiguity. The parser reports success or failure for a grammar without distinguishing between the case where all its parameter values are either correct or

irrelevant, and the case where its parameter values give a wrong analysis of a sentence that is parametrically ambiguous. The WSTL does not conflate ambiguity and irrelevance because the conception of parameter values as treelets permits "superparsing", in which ambiguity shows up as a choice point, while irrelevant parameters never intrude at all.

References

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

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

Clark, R. (1994) Finiteness, Boundedness and Complexity: Learnability and the Study of First Language Acquisition, in Lust et al. (eds.).

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

Gibson, E. A. F. (1991) *A Computational Theory of Human Linguistic Processing: Memory Limitations and Processing Breakdown*, unpublished Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA.

Kapur, S. (1994) Some applications of formal learning theory results to natural language acquisition. In Lust et al. (eds.).

B. Lust, G. Hermon and J. Kornfilt (eds.) (1994) *Syntactic Theory and First Language Acquisition: Crosslinguistic Perspectives*, Vol. II, Lawrence Erlbaum Associates, Hillsdale, NJ.

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

Roberts, I. (in press) Language change and learnability. In S. Bertolo (Ed.), *Parametric Linguistics and Learnability: a Self Contained Tutorial for Linguists*, Cambridge University Press

Sakas, W.G. (in prep) *A Dynamic Analysis of the Structural Triggers Learner.* Unpublished ms., CUNY Graduate Center

Sakas, W.G. and Fodor, J.D. (in press) The Structural Triggers Learner. . In S. Bertolo (Ed.), *Parametric Linguistics and Learnability: a Self Contained Tutorial for Linguists*, Cambridge University Press

Valian, V. (1990) Logical and psychological constraints on the acquisition of syntax. In L. Frazier and J. de Villiers (eds.) *Language Processing and Language Acquisition* Kluwer, Dordrecht.

Waner, S. and Costenoble, S. R. (1996) *Finite Mathematics Applied to the Real World*, HarperCollins, New York.