Search, Structure or Statistics?
A Comparative Study of Memoryless Heuristics for Syntax Acquisition

William Gregory Sakas (sakas@hunter.cuny.edu)
Department of Computer Science, Hunter College, CUNY
Ph.D. Programs in Linguistics and Computer Science, The Graduate Center, CUNY
695 Park Avenue, New York, NY 10021 USA

Eiji Nishimoto (enishimoto@gc.cuny.edu)
Ph.D. Program in Linguistics, The Graduate Center, CUNY
365 Fifth Avenue, New York, NY 10016 USA

Abstract

Although several studies propose computational models of the process by which children acquire the syntax of their native language, most focus on a single algorithm applied in a single domain. Typically, the focus is *learnability* – under what conditions an algorithm can or cannot acquire the grammar of the target (native) language. Here, we present a comparative study of 12 algorithmic heuristics that are run in a domain that consists of 16 abstract languages each generated by a different grammar specified in Chomsky's principles and parameters framework. The heuristics consist of both those used in previously established models and new variations that we introduce. In contrast to a learnability study, our focus is *feasibility* – how much time and/or effort is required to achieve the target grammar. We find that the best heuristics make use of structural information obtained by parsing input sentences during the course of learning, that the performance of statistically-based heuristics are next in line, and finally, that heuristics that make use of hill-climbing search and a simple *can-parse/can't-parse* outcome from the parsing mechanism perform least well.

Background

Principles and Parameters

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 (binary) parameters. For example, all languages have a subject 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. The syntactic component of a grammar in the *principles and parameters* (*P&P*) framework is simply a collection of parameter values – one value per parameter. The set of human grammars is the set of all possible combinations of parameter values.

Language Acquisition The P&P framework was motivated to a large degree by psycholinguistic data demonstrating the extreme efficiency of human language acquisition. Children acquire the grammar of their native language at an early age – generally accepted to be in the neighborhood of five years old. In the P&P framework, if the linguistic theory delineates over a billion possible grammars by positing 30 parameters (2^{30}=1,073,741,824), a learner need only determine the correct 30 values that comprise the target grammar (henceforth, G_{targ}).

Given the generally accepted presupposition that a compelling theory of human language should show grammars to be easily acquirable, and since, at least on face value, parameters seem transparently learnable, it is not surprising that parameters have been incorporated into many current generative syntactic theories. However, the exact process of parameter setting has been studied only recently (cf. Bertolo, 2001 and references therein; Briscoe, 2000; Clark, 1992; Fodor, 1998a; Gibson & Wexler, 1994; Yang, 2000; among others), and although it has proved linguistically fruitful to construct parametric analyses, it turns out to be surprisingly difficult to construct a workable model of parametric syntax acquisition.

Parametric Ambiguity and the Need for Heuristics 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, a sequence of the form Subject-Verb-Object (SVO) is parametrically ambiguous between underlying SVO order as in English, and *verb second* (V2) order as in German.[1] Although SVO sentences can be parsed by either grammar, the derivations will be different due to the different parameter settings. By contrast, a VOS sentence is not parametrically ambiguous with respect to the V2 parameter. It can be licensed only by the -V2 value (since the second token is not a verb or auxiliary).

Ambiguity is a natural enemy of efficient language acquisition. The key problem is that, due to ambiguity, there does not exist a one-to-one correspondence between the linear left-to-right word order of an input sequence and the correct parameter values for the target grammar (as described above for an SVO sentence with respect to + or -V2). So, even if the learner hypothesizes parameter values which license the **single**, current input sentence, those values may ultimately be incorrect for G_{targ}. In the face of parametric ambiguity, efficient search heuristics must be employed to guide the learner towards the target grammar as sentences are progressively consumed by the learner. The remainder of the paper presents a comparative study of 12 search heuristics that are incorporated into current parameter-setting models of language acquisition.

Overview

A Measure of Feasibility

As a simple example of a learning heuristic and of our simulation approach, consider a domain of 4 parameters and a memoryless

[1] See Appendix for the linguistic details of how we implement the V2 parameter.

learner[2] which blindly guesses how all 4 parameters should be set upon encountering an input sentence. Since there are 4 parameters, there are 16 possible combinations of parameter settings (2^4=16), i.e., 16 different grammars. Assuming that each of the 16 grammars is equally likely to be guessed, the learner will consume, on average, 16 sentences before achieving G_{targ}. This is one measure of a model's efficiency or *feasibility*.

However, when modeling natural language acquisition, since practically all human learners attain the target grammar, the average number of expected inputs is a less informative statistic than the expected number of inputs required for, say, 99% of all simulation trials to succeed. For our blind-guess learner, this number is 72.[3] We will use this 99 percentile feasibility measure (*99% score*) for most discussion that follows, but also include the average number of inputs for completeness.

Error-Driven Learning

Although an outside oracle could ascertain when the blind-guess learner has acquired the target grammar, the learner itself has no "built-in" mechanism for identifying that it has achieved the target. Even if the correct grammar is hypothesized, the learner will most likely abandon it on the next sentence (with a probability of 15/16) and hypothesize a different (incorrect) grammar.

A standard way to build target identification into an algorithm is to dictate that the learner be *error-driven*.[4] Assuming that all inputs are grammatically correct instances of sentences that make up the target language, one could provide the learner with the ability to produce a *can-parse/can't-parse* outcome[5] given the current input sentence, the current hypothesized grammar (G_{curr}), and the rule: *Don't change G_{curr} unless there is parse failure*. With this error-driven constraint, there is no need for an outside oracle to stop the learner from relinquishing the target grammar once it is attained. Since all sentences are generated (by definition) by G_{targ}, parse success is guaranteed once $G_{targ} = G_{curr}$, and thus the learner will not be motivated to shift from its current hypothesis.

An *Error-Driven Blind-Guess* (*EDBG*) learner is our first heuristic of interest. It is easy to show that the average and 99% scores increase exponentially in the number of parameters. Clearly, human learners do not employ any strategy that performs as poorly as this.

Table 1: EDBG, # of sentences consumed

	99%	Average
EDBG	86	15.09

[2] By "memoryless" we mean that the learner processes inputs one at a time without keeping a history of encountered inputs or past learning events.

[3] The average and 99 percentile figures (16 and 72) in this section are easily derived from the fact that input consumption follows a hypergeometric distribution. See Chung (1979) for an overview.

[4] Error-driven grammar learning was first introduced by Gold (1967) and has become a standard in learnability research.

[5] We intend for a "*can-parse/can't-parse* outcome" to be equivalent to the result from a language membership test. If the current input sentence is one of the set of sentences generated by G_{curr}, *can-parse* is engendered; if not, *can't-parse*.

The Simulations

In all experiments:

- The learners are memoryless.
- The language *input sample* presented to the learner consists of only grammatical sentences generated by G_{targ}.
- The heuristics are tested in a 4-parameter, 16-language domain (see Appendix for details).
- For each heuristic, 1,000 trials were run for each start/target grammar pair.[6]
- At any point during the acquisition process, each sentence of G_{targ} is equally likely to be presented to the learner.[7]

Subset Avoidance and Other Local Maxima Depending on the algorithm and the learning domain, it may be the case that a learner will never be motivated to change G_{curr}, and hence be unable to ultimately achieve the target. This is often referred to as a *local maximum*. For example, the EDBG learner will be trapped if G_{curr} generates a language that is a superset of G_{targ}. There is a wealth of remarkable learnability literature that addresses local maxima and their ramifications.[8] However, since our study's focus is on feasibility (rather than on whether a domain is learnable) given a particular algorithm, we posit a built-in avoidance mechanism, such as the *subset principle* and/or *default values* that preclude local maxima; hence, we set aside trials where a local maximum ensues.

The Heuristics

The heuristics we present can be separated into two families based on the way they process input sentences: those that guide the learner towards the target by use of a can-parse/can't-parse outcome, and those that take advantage of information gleaned from the parse trees that are constructed by the parser. Gibson and Wexler's (1994) *Triggering Learning Algorithm* (*TLA*) and Yang's (2000) *Variational Model* make use of can-parse/can't-parse outcomes only. Fodor's (1998a) *Structural Triggers Learner* (*STL*) takes advantage of more extensive structural information obtained from sentence parsing.

TLA

The TLA incorporates two search heuristics: the *Single Value Constraint* (*SVC*) and *Greediness*. In the event that G_{curr} cannot parse the current input sentence *s*, the TLA attempts a second

[6] For the STL models presented later in the paper, the start grammar is unspecified.

[7] Not reported here are results from simulations run on several different distributions of input sentences, in particular, those where the shorter (presumably simpler) sentences occur more frequently. The relative performance of the heuristics is substantially the same; however, in all cases acquisition requires more inputs. These distributions and their relationship to the rate of ambiguity in the domain are currently being analyzed. Also, see Niyogi & Berwick (1996) for mathematical treatment of how the distribution of inputs affects TLA performance.

[8] Discussion of the problem of subset relationships among languages starts with Gold's (1967) seminal paper and is discussed in Berwick (1985) and Wexler & Manzini (1987). Detailed accounts of the types of local maxima that the learner might encounter in a domain substantially similar to the one we employ are given in Frank & Kapur (1996), Gibson & Wexler (1994), and Niyogi & Berwick (1996).

parse with a randomly chosen new grammar, G_{new}, that differs from G_{curr} by exactly one parameter value (SVC). If G_{new} can parse *s*, G_{new} becomes the new G_{curr}, otherwise G_{new} is rejected as a hypothesis (Greediness). Following Berwick and Niyogi (1996), we also ran simulations on two variants of the TLA – one with the Greediness heuristic but without the SVC (TLA minus SVC, *TLA–SVC*) and one with the SVC but without Greediness (TLA minus Greediness, *TLA–Greed*). The TLA has become a seminal model and has been extensively studied (cf. Bertolo, 2001 and references therein; Berwick & Niyogi, 1996; Frank & Kapur, 1996; Sakas, 2000; among others). We will not rehash these earlier discussions here. We include the TLA in our study to present comparisons against other models.[9]

Table 2: TLA variants, # of sentences consumed

	99%	Average
TLA-SVC	117	18.12
TLA-Greed	102	19.44
TLA	227	22.56

The Variational Learner

Like other models, Yang's *Variational Learner* (*VL*)[10] incorporates the notion of a current grammar hypothesis (G_{curr}) which is applied to the current input sentence. However, unlike the other learners, the VL maintains, for each parameter, a *weight* varying from 0 to 1. Roughly, the weight of a parameter can be construed as a measure of the past successes (or failures) of either a 0 or 1 value for that parameter during prior parses of input sentences. If the weight is closer to 0, then the 0 value has been more successful; if the weight is closer to 1, the 1 value has been more successful.[11] The VL uses the weights to guide the selection of the next hypothesis.

Since Yang's emphasis was on learnability in the limit and not on feasibility, in order to compare performance against other learners, we make a minor adaptation by adding a stopping criterion (see Step 4 below). The algorithm proceeds as follows:

1. Initialize weights for all parameters to 0.5.
2. Choose a new G_{curr} randomly, but biasing the choice towards parameter values favored by the current weights.
3. If G_{curr} succeeds in parsing the current input, nudge the weights in the direction of the values that make up G_{curr} (*reward* G_{curr}). This has the effect of making those parameter values that make up G_{curr} more likely to be chosen in the future. Otherwise nudge the weights away from G_{curr}'s parameter values (*punish* G_{curr}).
4. If all the weights are within a target threshold of either 0 or 1 then learning ends. Else go to Step 2.[12]

We chose 0.01 as the threshold value for stopping the learning process. Given weights w_1, w_2, ..., w_n where n represents the number of parameters, learning ends if all weights are in the range: $0.0 \leq w_i < 0.01$ or $0.99 < w_i \leq 1.0$ (i.e., every weight is very close to either a 0 value or a 1 value). Note that this criterion could be UG-endowed; that is, an oracle is not required to stop the algorithm.

In both Yang's original VL and the version presented above, the amount that the weights are "nudged" during learning is controlled by the *learning rate* (γ).[13] We ran simulations with γ=.1, γ=.5, and γ=.75.

Yang's model is of particular interest because it is an explicit implementation of the idea to keep statistics on the effectiveness of parameter settings based on the success or failure of past learning events. The reward/punish scheme of the VL is arguably an extension of error-driven learning in that incorrect grammars are punished, but it is significantly different from the standard constraint in which G_{curr} cannot change after a successful parse. The scheme works because the VL is statistical in nature; as parameter values vie for domination, the ones most successful in the past (and hence rewarded) are most likely to be chosen in the future and will eventually prevail.[14]

However, the statistical nature of the VL, together with the stopping criterion, may lead the learner to a local maximum. A parasitic input sample might lead to an abundance of evidence (i.e., successful parses) for incorrect parameter values early in the course of learning. If the learner encounters enough misleading evidence, it would cross the stopping threshold prematurely (on parameter values that are different from those that make up G_{targ}). This effect can be prevented by keeping the learning rate low, which gives the weights elbow room to fluctuate more gently until the evidence eventually supports the correct target. Unfortunately, the cost of keeping the learning rate low can be quite severe. In our 16-language domain, the VL requires well over 100,000 input sentences[15] to achieve a 99% score with a learning rate of 0.1. When the learning rate was increased to 0.5, the expected number of inputs dropped to 242, and when the rate was increased to 0.75, the number dropped even further to 84, albeit with many more local maxima (learning failures).

To combat the extraordinarily high number of input sentences needed, we incorporated into Yang's VL a variation of the

[9] In particular, the data in Table 2 reinforce Berwick & Niyogi's (1996) conclusion that in addition to creating local maxima, SVC and Greediness reduce learning speed.

[10] We use the proper name and acronym for readability; they are not used by Yang.

[11] Of course, if the values from one or more parameters are strongly tied to values of another parameter(s), the weight does not represent a simple ratio or percentage of success. Still, Yang's intention is that the weights are an informative measure of past performance.

[12] This stopping criterion could lead the VL to converge on the wrong grammar. We consider these cases as local maxima, and as such, trials in which local maxima ensue are discarded. Also implemented was a different stopping criterion which requires that all the weights fall within a threshold of the values that make up G_{targ}. Clearly, this strategy could not be UG-endowed because it requires that the learner have advance knowledge of these values. It is also less efficient than the criterion above, though it does enforce convergence on the correct grammar.

[13] The exact amount follows the Linear reward-penalty (L_{R-P}) scheme (Bush & Mosteller, 1958). See Yang (2000) for details.

[14] See Yang (2000) for a proof of convergence as the number of inputs approaches infinity.

[15] We found that many trials required over 100,000 input sentences which was an arbitrary stopping point built into our simulation.

standard error-driven constraint.[16] The *Error-Driven Variational Learner* (*EDVL*) proceeds as follows:

1. Initialize weights for all parameters to 0.5.
2. Randomly choose a new G_{curr} to start with.
3. If G_{curr} succeeds in parsing the current input, do nothing (no reward; no punish). Otherwise, randomly choose a new grammar (G_{new}) biased by the weights, re-parse with G_{new}, and apply Yang's original reward/punish scheme to adjust the weights.
4. Set G_{curr} to values **directly indicated by the weights** – that is, if $w_i > 0.5$, then the value of parameter *i* becomes 1, if $w_i < 0.5$, then the value of parameter *i* becomes 0, if $w_i = 0.5$, then either 0 or 1 is chosen at random.
5. Go to Step 3.

Note that no stopping criterion is needed, and that there are no local maxima because a move to another grammar is always possible before G_{targ} is attained. As is shown in the table below, the addition of the error-driven constraint greatly improves performance. Surprisingly, contra Yang's original model, performance **deteriorates** as the learning rate increases. We speculate that this is because a high learning rate encourages excessive exploration of different grammars, thus superseding the (positive) conservative nature of the error-driven constraint.

Table 3: VL & EDVL, # of sentences consumed

		99%	Average
VL	$\gamma = .1$	Over 100,000	Over 33,000
VL	$\gamma = .5$	242	46.35
VL	$\gamma = .75$	84	17.91
EDVL	$\gamma = .1$	44	8.39
EDVL	$\gamma = .5$	55	9.23
EDVL	$\gamma = .75$	75	12.16

STL

Fodor's *Structural Triggers Learner* (*STL*) makes greater use of the parser than the models discussed so far. A key feature of the model is that parameter values are not simply 0 or 1, but rather bits of tree structure or *treelets*. Thus, a grammar in the STL sense is a collection of treelets rather than a collection of 1's and 0's. The STL is error-driven. If G_{curr} cannot license *s*, new treelets will be utilized to achieve a successful parse.[17] Treelets are applied in the same way as any "normal" grammar rule, so no unusual parsing activity is necessary. The STL hypothesizes grammars by adding parameter value treelets when they contribute to a successful parse.

The basic algorithm for all STL variants is:

1. If G_{curr} can parse the current input sentence, retain the treelets that make up G_{curr}.
2. Otherwise, parse the sentence making use of any or all parametric treelets made available by UG, and adopt those treelets that contribute to a successful parse.

The STL stands apart from other acquisition models in that it can detect when an input sentence is parametrically ambiguous. During a parse of *s*, if more than one treelet could be used by the parser (i.e., a *choice point* is encountered), then *s* is (possibly) parametrically ambiguous. The TLA and the VL do not have this capacity because they rely only on a can-parse/can't-parse outcome and do not have access to the on-line operations of the parser. Originally, the ability to detect ambiguity was employed in two variations of the STL: the *strong STL* (*SSTL*) and the *weak STL*.

The SSTL executes a full parallel parse of each input sentence and adopts only those treelets (parameter values) that are present in all the generated parse trees. Note that even if the surface word order of the input is ambiguous between several languages (i.e., the sentence belongs to more than one language), the SSTL can identify unambiguous parameter values (treelets) by looking at **all** of the tree structures that the parser constructs for the sentence. This makes the SSTL an extremely powerful model, and for this reason, it establishes an upper standard against which to compare other models. It is not, however, proposed as a psychologically realistic model. As with the Error-Driven Blind-Guess (EDBG) learner, it is clear that human learners do not exhibit an SSTL-like strategy. The consensus in sentence processing research is that adults are only capable of limited parallel parsing, if any (cf. Gibson, 1991). It does not seem plausible to suppose that children possess a more powerful mechanism than adults.

On the other hand, the weak STL executes a psychologically plausible left-to-right serial (deterministic) parse. One variant of the weak STL, the *waiting STL* (*WSTL*), deals with ambiguous inputs abiding by the heuristic: *Don't learn from sentences that contain a choice point.* These sentences are simply discarded for the purposes of learning. This is not to imply that children do not parse ambiguous sentences they hear, but only that they set no parameters if the current evidence is ambiguous.

Table 4: SSTL & WSTL, # of sentences consumed

	99%	Average
SSTL	14	3.35
WSTL	30	5.11

As with the TLA, these STL variants have been studied from a mathematical perspective (Bertolo et al., 1997; Sakas, 2000; Sakas & Fodor, 2001a). Although the simulation results indicate notably better performance than the other models examined thus far in this paper, previous mathematical analyses lend doubts to the ultimate success of the WSTL model. The WSTL requires some **fully** unambiguous sentences for any learning to take place. It is probably the case that fully unambiguous triggers are few and far between in the domain of human languages, and negative WSTL performance is exponentially tied to the rate of ambiguity in the domain; that is, in a more realistically ambiguous domain than the one we explored so far, the WSTL may consume and discard an extremely large number of sentences before attaining G_{targ}. This result has spurred a new class of weak STL variants

[16] The resulting algorithm is similar to one originally proposed in Fodor (1998b).

[17] In addition to the treelets, UG principles are also available for parsing, as they are in the other models discussed above. See Appendix for details that apply to the domain we use here.

which we informally call the *guessing STL* family (Sakas & Fodor, 2000).

The basic idea behind the guessing STL models is that there is some information available even in sentences that are ambiguous, and a strategy could exploit that information. We incorporate four different heuristics into the original STL paradigm:

- *Strong Oracle* (*SO*) – perform a parallel parse of the current input *s* and choose a hypothesis grammar that licenses *s* and is most similar (in terms of hamming distance) to G_{curr}.
- *Random Choice* (*RC*) – parse serially; when a choice point is encountered, randomly pick a parsing alternative and adopt the treelets that are present in the final tree structure.
- *Minimal Chain* (*MC*) – parse serially; when a choice point is encountered, pick the choice that obeys the *Minimal Chain Principle* (De Vincenzi, 1991), i.e., avoid positing movement transformations if possible.
- *Local Attachment/Late Closure* (*LAC*) – parse serially; when a choice point is encountered, pick the choice that attaches the new word to the current constituent (Frazier, 1978).

Although the MC and LAC heuristics are not stochastic, we regard them as "guessing" heuristics because, unlike the WSTL, a learner cannot be certain that the parametric treelets obtained from a parse guided by MC and LAC are correct for the target. These heuristics are based on well-established parsing preferences that adults employ, so it seems likely that children apply them also (Fodor, 1998b).

The SO heuristic was originally conceived by Fodor and Teller (2000) as an extension of the SVC. Their main point was the efficiency advantage that results from using the parser to **find** a successful parse of the current sentence, so that can't-parse trials are eliminated. Given this capability, the question of how to choose among possible parses arises. The SO criterion presupposes full parallel parsing, which is unrealistic, but our approximation to it would result from letting the parser employ new treelets only where current ones do not suffice. Our data show that the conservatism of the SO heuristic pays off: it gives the strongest performance of any learner in our study, including the SSTL.

The RC, MC, and LAC heuristics show a significant improvement over the waiting strategy (WSTL). The difference between the three variants is slight.

Table 5: guessing STL family, # of sentences consumed

	99%	Average
SO	10	2.33
RC	26	3.43
MC	26	3.44
LAC	24	3.25

Conclusions

One can expect 86 sentences to be consumed by a population of (baseline) EDBG learners before 99% of the population acquires the target grammar in our small domain of 16 languages, and TLA variants require more input than that. The implemented heuristics in the two paradigms forego information, structural or statistical, in favor of a simple mechanism – the information needed for language acquisition must somehow be available from the surface word strings that make up the languages of the domain. They will be successful only if either i) there are recognizable, unambiguous signals in the surface strings that trigger correct parameter values or ii) the distribution of cross-language ambiguity[18] in the domain being studied is conducive to the heuristics being employed. There is faint evidence for both cases. For (i), in a domain without *null subject/topic*, the fact that a VOS sentence does not have a finite verb or auxiliary as the second token is indeed secure evidence for the -V2 parameter value. Although true for the -V2 value in this case, it is unclear how other plausible syntactic descriptions will offer the same advantage for the gamut of complicated linguistic phenomena (e.g., null subject/topic) with which human languages are inundated. As for (ii), previous work by the authors (Sakas, 2000) demonstrates that the TLA is a feasible learner in *strongly smooth* domains – domains in which there is a monotonic correlation between the similarity of grammars and the languages that are generated by them. Although still an open question, linguists have argued that natural languages are not strongly smooth.

The Error-Driven Variational Learner (EDVL) is a more promising model of language learning. On average, 44 sentences will allow 99% of the population to attain the target grammar. Its success can be attributed to the use of a strategy that maintains statistics of past performance without the unreasonable requirement that the learner memorize an entire input sample.

The most efficient heuristics, however, are those that make the most use of tree structure produced by the parsing mechanism: the psychologically plausible STL variants require almost half the number of inputs consumed by the EDVL.

Conjectures and Ongoing Research

The relative success of the EDVL is important for a reason not explicated earlier. Preliminary investigation points to the fact that, uniquely among the heuristics in our study, the EDVL performs more efficiently in ambiguous domains than in unambiguous ones at the outset of learning. This could turn out to be crucial as the guessing STL family might easily be foiled by larger, more syntactically complicated domains which generate sentences that contain a multitude of choice points – the result being that the parse tree computed for an input sentence, which guides parameter setting, reveals little about the target grammar. However, it has been shown that the STL performs extremely efficiently after just a few initial parameters have been set (Sakas, 2000). One can imagine a hybrid model of a guessing STL heuristic combined with a VL-like statistical heuristic where the statistical heuristic is used to bootstrap learning, and as performance deteriorates, the structural heuristic acquires more control of the acquisition process. This is being investigated in ongoing research.

The hard data that comprise our current study, however, reveal that the utilization of structural information outperforms the statistical heuristic overall.

[18] For present purposes, cross-language ambiguity is defined as some measure based on the intersection of the sets of surface forms that make up the languages in the domain. See Fodor & Sakas (2001b) for other definitions and the effects of ambiguity on learning efficiency.

Appendix: Simulation Domain

The four parameters are: *Specifier Initial/Final*, *Complement Initial/Final*, *V2 Movement*, and *Null Subject-Topic*. We largely adopt the first three parameters from Gibson and Wexler (1994). The Specifier and Complement position parameters are non-transformational. It is assumed that the subject is base generated in Spec of IP, and that the verb moves to I in the final structure (if I is not filled with an auxiliary).

Input sequences are formed with tokens representing adverb, subject, verb, auxiliary, direct object, and indirect object. Following Gibson and Wexler, we assume that the learner can directly determine the role of a noun phrase. For example, the noun phrase *the big dog* is interpreted as a subject or one of the object types based on its role when uttered.

The following constraints on the domain are in place: all sentence types are degree-0 (i.e., no subordinate clauses); Spec of CP is to the left of C'; C always precedes IP; all adverbs are sentential (i.e., base generated in Spec of CP); and there is no transformational reordering of constituents (e.g., topicalization, wh-movement, scrambling, etc.) with the exception of the V2 movement described below.

The V2 Movement [+/-V2] parameter determines whether a finite verb moves to the second position in the root clause. That is, a "verb second" language [+V2] entails that the finite verb is transformationally fronted to C (from I) and that a topical element is moved into Spec of CP (if Spec of CP is not filled with an adverb). In a [-V2] language, the verb moves only up to I, and there is no movement of a topic into Spec of CP.

Extending Gibson and Wexler's domain, we add the Null Subject-Topic [+/-Null] parameter. This parameter represents either subject drop or topic drop depending on the value of the V2 parameter. In [-V2] languages, an overt subject is either optionally [+Null] (as in Spanish and Japanese) or obligatorily [-Null] (as in English). For [+V2] languages, the parameter works similarly, but instead of a subject, an overt topic may be optionally [+Null] or obligatorily [-Null].

Acknowledgments

We would like to thank Bob Berwick, Janet Fodor, Virginia Teller, Charles Yang, and the learnability project (CoMoLA) group at CUNY for much useful discussion and suggestions. This research was supported by CUNY Collaborative Grant #92902 and PSC-CUNY Grant #63387.

References

Bertolo, S. (Ed.). (2001). *Language Acquisition and Learnability*. Cambridge, UK: Cambridge University Press.

Bertolo, S., Broihier, K., Gibson, E., & Wexler, K. (1997). Characterizing learnability conditions for cue-based learners in parametric language systems. *Proceedings of the Fifth Meeting on Mathematics of Language*.

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

Berwick, R. C., & Niyogi, P. (1996). Learning from triggers. *Linguistic Inquiry*, *27* (4), 605-622.

Briscoe, T. (2000). Grammatical acquisition: Inductive bias and coevolution of language and the language acquisition device. *Language*, *76* (2), 245-296.

Bush, R., & Mosteller, F. (1958). *Stochastic Models for Learning*. New York: Wiley.

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

Chung, K. L. (1979). *Elementary Probability Theory with Stochastic Processes*. New York: Springer-Verlag.

Clark, R. (1992). The selection of syntactic knowledge. *Language Acquisition*, *2* (2), 83-149.

De Vincenzi, M. (1991). *Syntactic Parsing Strategies in Italian*. Dordrecht: Kluwer.

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., & Teller, V. (2000). Decoding syntactic parameters: The superparser as oracle. *Proceedings of the Twenty-second Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Frank, R., & Kapur, S. (1996). On the use of triggers in parameter setting. *Linguistic Inquiry*, *27* (4), 623-660.

Frazier, L. (1978). *On Comprehending Sentences: Syntactic Parsing Strategies*. Doctoral dissertation, University of Connecticut.

Gibson, E. (1991). *A Computational Theory of Human Linguistic Processing: Memory Limitations and Processing Breakdown*. Doctoral dissertation, Carnegie Mellon University.

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

Gold, E. M. (1967). Language identification in the limit. *Information and Control*, *10* (5), 447-474.

Niyogi, P., & Berwick, R. C. (1996). A language learning model for finite parameter spaces. *Cognition*, *61*, 161-193.

Sakas, W. G. (2000). *Ambiguity and the Computational Feasibility of Syntax Acquisition*. Doctoral dissertation, City University of New York.

Sakas, W. G., & Fodor, J. D. (2000). Setting syntactic parameters: A computational analysis of child-directed speech. *CUNY Collaborative Incentive Research Grant*. City University of New York.

Sakas, W. G., & Fodor, J. D. (2001a). The Structural Triggers Learner. In S. Bertolo (Ed.), *Language Acquisition and Learnability*. Cambridge, UK: Cambridge University Press.

Sakas, W. G., & Fodor, J. D. (2001b). Learning-relevant properties of natural language domains. *The Seventh Annual Conference on Architectures and Mechanisms for Language Processing (AMLaP)*. Saarbrücken, Germany.

Wexler, K., & Manzini, R. (1987). Parameters and Learnability in Binding Theory. In T. Roeper & E. Williams (Eds.), *Parameter Setting*. Dordrecht: Reidel.

Yang, C. D. (2000). *Knowledge and Learning in Natural Language*. Doctoral dissertation, MIT.