Cue-based learners in parametric language systems: application of general results to a recently proposed learning algorithm based on unambiguous 'superparsing'

Stefano Bertolo, Kevin Broihier, Edward Gibson and **Kenneth Wexler**
Department of Brain and Cognitive Sciences - MIT
Cambridge, MA 02139
{bertolo,kevin,gibson,wexler}@psyche.mit.edu

Abstract

Cue-based learners have often been proposed as models of language acquisition by linguists working within the *Principles and Parameters* framework. Drawing on a general theory of cue-based learners described in detail elsewhere (Bertolo et al., 1997), we show here that a recently proposed learning algorithm (Fodor's Structural Triggers Learner (1997)) is an instance of a cue-based learner and that it is therefore unable to learn systems of linguistic parameters that have been proved to be beyond the reach of any cue-based learner. We demonstrate this analytically, by investigating the behavior of the STL on a linguistically plausible space of syntactic parameters.

Parametric Linguistics and Cue-based Learners

If, as has been proposed by Chomsky (1981), human languages all obey a common set of principles and differ from one another only in finitely many respects (often referred to as *parameters*) and in these respects only in finitely many ways (the *values* of the *parameters*), then human language learning can be seen as a search problem in a finite hypothesis space: the child does not need to hypothesize grammars that fall beyond those that are consistent with the common set of principles (often referred to as *Universal Grammar*) and any of the possible assignment of values to the linguistic parameters. However, although finite, this space of hypotheses can still be quite large (recent principled estimates place this number around 2^{40} different possible grammars[1]) and it is therefore imperative for any parametric model of language acquisition to show how such a huge hypothesis space could be searched effectively, that is, rapidly. Indeed, this process appears to be so rapid in children that it is difficult to find evidence for the incorrect setting of a parameter in any language. For a review of these findings see Wexler (1996), who formulates the hypothesis of *Very Early Parameter-Setting*, namely, that basic clause structure parameters are set before the child begins to produce multiple word utterances, i.e. approximately 1;6.

It has been observed (Dresher & Kaye, (1990), Brent (1996), Fodor (1997)) that this huge hypothesis space could be searched effectively if children were capable of establishing conclusively the value of certain parameters by attending to linguistic events of a particular nature in their environment. In fact, if all parameters are binary, conclusively establishing the value of a parameter eliminates exactly half of the hypotheses from the hypothesis space. Ideally, 40 such observations could be sufficient to single out a grammar out of 2^{40} possible alternatives.

[1]This estimate can be obtained by restricting all parametric variation to the ability or inability of functional heads to attract other heads or maximal projections and by estimating the number of functional heads that are required for descriptive adequacy. On this see Roberts (1996).

Test input string	Response if test positive
T_1: *two a's in a row?*	*set* p_1 *to value* $b \cup a$
T_2: *two b's in a row?*	*set* p_1 *to value* b^*
T_3: *two c's in a row?*	*set* p_2 *to value* $d \cup c$
T_4: *two d's in a row?*	*set* p_2 *to value* d^*

Figure 1: A battery of tests for parametric language learning

The following artificial example should help to give an idea as to what these *observations* could amount to.

Example 1 *Suppose you were trying to determine, from a collection of positive examples, which one of the following four regular expressions generates the sample:*

$$a\left\{\begin{array}{c} b\cup a \\ b^* \end{array}\right\}c\left\{\begin{array}{c} d\cup c \\ d^* \end{array}\right\}ef^*.$$

One way to solve this problem could be to set up a battery of tests (see figure 1) to be applied to each one of the positive samples and to make choices about the value assignment that is appropriate for each parameter depending on the outcome of these tests. In this construction, the observable event of a sample string having two a's in a row is taken as a cue to the $b \cup a$ *value assignment for the first parameter.*

The goal of the cue-based learning enterprise is to show that it is simultaneously possible to reconstruct linguistic variation parametrically and to single out in each possible target language a set of cues that would allow a learner to acquire the correct setting for each parameter. Further motivation for this enterprise comes from the study of language change. Lightfoot (1997) has argued that cue-based learning could explain the 'catastrophic' nature of historically attested language changes.

A theory of cue-based learners

Although the central intuition about the design philosophy of a cue-based learner emerges quite clearly from the example above, a formal characterization of the class of these algorithms turns out to be quite useful on at least two counts. First of all, a formal definition will make it possible to capture some essential design features in learning algorithms that appear to be prima facie unrelated. Second, by establishing learnability results about the class of cue-based algorithms at large one would automatically have results that can be applied to each individual algorithm. We will now introduce a formal theory of cue-based learners based on material discussed at length in Bertolo et al. (1997).

Since, as we saw, the salient feature of a cue-based learner is to restrict the hypothesis space of a parametric learning problem, we first need to introduce a definition of parameter spaces.[2]

Definition 1 *A parameter space $\mathcal{P}$ is a triple $< par, L, \Sigma >$, where Σ is a finite alphabet of symbols and par is a finite set of sets $\{p_1, \ldots, p_n\}$. Given a p_i in par, its members are enumerated as $v_i^1, \ldots, v_i^{|p_i|}$. Given the cartesian product $\mathbf{P} = p_1 \times p_2 \times \ldots \times p_n$, a parameter vector $\overline{P}$ is a member of $\mathbf{P}$. The function $L : \mathbf{P} \mapsto 2^{\Sigma^*}$ assigns a possibly empty subset of Σ^* to each vector $\overline{P} \in \mathbf{P}$. The expression $\mathcal{L}(\mathbf{P})$ denotes the set $\{L(\overline{P_1}), \ldots, L(\overline{P_{|\mathbf{P}|}})\}$.*

In example 1, the space has two parameters, $p_1 = \{b \cup a, b^*\}$ and $p_2 = \{d \cup c, d^*\}$. The alphabet Σ is $\{a, b, c, d, e, f\}$ and the function L takes as input couples of regular expressions x and y and outputs the set of all strings generated by the regular expression $axcyef^*$. Notice that there is nothing specifically 'linguistic' about this definition of a parameter space. The results that we report can therefore be applied to any search problem where the search space falls under definition 1 and the search algorithm under definitions 4 and 5.[3]

Given a parameter space, it turns out to be useful to be able to refer to an assignment of values to some, but not all of the parameters.

Definition 2 *Let $\mathcal{P}$ be a parameter space. A partial assignment in $\mathcal{P}$ is any subset B of $\bigcup_{p_i \in par}(\{p_i\} \times p_i)$ such that for every p_i in par there is at most one $< p_i, v_i^m >$ in B. Given two partial assignments A and B in $\mathcal{P}$, B is said to be A-consistent iff $A \cup B$ is also a partial assignment in $\mathcal{P}$.*

Example 2 *If $\mathcal{P}$ is a parameter space with parameters $p_1, p_2, \ldots, p_{37}$ then the set $A = \{< p_1, 0 >, < p_{17}, 1 >, < p_{31}, 0 >\}$ is a partial assignment in $\mathcal{P}$, because it specifies a consistent value assignment for some, but not all, of the parameters of $\mathcal{P}$. The set $B = \{< p_3, 0 >, < p_{17}, 0 >, < p_{28}, 0 >\}$ is also a partial assignment in $\mathcal{P}$. However, B is not A-consistent since $A \cup B$ is not a partial assignment in $\mathcal{P}$, because it includes two conflicting value assignments to parameter p_{17}.*

Such partial assignments can in turn be used to isolate only those parts of a parameter space that agree on the values assigned to the parameters in a partial assignment. Crucially, such a portion of a parameter space is, by definition 1, itself a parameter space.

Definition 3 *Let $\mathcal{P}$ be a parameter space $< par, L, \Sigma >$. Then $\mathcal{P}[\emptyset] = < par^{\emptyset}, L, \Sigma > = < par, L, \Sigma > = \mathcal{P}$. If $\mathcal{P}[A]$ is a parameter space $< par^A, L, \Sigma >$ and B is an A-consistent partial assignment in $\mathcal{P}$, then the subspace $\mathcal{P}[A \cup B]$ is the parameter space $< par^{A \cup B}, L, \Sigma >$ such that, given $H = \bigcup_{x \in B} \pi_1(x)$, (where $\pi_1(x)$ denotes the first element of the ordered pair x) if $p_j \notin H$ then $p_j^{A \cup B} = p_j^A$ and if $p_j \in H$ then $p_j^{A \cup B} = \{v_j^m\}$ where v_j^m is the only $v \in p_j$ such that $< p_j, v_j^m > \in B$. Finally, $\mathcal{P}[A \cup \overline{B}]$ is the parameter space $< par^{A \cup \overline{B}}, L, \Sigma >$ where, for every p_i in H, $p_i^{A \cup \overline{B}} = p_i^A - p_i^{A \cup B}$ and, for every p_i not in H, $p_i^{A \cup \overline{B}} = p_i^A$.*

Example 3 *If A is the set of assignments $\{< p_1, 0 >, < p_2, 1 >, < p_3, 0 >\}$, $\mathcal{P}[A]$ denotes the portion of the parameter space $\mathcal{P}$ in which all languages agree on the values 0, 1 and 0 for parameters p_1, p_2 and p_3 respectively.*

We are now ready to formalize the notion of some parameter values being established as a result of observing certain events in the linguistic environment.

Definition 4 *Let $\mathcal{P}$ be a parameter space, B a subset of the set B^* of all partial assignments in $\mathcal{P}$ and C a non-empty subset of $\bigcup_{\overline{P} \in \mathbf{P}} L(\overline{P})$, a cue function of window size n for $\mathcal{P}$ is a function $\phi_C : \bigcup_{i=1}^{n} C^i \times B \mapsto B^*$ where C^i is the cartesian product of C with itself i times.*

The function ϕ_C can be seen as a formal representation of the battery of tests discussed in example 1. There, C is the set of all strings that have two a's, b's, c's or d's in a row, the size of the window is just one and ϕ_C returns value assignments to parameter independently of whatever assignments have already being established. So, for example,

$$\phi_C(< aace, \emptyset >) = < p_1, b \cup a > \text{ but also}$$

$$\phi_C(< aace, \{< p_2, d \cup c >\} >) = < p_1, b \cup a >$$

It is important to notice that definition 4 generalizes our original intuition in two important respects. First of all, it captures the possibility that the learner, upon observation of a linguistic event, could reach different conclusions depending on what its current state of belief (assignment of value to certain parameters) is. Secondly, it allows for the existence of linguistic events that can only be observed comparing n distinct data points (in the case of syntax learning, typically sentences).

Finally, a cue-based learner is a learning algorithm that does all its learning via a cue function. The crucial feature of such learners is the absence of any form of backtracking: as definition 5 shows, if the cue function returns any parameter assignment that is not in agreement with the current assignment, the inconsistent portion of the output of the cue function is simply discarded.[4]

Definition 5 *Let $\mathcal{P}$ be a parameter space, B^*, B and C as in definition 4 and ϕ_C a cue function of window size n for $\mathcal{P}$. A cue-based learner for $\mathcal{P}$ is a function*

$$\lambda_C : \{\mathcal{P}[A] | A \in B\} \times \bigcup_{i=1}^{n} C^i \mapsto \{\mathcal{P}[A] | A \in B^*\}$$

[2]An alternative, equivalent, definition can be found in Frank & Kapur (1996).

[3]For example, an appropriately empoverished version of Version Spaces (Mitchell 1978).

[4]For this reason, in its present formulation, this theory of cue-based learners cannot represent the notion of a 'default' value for a parameter which is so crucial for Dresher and Kaye's algorithm. A suitable extension that captures this notion and preserves the result reported here is discussed in Bertolo et al. (1997).

such that

$$\lambda_C(\mathcal{P}[A], \overline{s}) = \begin{cases} \mathcal{P}[A] & \text{if } \overline{s} \notin \bigcup_{i=1}^{n} C^i \\ \mathcal{P}[A \cup B_A] & \text{if } \phi_C(\overline{s}, A) \text{ is not} \\ & A\text{-consistent} \\ \mathcal{P}[A \cup \phi_C(\overline{s}, A)] & \text{otherwise} \end{cases}$$

where B_A is the largest A-consistent subset of $\phi_C(\overline{s}, A)$.

As is standard in formal models of language acquisition[5], we characterize cue-based learners as successful if and only if, for every possible target in a parameter space they eventually output a parameter space that contains only one language and that language is at least weakly equivalent to the actual target (that is, it generates exactly the same set of strings, although possibly with a different structural description). Given this characterization, we were able to prove a general result showing the existence of a property (the Global Natural Subspace Property – GNSP) that is necessary and sufficient for a parameter space to be learned by a cue-based learner.

Definition 6 *Let $\mathcal{P}$ be a parameter space and A a partial assignment. $\mathcal{P}[A]$ is said to have the Natural Subspace Property (NSP) iff either $|\mathbf{P}^A| = 1$ (i.e., the subspace contains a single language) or, for every $\overline{P} \in \mathbf{P}^A$, there is an $\overline{s} \in L(\overline{P})^i$ and an A-consistent partial assignment B such that $\overline{P} \in \mathbf{P}^{A \cup B}$ and, for every $< p_i, v_i^m > \in B - A$,*

$$\overline{s} \notin \bigcup_{\overline{P'} \in \mathbf{P}^{A \cup \{<p_i, v_i^m>\}}} L(\overline{P'})^i.$$

$\mathcal{P}$ has the Global NSP (GNSP) iff, for every partial assignment A, $\mathcal{P}[A]$ has the NSP.

To exemplify, the parameter space of example 1 has the GNSP because, in every subspace $\mathcal{P}[A]$, every language $L(\overline{P})$ contains at least a string s for which a relevant parameter p_k can be found, in the sense that s does not belong to any language of $\mathcal{P}[A]$ in which p_k is set differently than in $L(\overline{P})$.

As a special case of that result we proved that if in a parameter space there are two or more weakly equivalent languages, then the GNSP does not hold and, as a consequence, no cue-based learner can successfully learn the space.[6]

In the next section we will apply this result by proving that Fodor's Structural Triggers Learner (STL) cannot learn a plausible space of syntactic parameters that is an extension of the space of parameters (discussed in Gibson & Wexler, 1994) that the STL was designed to learn. This proof will be obtained in two steps: we will first show that the STL is indeed a cue based learner, as characterized in definition 5[7] and then we will show an extension of Gibson and Wexler's original space of syntactic parameters that contains clusters of weakly equivalent languages.

[5]For a general discussion of learnability criteria in systems with weak equivalences see Wexler & Culicover (1980). For learnability in parameter spaces, see Dresher and Kaye (1990), Clark (1992), Gibson & Wexler (1994), Frank and Kapur (1996), Niyogi & Berwick (1996)

[6]More precisely, we proved that the space can be learned only if the cue function on which the learner is based is *arbitrary*, in the sense that it returns different parameter values upon exposure to linguistic events that cannot be discriminated by any battery of tests, in the sense of the example discussed above. We take the restriction to *non-arbitrary* cue functions to be justified on grounds of psychological plausibility.

[7]It is impossible to prove anything of the kind without a rigorous definition of the class of cue-based learners. Hence the need for the somewhat lengthy definitions.

Comp-Head	Spec-Head	Verb-Second
VP → Verb Obj	IP → Subj I-bar	C → ∅
VP → Obj Verb	IP → I-bar Subj	C → Verb

Figure 2: Rules associated with the Comp-Head, Spec-Head and Verb-Second parameters

The limits of learning via unambiguous 'superparsing'

In her analysis of Gibson and Wexler's space of syntactic parameters Fodor (1997) discusses the advantages of a parametric learner that does not set the value of any parameter until the correct value has been conclusively established and does not abandon any value assignment that has been conclusively established. Fodor's insightful idea consists of showing that it should be possible to establish conclusively the value of syntactic parameters by means of a mechanism that a language learner must possess anyway, that is a parser.

Fodor speculates that, if it is possible to reconstruct the values of each parameter as alternative sets of grammatical rules whose application could be detected by a 'superparser' using their union set as its grammar, then the task of parametric language learning could be reduced to the task of unambiguously detecting the application of a given rule in all the 'superparses' of an input sentence. For concreteness, if the rules in figure 2 are associated to alternative values of the Comp-Head, Spec-Head and Verb-Second parameters,[8] then a sentence such as

(1)	*Max*	*sah*	*Ute*
	Max	saw	Ute
	'Max saw Ute'		

is *parametrically ambiguous* with respect to the Verb-Second parameter. In fact, the following are among the parses it generates with respect to a supergrammar that contains *all* of the rules in figure 2:

(2) $[_{CP}[_{SpecC}$ Max$][_{\overline{C}}[_C$ sah$][_{IP}\ldots\ [_{VP}$ Ute t$]]]]$

(3) $[_{CP}[_{SpecC}\emptyset][_{\overline{C}}[_C\emptyset][_{IP}\ [_{SpecI}$ Max$]\ [_{VP}$ sah Ute$]]]]$

As is apparent, while parse 2 uses the rule associated with the positive value of the Verb-Second parameter (C → Verb), parse 3 uses the rule associated with the negative value (C → ∅). As a consequence, the superparser cannot rely on sentence 1 to establish conclusively the value of the Verb-Second parameter. It can, however, rely on the following sentence

(4)	*Plötzlich*	*sah*	*Max*	*Ute*
	Suddenly	saw	Max	Ute

[8]These parameters determine respectively whether complements follow or precede their heads, whether specifiers follow or precede their heads and whether or not the verb of root clauses moves to the C position, causing Spec-C to be filled with some other material, typically a maximal projection.

```
1. Initialize the grammar G to be the union set of UG and all the values of all parameters;
2. Initialize the set A of parameter assignments to the empty set and the set P to include all parameters in the parameter space;
3. If, for every parameter $p_i$ there is an assignment $< p_i, v_i^m >$ in A, return A. Otherwise go to 4.
4. Receive a sentence s from the linguistic environment
   (a) initialize to the empty set the set N of parameter assignments revealed by s;
   (b) list all parses of s;
   (c) list the set R of all the rules that have been employed in *every* parse of s;
   (d) for every parameter $p_i$ in P and every value $v_i^m$ of $p_i$ compute the intersection $v_i^m \cap R$;
   (e) for every $v_i^m$, if $v_i^m \cap R \neq \emptyset$ set $N = N \cup \{< p_i, v_i^m >\}$;
5. Set $A = A \cup N$
6. Go to 3;
```

Figure 3: Structural Triggers Learner Algorithm

'Suddenly, Max saw Ute'

which, although ambiguous with respect to the Comp-Head parameter, cannot be parsed unless the rule associated with the positive value of the Verb-Second parameter is used:

(5) $[_{CP}$ Plötzlich $[_{\overline{C}}[_{C}$ sah$][_{IP}[_{SpecI}$ Max$]$ $[_{VP}$ Ute t$]]]]$
(6) $[_{CP}$ Plötzlich $[_{\overline{C}}[_{C}$ sah$][_{IP}[_{SpecI}$ Max$]$ $[_{VP}$ t Ute$]]]]$

The STL algorithm is a cue-based learner

If, as we have assumed in the exposition of Fodor's idea of parameter learning via 'superparsing', UG can be regarded as a set of rules and each value of each parameter is also a set of rules, then a formalization of the 'superparsing' algorithm (figure 3) will reveal straightforwardly that it is an instance of a cue-based learner. We will start by observing that step 4 of the STL algorithm could be seen as the function

$\phi_C(s, A) = \{< p_i, v_i^m > \mid \text{every parse of s requires } v_i^m\}$

Since, by construction of step 2 of the STL, both the set A and the set $\{< p_i, v_i^m > \mid \text{every parse of } s \text{ requires } v_i^m\}$ are partial assignments in the sense of definition 2, by definition 4 step 4 of the STL is a cue function. Three comments are in order here:

1. In this particular case the set C in the subscript of the cue function is simply the union set of all the languages in the parameter space. In other words, every string of every language is in a sense a cue. All the sentences that are parametrically ambiguous with respect to every parameter, however, can only cue the STL for the empty parameter assignment.
2. The cue function on which the STL relies is insensitive to the current assignment of values to the parameters. In other words, although we represented ϕ_C as a function of both s and A, the output of ϕ_C only depends on s.
3. Although definition 4 is general enough to include cue functions that take as input linguistic events consisting of sequences of n strings, for any number n, the cue function on which the STL relies takes as input only individual strings. Such restriction could be recommended on grounds of psychological plausibility. More generally, the smaller the window on which the cue function operates, the smaller the memory requirement of a cue-based learner.

Having shown that step 4 of the STL is a cue function ϕ_C, all we have to show is that the whole STL algorithm can be seen as a function λ_C that falls under definition 5. But this is certainly true, since, by construction, the STL extracts all the parameter assignments from the cue function based on the 'superparser'.[9]

A space of syntactic parameters

Having shown that the STL is indeed an instance of a cue-based learner we will now turn to a brief description of a space of syntactic parameters that is an extension of the space the STL was originally designed to learn. The space we present here has been implemented in order to investigate the interactions of seven distinct parameters that have been proposed by linguists on grounds of descriptive adequacy.[10]

In addition to the Complement-Head, Specifier-Head and Verb-Second parameters that have been briefly introduced above and discussed at length in Gibson and Wexler (1994), we have analyzed the effects of alternative settings for the four parameters that are here briefly described.

The Complementizer-Direction Parameter In languages such as Japanese

(7) *John-wa*	*Mari-ga*	*kawaii-da*	*to*	*omotteiru*
John-top	Mary-nom	pretty-is	that	think

'John thinks that Mary is pretty'

the position of the complementizer 'to' with respect to the embedded clause can indeed be derived as a special case of the setting of the Complement-Head parameter. In particular, just as the direct object precedes its head, the verb, in Japanese embedded clauses precede their head, the complementizer.

There are other languages, however, in which this correspondence is not observed. In particular, in German, which is also analyzed as a head-final language, embedded clauses do not precede but follow the complementizer:

(8) *Johann*	*glaubt*	*dass*	*Marie*	*hübsch*	*ist*
John	thinks	that	Mary	pretty	is

'John thinks that Mary is pretty'

In order to cover this case we have introduced a Dir-Comp parameter that determines the relative order of complementizers and the highest inflectional projection, independently of

[9]In fact, we should also prove that step 5 of the STL is guaranteed never to output an inconsistent parameter assignment, as is required by definition 5. For this to happen, there must be a target language L whose p_i-th parameter is set to value v_i^m which has two sentences s_1 and s_2 such that s_1 only belongs to languages that have p_i set to value v_i^m and s_2 only belongs to languages that have p_i set to a different value v_i^k. However, this is impossible, since, by hypothesis, s_2 is a member of L and so it belongs to at least a language that has p_i set to value v_i^m.

[10]A LISP implementation can be obtained from the first author.

the value of the Complement-Head parameter. In this respect a language like Japanese would be a complementizer-final language while German would be a complementizer-initial language.

The Verb-to-Agr Parameter Pollock (1989) has observed that French and English differ from one another, among other things, in the possibility of lexical material intervening between the verb and its direct object in French but not in English:

(9) *Jean*	*embrasse*	*souvent*	*Marie*
John	kisses	often	Mary

'John often kisses Mary'

(10) *John kisses often Mary

In fact, in French VP adverbs like 'souvent' are actually required to appear between the verb and its object as the ungrammaticality of the following example shows:

(11) *Jean souvent embrasse Marie

Pollock has explained such variation assuming that in French, but not in English, verb movement to Infl is obligatory. Since in our system of parameters we adopt a version of Pollock's split-Infl hypothesis, splitting Infl into Tense and Agr (with Tense dominating Agr), this parametric variation is reconstructed by leaving the verb in situ for one value of the parameter and moving it obligatorily to Agr for the alternative value.

The Verb-to-T Parameter No possible value assignment to the Comp-Head, Spec-Head, Dir-Comp, Verb-Second and V-to-Agr parameters could capture the VSO word order that is typical of Celtic languages such as Irish or Breton, if we assume, as we do in our system, that subjects are generated in Spec-Agr. In order to capture the VSO word order, we have therefore followed Carnie's (1995) analysis of Irish and introduced a second verb movement parameter that moves the verb to Tense. The V-to-Agr and the V-to-T movements are analyzed in our parameter space as sequential movements. Depending on the value of V-to-Agr, the verb does or does not move to Agr. After this, if the V-to-T parameter is set to the value that requires the movement to take place, the verb is moved from wherever is was (either V or Agr) to T.

The Embedded-Verb-Second Parameter This parameter determines whether verb-second movements take place in embedded clauses. Among those languages that exhibit verb-second effects in root clauses it is possible to draw a distinction between those that have obligatory verb-second movements in embedded clauses and those that make such movements possible (and in that case also optional) only when the embedded clause is being introduced by verbs of a special kind. For example, in Danish and German, embedded verb-second movements exist only if the matrix verb is a "bridge-verb" a verb expressing a cognitive state such as 'believe', 'hope' and the like (Iatridou and Kroch 1992, Vikner 1994).

Clusters of weakly equivalent languages

In our system of parameters, as it is currently implemented, the only lexical material that could possibly intervene between the Agr and the Tense head is a subject NP in Spec-Agr. It is important to realize, however, that this only happens

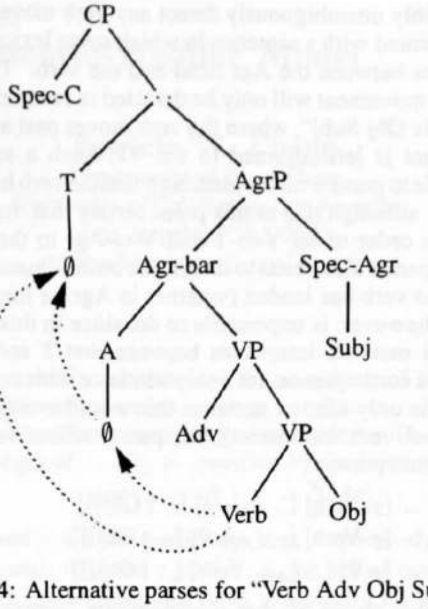

Figure 4: Alternative parses for "Verb Adv Obj Subj"

when complements and specifiers are positioned in opposite directions with respect to their heads, as it would happen in languages that are comp-final and spec-initial or comp-initial and spec-final. When a language is either comp-final and spec-final or comp-initial and spec-initial, however, the subject NP appears in the string either after or before both the Tense and Agr (or Agr and Tense) positions respectively. When this happens, the order of lexical elements in a string is not sufficient to determine the internal structure of a sentence with respect to the V-to-Agr and V-to-T movement and as a result, three distinct parameter assignments (V-to-T+ and V-to-Agr+; V-to-T+ and V-to-Agr–; V-to-T– and V-to-Agr+) end up generating exactly the same set of strings.

In order to see how this state of affairs is problematic for the STL, let's step through the case in which the target language is spec-final, comp-final, complementizer-initial, verb-second–, embedded-verb-second– v-to-t+ and v-to-agr+. To simplify the discussion, let's assume that the STL has already correctly determined the values of all parameters but V-to-T and V-to-Agr.[11] To start with, it is easy to see that the learner could

[11] It is actually instructive to see why a single sentence such as "Verb Comp Verb Obj Subj Subj" would be sufficient for the STL to establish this. The fact that the matrix verb appears in the first position is an unambiguous indication that the language cannot be verb-second+. So it has to be verb-second–, and therefore also embedded-verb-second–. The subject of the matrix clause follows the matrix verb, which indicates that the language is spec-final. The fact that the embedded clause follows the matrix verb reveals that the language is comp-final. Finally, since the complementizer of the embedded clause precedes the embedded verb, the language is complementizer-initial. The fact that so many parameter values can be fixed on the basis of a single degree-1 sentence is an attractive feature of the STL. The cost of computing all possible 'superparses' for a sentence, however, especially at the beginning, when the 'supergrammar' is still very large, may offset this advantage. We are

not possibly unambiguously detect any verb movement until it is presented with a sentence in which some lexical material intervenes between the Agr head and the verb. This means that verb movement will only be detected in sentences such as "Verb Adv Obj Subj", where the verb moves past a preverbal adverb that is left adjoined to the VP. Such a sentence is impossible to parse without assuming that the verb has moved. However, although it is at this point certain that the verb has moved, in order to set V-to-T and V-to-Agr to their correct value the parser also needs to determine unambiguously where exactly the verb has landed (whether in Agr, or higher up, in T). This, however, is impossible to do, since in this language no lexical material intervenes between that T and the Agr head. As a consequence, for every sentence with a pre-verbal adverb (the only kind of sentence that would reveal anything at all about verb movement), the parser will *always* return three distinct parses:

(12) $[_{CP} \ldots [_{T}$ Verb$] [\ldots [_{Agr}$ t$] [\ldots$ t Obj$]]]$

(13) $[_{CP} \ldots [_{T}$ Verb$] [\ldots [_{Agr} \emptyset] [\ldots$ t Obj$]]]$

(14) $[_{CP} \ldots [_{T} \emptyset] [\ldots [_{Agr}$ Verb$] [\ldots$ t Obj$]]]$

What this shows is that unambiguous 'superparsing' is *never* possible for strings coming from a cluster of weakly equivalent languages. But since the STL learns the value of parameters *only* via unambiguous 'superparsing' this means in turn that in this particular parameter space the value of certain parameters is never acquired by the STL. It should be noted that the problem we have cited is *not* simply that the cue-learning mechanism converges on a grammar that is only weakly equivalent to the "correct" grammar; rather, the problem is that the cue-learning mechanism doesn't converge on a fully-specified parameter-setting at all; the relevant parameters remain unset, for *any* cue-learning mechanism.

Conclusions

As usual, whenever a negative learnability result is provided, it is possible to proceed in two different directions, depending on what assumptions one is more attached to. For example, one could argue that the parameter space should be refined by including data that would reveal the actual landing site of verb movement and make unambiguous 'superparsing' feasible. Alternatively, one could modify the STL algorithm, by allowing a certain amount of guessing when the learner gets pushed up against a cluster of weakly equivalent languages. Such a change would obviously place the resulting algorithm outside the class of cue-based learners. It must be noted, however, that this solution encounters several problems. First of all, there is an issue of 'timing': how would the learner know when it's time to stop superparsing and to start guessing? Secondly, this solution is guaranteed to work only when *all* the languages in the remaining subspace are equivalent to one another. The present case shows however that, since this situation doesn't always obtain, non-deterministic extensions of the STL are not always guaranteed to succeed. Observe, in fact, that a random assignment of value to the V-to-T and V-to-Agr parameters has one chance in four of returning v-to-t- and v-to-agr- yielding a language that, as the reader can verify, is distinct from the intended target (v-to-t+; v-to-agr+). Such cases are discussed at length in Bertolo et al. (1997). Attempts to circumvent this second problem by letting the learner select a random value assignment *only among those settings that yield weakly equivalent languages* would require the learner either to have access to a look-up table that lists all the language equivalences or to be able to compute them.

currently investigating this issue.

Acknowledgments

Stefano Bertolo was supported by a post-doctoral fellowship from the McDonnell-Pew foundation, Kevin Broihier (grant DIR 9113607) and Edward Gibson and Kenneth Wexler (grant SBR-9511167) were supported by grants from NSF.

References

Bertolo, S. Broihier, K., Gibson, E. & Wexler, K. (1997). Characterizing learnability conditions for cue-based learners in parametric language systems. MIT manuscript.

Brent, M. R. (1996). Advances in the computational study of language acquisition. *Cognition*, 61, 1–38

Carnie, A. (1995) Non-Verbal Predication and Head-Movement. Doctoral dissertation. Cambridge: MIT.

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

Clark, R. (1992). The selection of syntactic knowledge. *Language Acquisition*, 24, 299–345

Dresher, E. & Kaye, J. (1990). A computational learning model for metrical phonology. *Cognition*, 34, 137–195

Fodor, J. (1997). Unambiguous Triggers. To appear in *Linguistic Inquiry*.

Frank, R. & Kapur, S. (1996). On the Use of Triggers in Parameter Setting. *Linguistic Inquiry*, 27(4), 623–660

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

Iatridou, S. & Kroch, A. (1992). The licensing of CP-recursion and its relevance to the Germanic verb-second phenomenon. *Working Papers in Scandinavian Syntax*, 50, 1-24

Lightfoot, D. (1997). Catastrophic change and learning theory. To appear in *Lingua*.

Mitchell, T. M. (1978). Version Spaces: An approach to concept learning. Doctoral dissertation. Stanford University.

Niyogi, P. & Berwick R. C. (1996) A language learning model for finite parameter spaces. *Cognition*, 61(1-2), 161–193

Pollock, J.Y. (1989). Verb Movement, Universal Grammar, and the structure of IP. *Linguistic Inquiry*, 20(3), 365–424

Roberts, I. (1996). Language change and learnability. In S. Bertolo (Ed.), *Learnability and Language Acquisition: a self contained Tutorial for Linguists* (pp.66–82). MIT Manuscript.

Vikner, S. (1994). *Verb Movement and Expletive Subjects in the Germanic Languages*. Oxford: Oxford University Press.

Wexler, K. (1996). The Development of Inflection in a Biologically Based Theory of Language Acquisition. In M.L. Rice (Ed.), *Toward a Genetics of Language*. Mahwah, NJ: Lawrence Erlbaum Assoc.

Wexler, K & Culicover, P.W.(1980). *Formal Principles of Language Acquisition*. Cambridge, MA: MIT Press.