AMBIGUITY AND THE COMPUTATIONAL FEASIBILITY OF SYNTAX ACQUISITION

by

WILLIAM GREGORY SAKAS

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York

2000

This manuscript has been read and accepted for the Graduate Faculty in Computer Science in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

Date	Dr. Virginia Teller Chair of Examining Committee
Date	Executive Officer

Dr. Janet Dean Fodor, Graduate Center
Dr. Cullen Schaffer, Hunter College
Dr. Robin Clark, University of Pennsylvania

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

Abstract:
Ambiguity and the Computational Feasibility of Syntax Acquisition
by
William Gregory Sakas

Advisor: Professor Virginia Teller

The thesis presents a framework that can be used for empirical and formal analysis of parameter setting models of language acquisition. Such models attempt to mirror computationally the process by which children acquire the grammar of their native language. Research into formal language learning theory standardly focuses on issues of *learnability* – Under what conditions is learning possible? The thesis contributes to the important, but under-investigated question of *feasibility* – Is acquisition possible within a reasonable amount of time and/or with a reasonable amount of work? The proposed framework formalizes existing notions such as the rate of parametric ambiguity and parametric expression within a generally defined parameter space, so that different types of learning algorithms and grammar spaces can be explored. Two influential learning algorithms are examined in detail: *The Triggering Learning Algorithm* (Gibson and Wexler, 1994) and *The Structural Triggers Learner* (Fodor, 1998). Empirical results indicate that the Triggering Learning Algorithm's simple hill-climbing search heuristics are sufficient to acquire the target grammar without the learner's consumption of an unreasonable number of input sentences when the learning space contains a strong correlation between the similarity of languages and the grammars that generate them. The results also indicate that the Structural Triggers Learner's use of

structural information lying beneath the surface word order of an input sentence allows for feasible learning when the rate of parametric expression varies across the input sentences encountered by the learner. Notably, however, both models are acutely sensitive to changes in the amount and type of ambiguity present in the domain. A small change in just one of the factors that contributes to the distribution of ambiguity has a large impact on learning efficiency.

Preface

The research presented here is both the result of several years of work and a springboard to work that is currently underway. Much of the research leading up to this dissertation has been published and is publicly available. A stylistic choice was made not to include previously published results in this manuscript. Likewise, the latest simulation results are also not included; updated versions of the dissertation will be available.

Contact me if you are interested in obtaining any of my referenced works or the most current version of the dissertation. My email address is:

Sakas@hunter.cuny.edu

or by post:

William Sakas

Department of Computer Science

Hunter College

695 Park Avenue

New York, N.Y. 10021

Acknowledgements

After working as a high school teacher, professional musician, computer programmer, and florist, I landed in my first course in graduate school. My advisor, Virginia Teller, was co-teaching a seminar in cognitive science. Now, full circle, at the end of my tenure as a graduate student, she has supported me throughout the writing of this manuscript; reading drafts I delivered both on and (mostly) off schedule, and providing invaluable feedback on every page. I owe her a great debt. Janet Fodor introduced me to psycholinguistics and provided much inspiration and intellectual guidance while I developed the ideas contained in the thesis. I will always be grateful for her unwavering faith in me both as a student and a collaborator. I must also thank Cullen Schaffer for supporting my initial venture into the world of machine learning and for his work that motivates the *No Best Strategy Conjecture* of the last chapter. Robin Clark's groundbreaking research lays the foundation for much of the dissertation; for this and for his participation on my exam committee, I am very grateful.

My mother, brother and sister were very supportive as I disappeared into the final stages of writing – the (sometimes) tactful prodding, "Is it finished, yet?" helped keep me from working on the thesis into yet another millenium. Finally, to my best friend and wife, Mari, I will always be grateful for your special support, patience, and smile when I was most stressed and irritable. I am truly a very lucky fellow. Oh! To Cookie and Ally: Thanks poochies, for taking me on all those long, early morning walks – they really helped clear my head.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Why Computationally Model Language Learning?

> *"...it may be necessary to find out how language learning could work in order for the developmental data to tell us how it does work." (Pinker, 1979)*

How does a child acquire language[1]? The process has been looked at with some scrutiny from many fields within cognitive science. Linguists endeavor to describe language facts in such a way that the same theory can accommodate linguistic phenomena in all natural languages (Chinese, Swahili, German, etc.) with the strong prerequisite that the properties of individual languages can be readily acquired on the basis of the kinds of evidence that are available to children. Developmental psychologists examine the speaking patterns of both parents and children and look for relationships between them and the nature of language skills at different ages or developmental stages. Mathematicians examine formal models of language in order to establish bounds on what can and can't be learned in principle and computer scientists

[1] Unless otherwise stated, throughout I will use *language* to refer to the syntactic structure or grammar of language and put lexical (and phonological, etc.) considerations aside (e.g. it is irrelevant here whether a dog is called "dog" or "chien"). There has been much recent work on lexical, phonological and semantic acquisition (see Brent 1996 and references therein for a survey, and Tesar and Smolensky 1998 for an approach to phonology acquisition within an Optimality Theoretic Framework).

develop and implement models and run computer simulations of the acquisition process.

Recently there has been renewed interest in the computational and mathematical modeling of language acquisition and the interrelationship between such models and linguistic and psycholinguistic theory. This follows a general trend in cognitive science towards multidisciplinary research (c.f. Schunn 1998). The hope is that through the implementation and/or mathematical analysis of such models, certain bounds can be established which can be brought to bear on pivotal issues in developmental psycholinguistics.

As an example of the interplay between computational modeling and human language acquisition research, consider this seminal learnability result by the mathematician Edward Gold in 1967.[2]

> Exposed to input strings of an arbitrary target language (L_{targ}), where L_{targ} is a member of the class of all possible hypothesis languages ($\mathcal{H}$), it is impossible to guarantee that a learner can identify L_{targ}, if $\mathcal{H}$ is any class in the Chomsky hierarchy.[3] Moreover, no learner is uniformly faster than one that executes simple error-driven enumeration of languages.

Gold's theorem, that the formal languages of the Chomsky hierarchy are unlearnable, is frequently cited in debates over what (if any) features of language are innate (c.f. Pinker 1979, Elman et al. 1996, Pullum 1996, Marcus, in press). Developmental

[2] What follows is an informal formulation of Gold's theorem; see the original work for discussion and proof.

[3] Actually the theorem holds even if $\mathcal{H}$ is less expressive. If $\mathcal{H}$ contains all the finite languages and just one non-finite language, $\mathcal{H}$ is unlearnable.

psycholinguistics has demonstrated that human children learn their grammar largely or solely from exposure to sentences from their native language – without any significant degree of guidance or correction from their parents or caregivers (Brown and Hanlon 1970, Marcus 1993). This, together with Gold's proof that, in general, identification of a target language is impossible, helps to impel the nativist claim that children must possess some congenital knowledge of language. In rejoinder, the empiricists either attack the linguistic plausibility of Gold's formal paradigm or take Gold's learning scenario to heart and attack the *no negative evidence* assumption that was crucial to the proof. My intention is not to rehash or review this ongoing debate here. Rather I point to the fact that Gold's computational work has remained current within cognitive science[4] and serves as an indication of the significance of formal, computational modeling of language learning.

1.2 The Linguistics of the Generative Tradition

Within the *generative tradition*, syntactic entities are combined by rules to generate phrases and sentences. Established by Noam Chomsky (1957, 1965), this research agenda attempts to ascertain both the specific rules of a particular language (e.g. English) as

[4] Although frequently misunderstood. The general misconception stems from the fact that Gold was concerned with the strictly defined classes that make up the Chomsky Hierarchy. By constraining $\mathcal{H}$ so that it is not strictly the class of regular languages (or context-free languages, etc.) but rather a union of subsets of different classes, learnability can be achieved. The computer scientist Dana Angluin provides a proof that establishes necessary and sufficient conditions to guarantee the learnability of $\mathcal{H}$ (Angluin, 1980).

well as those general principles underlying the language competence of *all* human speakers regardless of their native tongue (i.e. the Universal Grammar or UG).

The starting point of this tradition was a grammar formalism composed of *phrase structure* or *rewrite rules*, which are recursively applied to yield the *phrase structure* (PS) of an utterance. Although Chomsky emphasized early on that grammars based on phrase structure rules alone are insufficient to adequately capture all natural language phenomena,[5] the formal study of PS grammars, under the heading of *formal language theory*, remains to this day an important topic in computer science and mathematics.

In order to increase the descriptive power of PS grammars, many generative linguists embrace the notion of *transformational rules* (introduced by Chomsky, 1957). Roughly, the idea is that a structure generated by a PS grammar can be modified or operated on to produce a second structure that captures some aspect of language that would have been impossible or clumsy to capture with PS rules alone (e.g. the relationship between a passive sentence and its active counterpart). More than one transformation may apply in the derivation of a sentence, so a sentence may have many phrase structures 'en route' to the final one. The initial and final structures of a derivation are generally singled out. The structures serve different functions depending on the specifics of the linguistic theory and are referred to by different names accordingly. Since the analysis

[5] This early view that PS grammars lacked sufficient descriptive power has not been entirely sustainable. There exist fruitful research programs on enriched varieties of PS grammars (e.g. HPSG and Categorial Grammars) that have managed to overcome at least some of what were perceived as insuperable barriers early on.

presented in the thesis does not directly depend on the details of any particular linguistic theory, I will assume two generic levels and refer to the sentential structures as the *base-generated structure* (before any transformations have been applied) and *surface structure* (after all transformations).

Of importance to most past and current generative theories is the *movement* transformation operation. Movement changes the location of a syntactic item from its place in the original base structure to a new location in the surface structure. For example, in English, wh-pronouns are moved to the front of a sentence in the formation of questions.

(1) He will give *what* to the dog.

(2) What will he give *t* to the dog?

Structure (1) is the base-generated phrase structure. The PS grammar generates (1), and a movement transformation moves *what* up to the front of the surface structure, leaving behind a non-overt syntactic element called a *trace* (*t*), which results in surface string (2). (Another transformation – subject-auxiliary inversion, reverses the order of *he* and *will*.) It is important to note, especially in the study of acquisition, that the base structure does not appear in the linguistic environment perceived by the learner. That is, during the course of learning, a child has only surface structures to work with and no exposure at all to base structures. Furthermore, it is a fact of human languages (at least

in a generative account) that different base structures may give rise to the same surface word string. This may occur within a single language (e.g., *Flying planes can be dangerous*). It can also occur in cases of cross-linguistic *ambiguity*; where the rules that make up the grammars of different languages generate the same surface string. For example, a surface sequence of *Subject Verb Object* may either be generated as a base structure with no movement (as is common in English) or be derived from a base structure such as *Subject Object Verb* by movement of the verb to the second position (as occurs in German).[6] Many examples of cross-linguistic ambiguity, such as this one, exist. It seems reasonable to assume that the degree of ambiguity in natural language is quite high.[7] Ambiguity is a natural enemy of language acquisition and, as I will put forth in what follows, is a critical factor in determining the feasibility of an acquisition model.

Transformational rules allowed linguists to describe many complex facts of natural language which PS grammars could handle only with difficulty or not at all. However, as the facts about specific natural languages were explored in more and more detail, the explanatory power of the transformations themselves came into question. Many transformations were essentially ad hoc, descriptive statements of observed facts, and were postulated with great liberality. As Chomsky and others noted, if there were no

---

[6] Natural languages also exhibit within-language ambiguity – where a surface string may be generated by different sequences of grammar rules drawn from the same grammar. For example in the sentence: *Mari saw Momo the cockatiel on the TV*, English grammar allows the prepositional phrase: *on the TV* to modify either *Momo the cockatiel* or *saw*. From this point, I will use the term *ambiguity* to refer to cross-language ambiguity.

[7] The exact extent of cross-language ambiguity is still an important open question. See Chapter 5 for a brief discussion.

limits on the class of possible transformations for natural language, transformational grammars might *describe* but not *explain* the properties of natural languages. It is worth noting that Chomsky (1965 and since) defined *explanatory adequacy* in terms of language acquisition – an explanatory theory clarifies why a learner selects the grammar she or he does, when faced with a sample of the sentences of the language. Early transformational theory permitted too many choices, and offered no good criteria for the learner's selection among them.[8]

Eventually, it proved most successful not to set limits on a class of transformational rules, but to allow a maximally general transformation (*move* α, or even *affect* α) and to call on lexical facts and some general principles of syntactic organization to constrain the legitimate output of any transformational operation. A similar shift occurred in the conception of the PS grammar defining base structures. These became maximally general principles of what is called *X-bar theory* (Jackendoff, 1977), which impose specific internal relationships between phrasal components – regardless of what the "main" or *head* syntactic category of the phrase is. Roughly speaking, the underlying principles that determine how the ingredients of a noun phrase, verb phrase, and prepositional phrase interact are identical. Moreover, these principles hold true across different languages with small but sharply delineated distinctions. For example, in

[8] However, see Wexler and Culicover (1980) who establish a collection of constraints on transformations that guarantee successful learning on the basis of sentences with two levels of embedding (degree-2 learning).

English the head of a phrase (e.g. a verb or a preposition) comes before its argument (e.g. an object), whereas in Japanese, the head follows its argument.

This principle-based (as opposed to rule-based) theory of phrase structure and its transformation became known as *Government and Binding Theory*, or *GB Theory*. As will be shown below, this incorporated a new conception of how natural languages can differ from each other, and permitted an entirely novel conception of language acquisition. In order to establish a basis for the computational research presented in the thesis, only three aspects of GB syntax are relevant:

i. X-bar theory

ii. Movement (and other) transformations (and the notions of base structure and surface structure)

iii. Natural languages are assumed to share the same innate universal principles governing i and ii and to differ only with respect to their lexicons and the settings of a finite number of *parameters*.

This explication of human language is often called the *principles and parameters* (or P&P) hypothesis or framework. The principles and the parameters constitute *Universal Grammar* (UG): a lexicon and an array of settings (or values) of the parameters constitute a natural language grammar, which combines with UG to generate a particular natural language. Thus the parameters represent the points of variation

between languages. 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. Overt subjects are required in English, but not in Spanish. Consequently, the null subject parameter is set "off" for English and "on" for Spanish. Within X-bar theory, GB posits a parameter setting for the position of the head of a phrase in relation to its argument(s). As noted above, in English the head precedes the argument; in Japanese the argument precedes the head.

Language acquisition is the process of selecting the correct value of each such parameter for the language the learner is exposed to.[9]

The dissertation presents a method of analyzing computational learning models in the P&P framework. But before such models are discussed in more detail it is important to place the P&P framework clearly with respect to the nativist/empiricist debate discussed in section 1.1. Are the principles of human language innate or learned? P&P models (as most of generative linguistics) take the *innateness* of the principles as a basic

[9] The most recent work within transformational theory (Chomsky, 1995) has altered many details of derivations (giving up the notion of a significant level of deep structure, for example), but has retained the idea that languages differ only lexically or with respect to a relatively small number of syntactic factors that interact with general syntactic principles. Parameter values are now identified with specific syntactic features on 'functional' heads. However, the general notion of parametric variation is retained and the conception of language acquisition as parameter setting therefore also carries over from the P&P framework.

mandate.[10] Universal Grammar is universal *because* it is innate; it is an aspect of human biology. The only thing that is not innate, besides the lexicon, and which therefore needs to be learned, is the proper settings of the parameters that contribute to determining the surface strings of the language the learner is encountering. However, there are many acquisition models in which language structure is not innate, i.e., for which there is no UG at all. These warrant brief mention.

1.3 Non-generative Accounts of Acquisition

There has been much important recent work in the field of non-syntactically grounded language acquisition models. They can be grouped into three broad methodological categories: *connectionist*, *calculationist*[11] and *cognitivist*.

The connectionist approach employs a computational device that coarsely mimics behavior exhibited by biological neurons.[12] There are many fruitful variations of this

---

[10] An interesting alternative development within the generative tradition is Optimality Theory or OT (Prince and Smolensky, 1993), in which a grammar is a set of ranked constraints. The constraints are innate and the ranking is learned. Variation between languages is created by different rankings of the same constraints. Although OT has been successfully applied in phonology, and several interesting learnability results have been produced (Tesar and Smolensky, 1998), it is unclear whether either the explanatory power or the learnability results will translate smoothly to the domain of syntax. Active work in both areas is ongoing at this time; see, in particular, Grimshaw (1997) and Bresnan (1998) for OT-based syntactic analyses.

[11] A more traditional term would be *statistical* . However the term *calculationist* better indicates research that incorporates not only statistical methods, but also techniques from machine learning, genetic algorithms and other information theoretic paradigms.

[12] The most common implementations are a specific type of *artificial neural network*, called a simple recurrent network (SRN). However there are other devices with a strong connectionist affinity that do not specifically implement standard neural networks.

approach, which has been promulgated for syntax learning most notably by Jeff Elman (see Elman 1995, Elman et al. 1997 and references therein). The calculationist approach includes models of learning that make use of statistical, probabilistic or information theoretic techniques (cf. Charniak 1993, Brent 1996, Brill 1993, Clark 1992; in press, Jurafsky 1996, Pearlmutter et. al 1994, de Marcken 1996, Finch 1993, among others).[13]

Both connectionist and calculationist models are data-driven, rather than principle-driven, in the sense that they look for regularities within the words and strings of a corpus (constituting the input to the learning model) and on this basis assign a structure to the language being learned. Contrary to the innateness assumption, *all* structural information[14] must be discoverable in the data – there is no preexisting innate language mechanism that can be brought to bear during acquisition. It is worthwhile to point out that a common misconception of these paradigms is that they do not generate, learn or make use of *any* rules. This may or may not be the case. For example, Kremer, in his 1996 dissertation, investigates which types of formal language grammars (made up of rules) are learnable by canonically restricted neural networks.

---

[13] It should be noted that there is some overlap with generative linguistics. For example, Charniak is concerned with statistical approaches to PS grammars and Clark is concerned with information theoretic approaches to parameter setting.

[14] Some of the non-generative schools avoid the term "structure" altogether and view the learning process as designed to recognize distributional patterns rather than determine structure. However I forego addressing the interesting question of whether patterns constitute or at least imply structure and use the term structure to include either type of description.

The cognitivist approach, developed most notably by George Lakoff and Ronald Langacker, eschews the idea that language (and language acquisition) can be studied as an independent part of cognition. The belief is that language is a function of overall cognitive processing, inextricably related to other cognitive systems such as the visual system, motor system, emotions, etc. There is unfortunately little work on computational models of acquisition within this framework. However there exists some recent integration with connectionist approaches (cf. Feldman et al 1990, Reiger 1996).

The main disadvantage of these non-generative approaches is that, since they cast aside an existing base of linguistic theory, researchers in language learning are in the position of having to explain both *what* is learned as well as *how* it is learned. This may be one reason that there has been little progress towards achieving cross-linguistic learning results. For example, statistical context free grammars have been applied with some degree of success to approximate human ambiguity resolution preferences in English; however, a single system has not been developed that performs across several (much less many) languages. Likewise, the connectionists are able to demonstrate how one artificial neural network or another is capable of predicting parts of speech. But again, most research is circumscribed by a single language and in turn is usually restricted to a small subset thereof. Whether this is a constitutional limitation or can be overcome by future refinements of the approach remains to be seen.

The primary advantage of these approaches is that they dovetail with research in areas outside of linguistics but still within the realm of cognitive science. In recent years, connectionist research has surged forward and broadened in scope almost to the point of becoming the center of a unified philosophy of cognition. And many calculationist approaches to syntax acquisition borrow heavily from advances in statistical pattern recognition and machine learning that have been applied with significant success in the fields of speech recognition, information retrieval and visual recognition among others.

To summarize: the computational learning mechanisms of the non-generative models are well understood because they parallel existing methodologies, but it is difficult to ascertain the power or precision of their predictions without a cross-linguistic theory of language.

In contrast, the many detailed language descriptions given within the generative framework (especially within the P&P framework), provide a rich testing ground for acquisition theories. Of course, there is no guarantee that generative linguistics will continue to be successful, but at present the research is rich and active, and stable enough to make it worthwhile to explore in detail how the acquisition of P&P grammars may proceed.

1.4 The Principles and Parameters Framework

In the P&P framework, the syntactic component of a grammar 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.[15] These 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). If natural language requires 30, then there would be 2^{30} = 1,073,741,824 possible grammars, assuming there are no co-occurrence constraints limiting the combinations of values. Importantly, the number of distinct grammars (hence the number of languages) is exponential in the number of parameters.

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, even if the linguistic theory delineates over a billion possible grammars, a learner need only determine the correct 30 values that correspond to the grammar that generates the sentences of the target language. Indeed, in the extreme case, only 30 learning events need to occur – one that selects the proper value for each parameter.

---

[15] Issues arise when a language does not exhibit a parameterized feature that exists in another language. *"Irrelevant"* parameters control properties of phenomena not present in the target language, such as clitic order in a language without clitics. In the absence of clitics, any setting of a clitic-order parameter would generate the same language (set of surface structures) as any other. See further discussion below.

1.4.1 Learning in Parametric Spaces and the Parametric Principle

Learning in the P&P framework is simply the process of determining the parameter values that constitute the target grammar. The classic view[16] is that of a parameter as an *on/off* switch and parameter setting as flipping a switch. The idea is that an input sentence is processed by a learning device which can recognize in it some property that reveals either the *on* or the *off* setting of some parameter. The learning device then simply flips that parameter switch to the correct setting, if it is not already so set (Figure 1 below). However, this does not work for natural languages because natural language parameters reflect deep properties of a language that are not readily revealed in the surface word order of a sentence.

[16] The material in this section largely follows Janet Fodor's presidential address for the Linguistics Society of America in 1998.

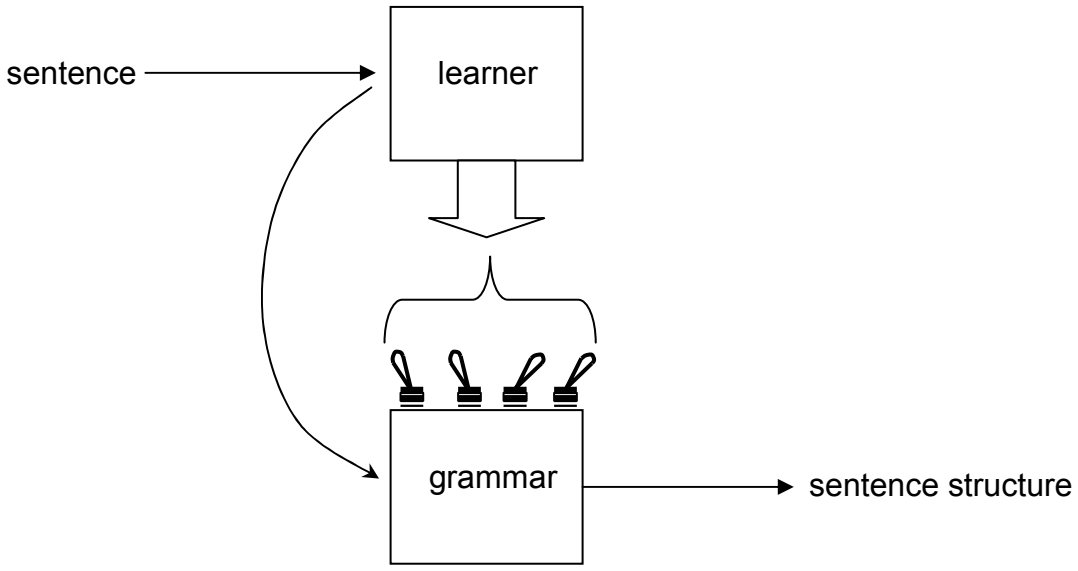


Figure 1. A simple parametric learning architecture. The learner receives the input sentence and consequently flips the appropriate parameter switches so that the sentence structure can be determined.

As a simple example, consider that English is verb-initial – the verb comes at the beginning of a verb phrase. Japanese, on the other hand, is verb-final. In accord with the switch-flipping metaphor, imagine a parameter switch called the "V-before-O" switch. If the switch is on, a main verb precedes its objects; if off, objects precede the main verb. (This is one instantiation of the X-bar theoretic head/argument direction parameter from Section 1.2 above.) Assuming that the target grammar is English, the learner will receive input sentences such as *Give the ball back!* and *I will eat the cake with you.* Both of these appear to be evidence that the parameter should be flipped to the on position. However the learner will also encounter sentences like: *This ball, we never take!* and *What games does Jaime play?* These are clearly grammatical English sentences.

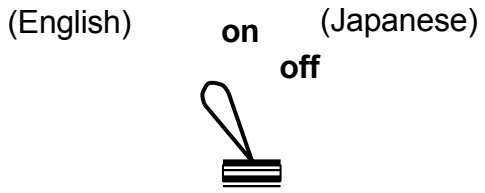


Figure 2. English setting of the V-before-O parameter

Just as clearly, the V-before-O parameter shouldn't be set to the off position, even though in these sentences the main verb comes *after* the object. The reason is that the underlying order (which is relevant to the parameter) can be altered by movement operations, so that the correct value for the parameter is not apparent from the surface word order. Only after determining which transformational operations have been applied can the switch be flipped to its correct position. But in order to determine the correct derivation of Verb-Object position, one would need to know the settings of the parameters that govern the position of the subject in relation to the verb and object. But the learner may not have learned those settings before attempting to set the V-before-O parameter.

There is a paradox here as Valian (1990) has pointed out. To set parameters, one needs to know the structure of the sentences one hears. In order to determine the structure of a sentence, one needs to know the grammar that generated it. But the grammar is not known – it is what the learner is trying to learn. One way around this is for the learner to hypothesize that the parameters that govern subject-verb order are set in

some way and then try out or test a setting of the V-before-O parameter. If the test is unsuccessful the learner would reconsider and attempt a different setting of the subject-verb parameters in order to test a new hypothesis about the setting of the V-before-O parameter.

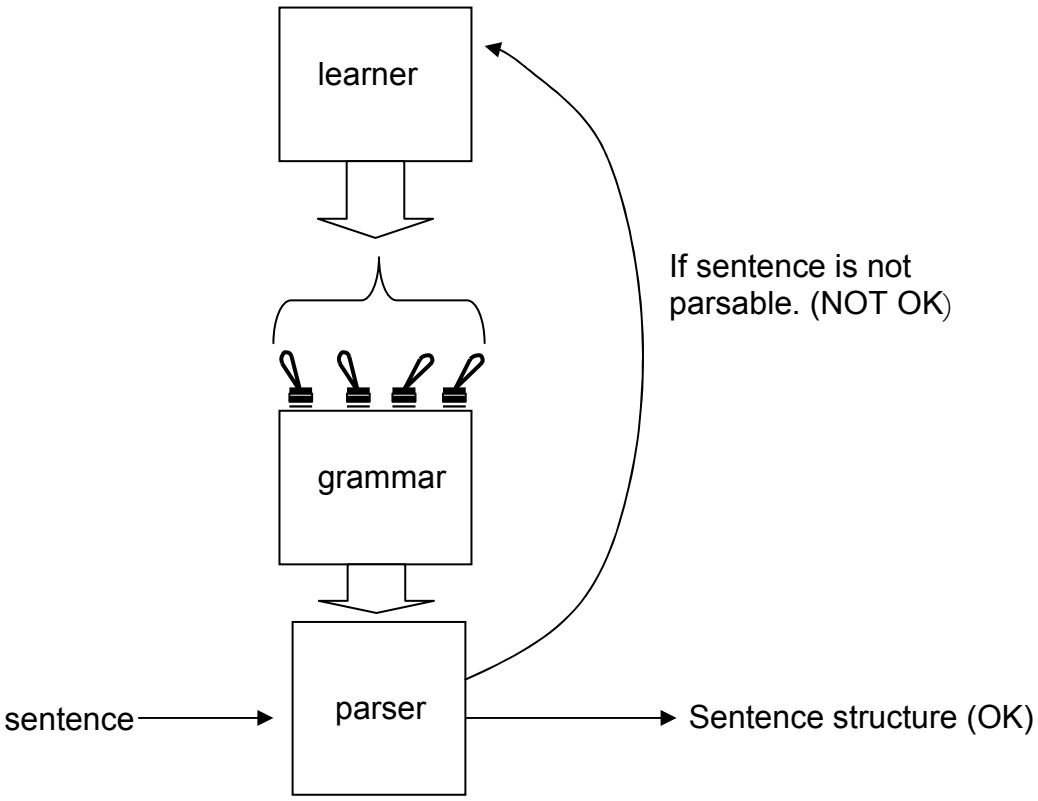


Figure 3. Hypothesize-and-test architecture. A grammar is in place. The parser uses that grammar to process a sentence. This yields an OK/NOT OK result. If the parse test is OK then the learner outputs the structure and retains that grammar. Otherwise the learner hypothesizes other parameter settings in an attempt to get a successful parse.

It can formally be shown that there exist necessary and sufficient conditions that guarantee that this hypothesize-and-test strategy will eventually converge, i.e. that all of the parameters will eventually be set correctly for the target language (cf. Bertolo in press for an application of Angluin's 1980 context-free learnability theorem[17] in a proof establishing that parametric systems of this sort are learnable). Although these conditions allow for a large class of formal languages, it remains to be shown that the class of human languages can be counted among them.

Furthermore the model faces a potentially enormous computational workload. In order for the learner to finally hypothesize the correct grammar, repeated parsing needs to be executed. Suppose, as in the previous section, that there are 30 independent, binary parameters. The total number of grammars is 2^{30} (over one billion). The worst case is where a sentence requires that all 30 parameters be set correctly and where the current hypothesis is exactly the opposite of the correct grammar for the target language: each and every switch needs to be flipped to its opposite position. In this case, the sentence might have to be parsed over a billion times – once by each possible combination of settings. (The exact number of parses would depend on the precise strategy the learner uses to choose which parameter to test next. It could be even higher if the grammars are unordered and/or no record is kept of grammars that have been tried and failed.)

[17] Although Gold is not normally credited for this parametric learning model, this was almost exactly his original paradigm. The difference is in the way that grammars are chosen to test. In his (and Angluin's) work whole grammars are enumerated without the use of parameters.

But now, the hypothesis-and-test learner is essentially entertaining whole grammars (using the parameters simply as a method of coding or enumerating the grammars) and circumventing the cardinal benefit of the P&P framework: efficiency. By evaluating whole grammars and not setting individual parameters, the learner is violating what we have called the *Parametric Principle: set individual parameters; do not evaluate whole grammars* (Fodor, 1998; Sakas and Fodor, in press).

In this crude form, the hypothesis-and-test learning procedure has not been advocated as a model of human language acquisition. The thesis will present analyses of two existing models of acquisition which are more sophisticated. One, Gibson and Wexler's *Triggering Learning Algorithm* (*TLA*) (1994), attempts to improve on the hypothesis-and-test model by adding attractively simple heuristics to the learner's strategy for selecting which parameters to test and when to test them. The other, Fodor's *Structural Triggers Learner* (*STL*) (1995, 1998), is a parametrically principled learner whose strategy for parameter setting is to avoid guessing in the face of ambiguity. It sets no parameters unless the input sentence gives perfect (unambiguous) evidence for the correct settings.

The basic result obtained is that parameter setting can be very hard work for either model. Although there are situations under which either model can feasibly acquire the grammar of an arbitrary target language, I will show that constraints on the input must be adhered to for successful learning in both cases.

1.5 The Methodology and Scope of the Thesis

Several broad ideas will serve as the foundation for the thesis:

- Generative linguistics in the principle and parameters framework is an active and fruitful area of research.

- Results from formal modeling and computational simulation can point to novel questions in linguistics and psycholinguistics and promote research that will help to refine current theories.

- The circumstances leading to the success or failure of a proposed model of language acquisition is insufficient as a measure of a model's goodness. The computational workload of the learner during the acquisition process must also be examined. That is, for a psychologically realistic model (as noted by Pinker 1979), feasibility is as important as learnability per se.

- A major factor affecting feasibility is the degree of parametric (i.e. cross-linguistic) ambiguity inherent in the language domain.

1.5.1 Goals

The primary goal will be to construct a stochastic framework for modeling language acquisition that will be general enough to apply to substantially different learning strategies. Though a stochastic characterization of parameter setting exists (see

discussion in Section 2.2.1 below for an overview of Niyogi and Berwick's framework), the work presented here extends it by focussing on the following characteristics which are not present in the existing formulation:

- A characterization of parametric ambiguity that can be used to compare and contrast various learning models by isolating the strengths and weaknesses of their core strategies.

- An abstraction of the language space in terms of parametric ambiguity which can be manipulated to mirror existing linguistic notions, so that different learning scenarios can be evaluated.

These abstractions are compatible with authentic linguistic analyses of the natural language domain, but nevertheless allow an evaluation of learning procedures that is not tied to the details of any one variety of linguistic description within the P&P framework.

Specifically, two important questions will be addressed:

- Are the internal mechanics or heuristics that are embodied in the learning architecture beneficial? Are they efficient or wasteful? Do they make feasible learning possible?

- What are the bounds on the shape of the language space (in terms of the distribution of ambiguity) that are required for feasible parametric learning?

In short, the thesis attempts to answer the question: Given a parameter setting procedure *P*, under what conditions is *P* a feasible learner for natural languages?

1.5.2 Contributions

The main contributions of the thesis are the following:

- A methodology for determining the feasibility of parameter-setting models of acquisition that incorporates an abstraction of language centered on the rate of cross-linguistic (parametric) ambiguity.

- Results from the examination of two influential parameter-setting learning procedures that establish bounds on the linguistic environment (with respect to parametric ambiguity) that must be in place in order for either to be a feasible model of language learning.

The study of language learnability has a long and fruitful history. The thesis will not argue for or against a particular paradigm in terms of learnability. Instead its focus is on the efficiency or feasibility of a model. However, the work takes to heart the elegant abstraction of traditional learnability research in order to put forth a new way of addressing the simple question: "Is model *P* a plausible model of human acquisition?"

In much the same way as learnability proofs in the tradition of Gold rely on the computational abstraction of linguistic theory, so does the framework presented in the dissertation. By formulating ambiguity so that computational methods can be employed to study learning performance, the thesis provides a valuable way to delineate bounds on the distribution of ambiguity that must respected in order for a learner to succeed.

2 Formalizing the Acquisition Process

2.1 Definitions

In this chapter, an initial formalization of parameter setting is presented. Initially, I take the view of syntax acquisition as state space search where each state represents a grammar and the learning algorithm entertains a single grammar at a time.[18] As the learner receives input sentences, it decides whether to retain its currently hypothesized grammar or to 'shift' to another grammar in the space. Later in the thesis, an alternative framework is presented where parameter values, rather than grammars, underpin the states of the system.

The following definitions will make concrete several of the notions discussed up to this point as well as being useful for the discussions that follow.

- A *grammar*, denoted as G_i, G_j,…., is a vector of parameter values.
- A *parameter* , denoted as p_i, p_j,…., is an index of an element in a grammar vector.
- A *parameter value* is either 0 or 1.
- The *target grammar* G_{targ} is the grammar that is to be acquired by the learner.

[18] See Clark, 1992; Nyberg, 1994 and Yang, 1999 for models which entertain more than one grammar at a time.

- A *parameter space,* denoted as $\mathcal{H}_{n.}$,[19] is the set of all possible grammar vectors of size *n*, with one grammar marked as the target grammar G_{targ}. The learner's currently hypothesized grammar at any point in the learning process is denoted as G_{curr}.
- $G_{curr} \rightarrow G_i$ denotes a change of hypothesized grammar, made by the learner, from grammar G_{curr} to G_i .

Figure 4 presents an example of a possible state space for parameter space $\mathcal{H}_{n,}$

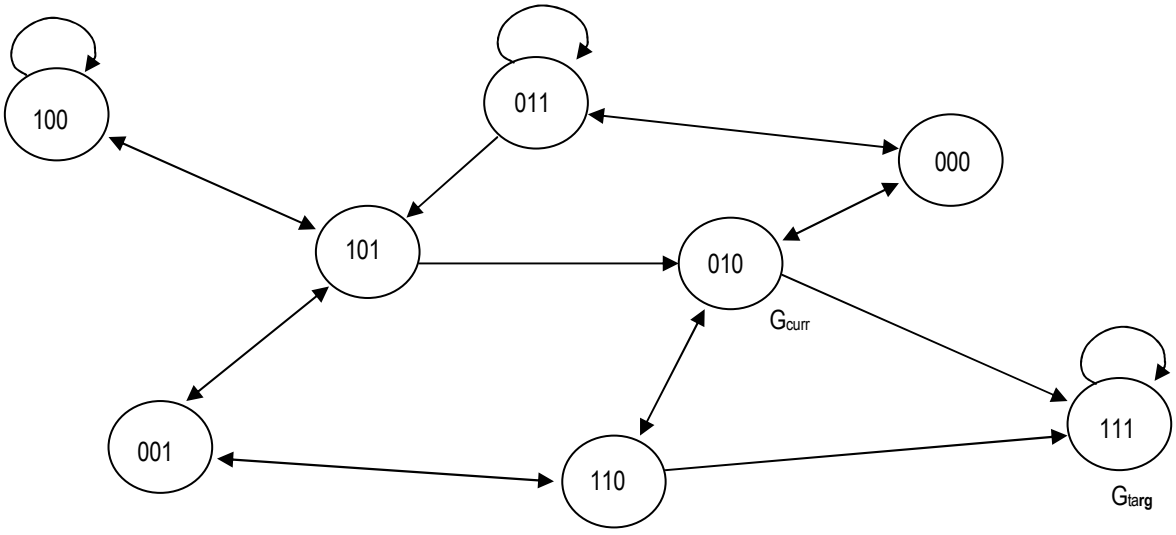


Figure 4. A possible state space for parameter space $\mathcal{H}_3$ Nodes represent grammars and arcs represent a possible change from one hypothesized grammar to another; the arcs are compelled by specifics of the learning algorithm and the input it receives. The target grammar to be achieved is G_{targ} = 111 and the learner is currently entertaining G_{curr} = 010.

[19] Technically $\mathcal{H}$ should have two subscripts – one for the number of parameters and a second indicating the target grammar. For clarity, the second is omitted since it will be clear from the context which grammar is being considered as the target.

Niyogi and Berwick (1996) point out that the process of acquiring parameter settings can be formalized as a Markov structure with probabilities assigned to the arcs in the state space diagram. Moreover, feasibility results can be obtained by applying standard Markov theory. This important insight serves as the springboard to the work that is presented in the thesis and hence warrants some discussion in the next section. But first, some more definitions need to be in place.

- A *input* or an *input sentence,* denoted *s*, *t*,…., is a string generated by the target grammar and consumed by the learner. Unless otherwise specified, an input is *arbitrarily* drawn from $L(G_{targ})$.

- A *trigger* is an input that causes the learner to change or "flip" one or more parameter values. Equivalently, a trigger is an input that causes the learner to shift to a new grammar hypothesis. The resulting parameter value or grammar need not be correct.[20]

- The *text* or *input sample* is an ordered random selection of strings from the target language $L(G_{targ})$. That is, it is simply a stream of sentences (repeats allowed) generated by the target grammar.

[20] This definition differs from other notions of triggers. See, in particular Gibson and Wexler (1994) and Frank and Kapur (1996) for extended discussion on useful formulations of the notion of a trigger.

- To *parse* an input, given grammar G_i, means to make the determination that a sentence *s* belongs to the language generated by G_i, i.e. to determine that $s \in L(G_i)$.

- A sentence *s*, is *ambiguous* between grammars G_i, G_j, G_k …. if *s* can be parsed by G_i, G_j, G_k . . . In such cases, $s \in L(G_i) \cap L(G_j) \cap L(G_k)$.

2.2 Parametric Learning as a Markov Process

A *Markov process* (or system or chain) is a stochastic process in which the resulting behavior of the process at time *t* is entirely determined by the state of the process at time $t - 1$; future behavior is not predicted by past behavior.[21] Markov processes accurately model systems that have no memory. A system in state *S* always behaves in exactly the same way, regardless of how long it has been in state *S* or what other states it has passed through (or not passed through) on the way to *S*.

As a simple example consider the transition diagram below.

[21] In many formulations, including the one presented in this dissertation, time is measured in terms of number of inputs. E.g. $t = 20$ means 20 inputs have been consumed by the system.

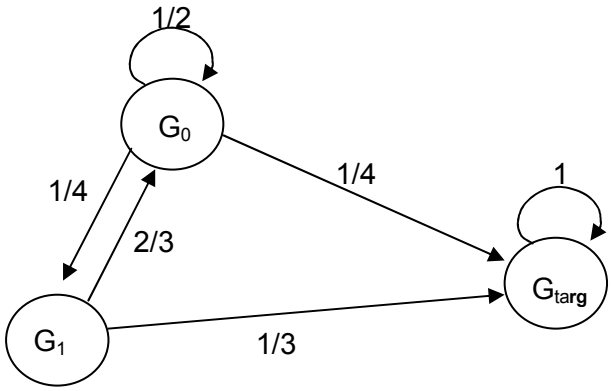


Figure 5. A Markov transition diagram.

Imagine that the learner is in the state of entertaining grammar G_0 and receives an arbitrary input sentence (generated by the target grammar, G_{targ}). The diagram indicates that there is a probability of ½ that the learner will remain in state G_0 and a ¼ chance that it will shift to entertain either the target grammar or grammar G_1. Likewise if the learner is entertaining G_1, there is a 1/3 chance that the target will be acquired after an input, a 2/3 chance the learner will shift to state G_0, and zero probability of the system remaining in state G_1.

Several points should be made about this formulation. First, if the learner is in state G_{targ}, the probability of remaining in G_{targ} is equal to 1; no other transitions are possible. This would be the case if the input sample is generated by the target grammar and the learner is error-driven (see Section 3.2 below) in that it will not entertain a new grammar hypothesis if the current grammar can parse the input. In Markov terms,

G_{targ} is called an *absorbing state* and a system that contains at least one absorbing state is called an *absorbing system*. All the non-absorbing states are referred to as *transient* states.[22]

Second, the sum of the probabilities of all arcs leaving any given state must equal 1. Although the actual probabilities are determined by the specifics of the learning algorithm and the input sample, all possible outcomes must be accounted for. Finally, the diagram can be converted to a transition matrix where the left column indicates the current state of the system and the top row indicates the resulting state after an input is consumed, with the transition probabilities making up the heart of the matrix. This is illustrated in Figure 6 – transitions between transient, non-absorbing states are shaded.

	G_0	G_1	G_{targ}
G_0	1/2	1/4	1/4
G_1	2/3	0	1/3
G_{targ}	0	0	1

Figure 6. A transition Matrix derived from the chain depicted in Figure 5 above. The submatrix *N* that gives the transition probabilities of the *transient* (non-absorbing) states is shaded.

There is a large arsenal of existing Markov techniques that can applied to Markov processes in general and to absorbing systems in particular. I will make use of a relatively straightforward technique that is used to determine the average number of

[22] Two technical notes. For most of the mathematics of absorbing systems to apply, the absorbing state must be reachable from all transient states. Also, a system that contains a *class* of states (more than one) from which there is no 'escape' can also be treated as an absorbing system even if there is no single absorbing state.

inputs that the system can be expected to consume before entering the absorbing state (G_{targ}, in this example).

The *fundamental matrix Q* of a transition matrix of an absorbing system is defined as the inverse of the identity matrix *I* (= 1 on the diagonal, 0 elsewhere) minus the (sub)matrix *N* that gives the transition probabilities of the *non*-absorbing or *transient* states (see Figure 6),. that is:

Equation 1 $$Q = (I - N)'$$

From the example above, *Q* can be established as follows:

$$Q = \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 1/2 & 1/4 \\ 2/3 & 0 \end{bmatrix} \right)'$$

$$Q = \begin{bmatrix} 3 & .75 \\ 2 & 1.5 \end{bmatrix}$$

The row sum of the fundamental matrix *Q* gives the expected number of inputs to absorption starting from the corresponding state in the original transition matrix. So, starting from state G_0, the number of inputs the system will consume, on average, before entering state G_{targ} (the absorbing state) is 3 + .75 = 3.75, and starting from state G_1 the average number of inputs is: 2 + 1.5 = 3.5.

Clearly, the formulation of acquisition as a Markov structure readily allows for the derivation of feasibility results in terms of the number of inputs consumed during the acquisition process.

The basic procedure I will follow is:

- Determine and define linguistically relevant factors that will determine the shape of the learner's environment;
- Determine the transition probabilities based on the internal mechanisms of a given learning algorithm and how it will respond to the environment;
- Calculate the fundamental matrix, which will yield the expected number of sentences required for the learner to converge on the target.

2.2.1 Niyogi and Berwick's Approach

Niyogi and Berwick (1996 and elsewhere) point out that many psycholinguistic accounts of learnability make the assumption that the human learning mechanism is memoryless and thus can be modeled as a Markov process. They cleverly exploit Markov theory in order to attain both learnability and feasibility results. This is important because it represents the first application of Markov chain theory to a parameter-setting model of natural language syntax. Niyogi and Berwick also uncover a subtle aspect of stochastic learning algorithms that affects their ultimate success or

failure, and that had previously been missed in the learnability literature. Specifically, they show that success is not solely dependent on whether a series of triggers exists in the environment that can lead the learner to the target. Rather an algorithm might, with some degree of probability, wander off the "correct" path into a state from which no triggers exists that will guide the learner to the target.

However, their discussion leaves several important questions unanswered. Although their learnability results are general in that they give necessary and sufficient conditions for a target grammar to be unattainable, the feasibility results they present are tied directly to the idiosyncrasies of the parameter domain under investigation. This is because the method they employ to compute the transition probabilities relies on the intersections of neighboring languages in the parameter space and therefore is bound to the particular composition of the languages in the space. Niyogi and Berwick point to the fact that their formulation can be used to "falsify" or "validate" certain kinds of parametric theories on general grounds of feasibility, but leave open the question of how to do so.

They also admit that in order to apply their methodology to a parameter space of reasonable size, one would be required to manipulate extremely large matrices. For 30 parameters, the transition matrix would need to be $2^{30} \times 2^{30}$ elements. Any calculations involving a matrix of this size would be computationally intractable. This leaves open a second crucial question of how the size of the input sample required for successful

learning scales up as the number of parameters increases. This is because the dimension of the matrix increases exponentially with respect to the number of parameters in the domain. Five parameters would require a matrix of $2^5 \times 2^5 = 32 \times 32$ elements, whereas 30 parameters requires $2^{30} \times 2^{30} = 1{,}073{,}741{,}824 \times 1{,}073{,}741{,}824$ elements.

This dissertation builds on Niyogi and Berwick's work by presenting a tractable probabilistic framework that can distinguish among different sources of learning difficulty and their effect on the acquisition process, as the size of the parametric domain expands.

The thrust will be to abstract away from the linguistic particulars of the language space (such as the exact sentences that make up each language) by isolating factors that will capture broader linguistic notions and by incorporating these abstractions into a Markov model. One such notion is that of *smoothness* which I discuss below.

2.3 Distance from the Target and Linguistic Smoothness

In order to get a handle on performance as the size of the search space increases, I consider classes of grammars that share a property or properties that allow the learning process to be tractably modeled. One useful metric is to characterize a class of grammars by their distance from the target grammar.

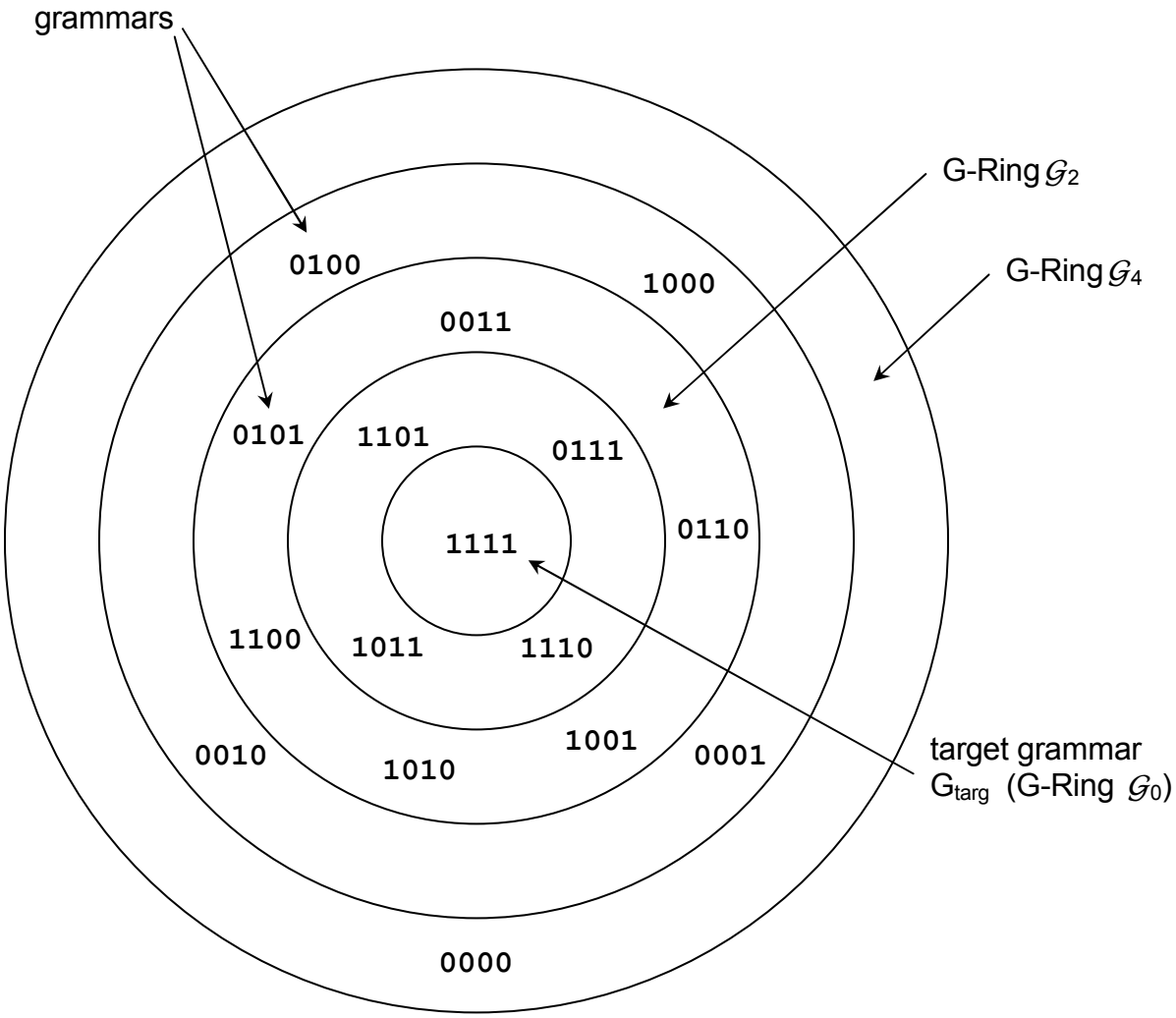


Figure 7. Parameter space $\mathcal{H}_4$ with G_{targ} = 1111. Each *G-Ring* contains exactly those grammars a certain hamming distance from the target. For example, ring $\mathcal{G}_2$ contains 0011, 0101, 1100, 1010, 1001 and 0110 all of which differ from G_{targ} by 2 bits.

Definition:

- A ***G-Ring*** ($\mathcal{G}_i$) is a set of grammars that share the same hamming distance h (the number of parameter values that differ) from the target grammar: $\mathcal{G}_i = \{ G \in \mathcal{H}_n \mid h(G_{targ}, G) = i \}$

Unless otherwise specified, I will assume in what follows that the target grammar G_{targ} is the vector consisting of parameter values that are all equal to 1. There is no generality lost by this convenient assumption. It is easy to define an isomorphism that maps the parameter space under this assumption to another parameter space with an arbitrarily chosen target grammar.

Using this approach, the learning process can be viewed as the learner moving (or not) from a grammar in one G-Ring to a grammar in another G-Ring, rather than from one grammar to another grammar.

This helps to make concrete the linguistic notion of *smoothness*. Basically, smoothness means that there is a correlation between the similarity of grammars and the similarity of the languages that they generate. Two requirements on the parametric space will be considered in the analysis in the chapters that follows. They are:

- ***Weak Smoothness Requirement*** – All the members of a G-Ring can parse s with an equal probability.

- ***Strong Smoothness Requirement*** – The parameter space is weakly smooth and the probability that *s* can be parsed by a member of a G-Ring increases monotonically as the index (i.e. distance from the target) of the G-Ring decreases. Basically, the closer the G-Ring to the target grammar, the higher the probability that a grammar in the ring can parse *s*.

By establishing smoothness requirements using G-Rings, I will tractably model the learning process using the Markov techniques described above. This is possible because the transition matrix need only be of a dimension equal to the number of parameters plus one (= number of G-Rings) rather than a dimension exponential in the number of parameters (2^n, n = number of parameters). Of course, some linguistic accuracy is lost. It is almost certainly true that natural languages are not weakly smooth. A language may be vastly different than other languages that are similarly distant from the target. Chomsky (1988) notes that a change of even one parameter value may make a considerable difference to the surface sentences of the language. Still, modeling smoothness is a worthwhile endeavor. Results presented in the next chapter strongly indicate that smoothness plays a large role in terms of acquisition feasibility.

3 The Triggering Learning Algorithm

In 1994, Gibson and Wexler (henceforth G&W) in a seminal article presented the Triggering Learning Algorithm (TLA), a procedure for learning the settings of linguistic parameters. See Table 1 for a description of the algorithm. One significant aspect of this research is that the TLA embodies many important concepts that had existed in the acquisition literature in different guises (see particularly Clark 1992). Also of importance is the language space that G&W construct to test the TLA. The space is based on three linguistically authentic parameters, and G&W use it to show under what circumstances a language domain is learnable by the TLA. Given that the TLA model embodies some psychologically and linguistically desirable features, and that finite parameter spaces are presupposed to be easy to learn, a notable and somewhat surprising result is that the TLA sometimes fails to acquire the target because it gets trapped in local maxima within this very simple parametric framework.

Repeat until convergence ($G_{curr} = G_{targ}$):

1) Receive a string s from $L(G_{targ})$.
2) If s can be parsed by $G_{curr,}$ do nothing (goto 1).
   Otherwise change a single randomly chosen
   parameter to its opposite value.
   This yields a new grammar (G_{new}).
3) If G_{new} can parse s, make G_{new} the current
   grammar ($G_{curr} \leftarrow G_{new}$).
   Otherwise, don't change G_{curr}.

Table 1. The Triggering Learning Algorithm (informal description).

In what follows, I first review the space of grammars and languages that G&W construct and then present a more detailed characterization of the algorithm and finally a probabilistic analysis of its performance.

3.1 The Language Space

G&W develop a language domain of eight languages defined by three parameters that control word order variation. The three parameters are:

- **Specifier-Head Parameter (SV/VS)** – determines whether the specifier is at the beginning or end of a phrase. In practice, this parameter determines solely whether the subject precedes the verb phrase or the verb phrase precedes the subject. In the given language domain, the only specifiers are the specifiers of CP and IP, and the position of the former is fixed as initial.

- **Complement-Head (VO/OV)** – determines whether the complement comes at the beginning or end of a phrase. In this domain, it determines two linked properties: whether the main verb precedes or follows the object, and whether an auxiliary verb precedes or follows the VP.

- **Verb-Second (+V2/-V2)** – determines whether or not the finite verb (= the auxiliary verb if one is present; else the main verb) is moved to become the head of the CP projection, with some XP (subject, object, indirect object or adverb) moved to its specifier position (which is always pre-head).

In short: The SV/VS and VO/OV parameters dictate base structure word order, and +V2/-V2 dictates whether a pair of movement transformations apply to the base structure to yield the surface string (an example is offered in (4) – (6) below). These three parameters define eight grammars as listed in Table 2 which generate the languages listed in Table 3. Note that there are no rules or parameters for embedding a clause within a clause, and no recursive operations at all, so the languages generated are finite.

Grammar	Spec-Head	Comp-Head	Verb-Second
1	SV	OV	-V2
2	SV	OV	+V2
3	SV	VO	-V2
4	SV	VO	+V2
5	VS	OV	-V2
6	VS	OV	+V2
7	VS	VO	-V2
8	VS	VO	+V2

Table 2. The eight grammars that make up the G&W parameter space.

Input sentences are assumed to be tagged with role assignments characterizing the constituent words or phrases. The tokens and respective roles they represent are: S (subject), O or O1 (direct object), O2 (indirect object), V (verb), aux (auxiliary verb) and adv (adverb). (The adverb in always sentence-initial.) Structure inside such elements (e.g., internal to a subject noun phrase) is not analyzed.

For example, the English sentence:

(1) *Frequently Becky will give the dog candy.*

comes to the learner after pre-analysis as:

(2) *adv S aux V O1 O2*

which belongs to language 3 in Table 3 below.

A few other simple examples serve to give a flavor of the languages in the domain and how they are generated.[23] Consider the string:

(3) *S V O*

This simple input is licensed by the grammars that generate languages 2, 3, 4, 6 and 8. The derivation is straightforward in the –V2 language (Language 3). However, the derivations generated by the +V2 grammars involve movement transformations. For

[23] From this point on, all examples will be in analyzed format, e.g., *aux* rather than *will* or *must*.

language 2 (SV OV +V2), the base generated string is *S O V* (which follows from the SV and OV parameter settings). The +V2 setting mandates two movements: V must move to the second position in the surface structure, and some other element – in this case *S* – moves to the first position.[24] This pair of operations, known as V2 movement, has the following consequences for the surface string. (4) is the terminal string of the base-generated structure. (5) corresponds to an intermediate structure and (6) depicts the final surface string.

(4) S O V

(5) S t_S O V

(6) S V t_S O t_V

[24] In the example that follows, I assume XP movement precedes V movement. Since the TLA makes use of an 'all or nothing' parse test, the details of the derivation are inconsequential to learning performance.

Language 1: sv ov -v2

s v
s o v
s o2 o1 v
s v aux
s o v aux
s o2 o1 v aux
adv s v
adv s o v
adv s o2 o1 v
adv s v aux
adv s o v aux
adv s o2 o1 v aux

Language 2: sv ov +v2

s v
s v o
o v s
s v o2 o1
o1 v s o2
o2 v s o1
s aux v
s aux o v
o aux s v
s aux o2 o1 v
o1 aux s o2 v
o2 aux s o1 v
adv v s
adv v s o
adv v s o2 o1
adv aux s v
adv aux s o v
adv aux s o2 o1 v

Language 3: sv vo -v2 (English-like)

s v
s v o
s v o1 o2
s aux v
s aux v o
s aux v o1 o2
adv s v
adv s v o
adv s v o1 o2
adv s aux v
adv s aux v o
adv s aux v o1 o2

Language 4: sv vo +v2

s v
s v o
o v s
s v o1 o2
o1 v s o2
o2 v s o1
s aux v
s aux v o
o aux s v
s aux v o1 o2
o1 aux s v o2
o2 aux s v o1
adv v s
adv v s o
adv v s o1 o2
adv aux s v
adv aux s v o
adv aux s v o1 o2

Language 5: vs ov -v2

v s
o v s
o2 o1 v s
v aux s
o v aux s
o2 o1 v aux s
adv v s
adv o v s
adv o2 o1 v s
adv v aux s
adv o v aux s
adv o2 o1 v aux s

Language 6: vs ov +v2

s v
o v s
s v o
s v o2 o1
o1 v o2 s
o2 v o1 s
s aux v
s aux o v
o aux v s
s aux o2 o1 v
o1 aux o2 v s
o2 aux o1 v s
adv v s
adv v o s
adv v o2 o1 s
adv aux v s
adv aux o v s
adv aux o2 o1 v s

Language 7: vs vo -v2

v s
v o s
v o1 o2 s
aux v s
aux v o s
aux v o1 o2 s
adv v s
adv v o s
adv v o1 o2 s
adv aux v s
adv aux v o s
adv aux v o1 o2 s

Language 8: vs vo +v2

s v
s v o
o v s
s v o1 o2
o1 v o2 s
o2 v o1 s
s aux v
s aux v o
o aux v s
s aux v o1 o2
o1 aux v o2 s
o2 aux v o1 s
adv v s
adv v o s
adv v o1 o2 s
adv aux v s
adv aux v o s
adv aux v o1 o2 s

Table 3. The sentences of the eight languages that are generated by the grammars in G&W's parameter space.

Comparable derivations can be worked through by means of which S V O can be arrived at from the base structures dictated by the parameter settings for languages 4, 6 and 8. (Assuming that movement transformations leave a trace in the pre-movement position, the position of the traces of S and V will differ from language to language, but these are phonologically null and hence are assumed not to be accessible to the learner.)

The linguistic details – e.g. whether the component movements of V2 are ordered, whether traces are actually left behind or are deleted, etc. – are not central to the TLA performance. The important point is that the learner only sees S V O; the base structure and the derivation are not part of the input. The grammar determines the derivation but the grammar is what the learner is attempting to ascertain. This is a restatement of the parsing paradox (see Section 1.4 above): To determine the derivation of a sentence, in order to learn the grammar from it, one needs to know the grammar that generated it.

3.2 The Algorithm

Three important constraints guide TLA learning. They are:

1) **Error Driven Constraint** – the learner will retain the current grammar unless the current input sentence can not be parsed by it.

2) **Single Value Constraint (SVC)** – the learner can test a new hypothesis (G_{new} above) only if it differs by no more than one parameter value from the current grammar (G_{curr}).
3) **Greediness**[25] – The learner can adopt G_{new} only if G_{new} can parse the current input.

The TLA neatly sidesteps the parsing paradox by utilizing a *parse test* as a method for experimenting with alternative grammars. On receiving an input string the TLA first tries to parse it with its current grammar G_{curr}. If this succeeds, no learning is called for. Though G_{curr} may not be correct, the learner at least has no specific reason to believe that it is wrong. If the parsing attempt with G_{curr} fails, the learner tries again with a modified grammar G_{new} that it arrives at by resetting one parameter, chosen at random. If the parsing attempt with G_{new} also fails, G_{new} is no improvement over G_{curr}, so the TLA retains G_{curr}., in accordance with the Greediness constraint. If G_{new} does permit a successful parse of *s*, the TLA shifts from G_{curr} to G_{new}. Although G_{new} is not necessarily the target grammar, it at least has the merit of being compatible with the current sentence. This is a necessary condition, though far from a sufficient one, for being the target grammar. The algorithm is summarized in Table 4.

[25] Frank and Kapur (1996) correctly point out that "Greediness" is a bit of a misnomer for this constraint – at least if the name is intended to parallel the traditional 'greedy' algorithm of computer science. Strictly speaking, a truly greedy TLA would consider *several* grammars (typically the number is constrained by a locality heuristic) and pick the best (as determined by an objective function) to adopt as its new hypothesis. Since the TLA randomly picks a new grammar to test, it is not greedy in this sense.

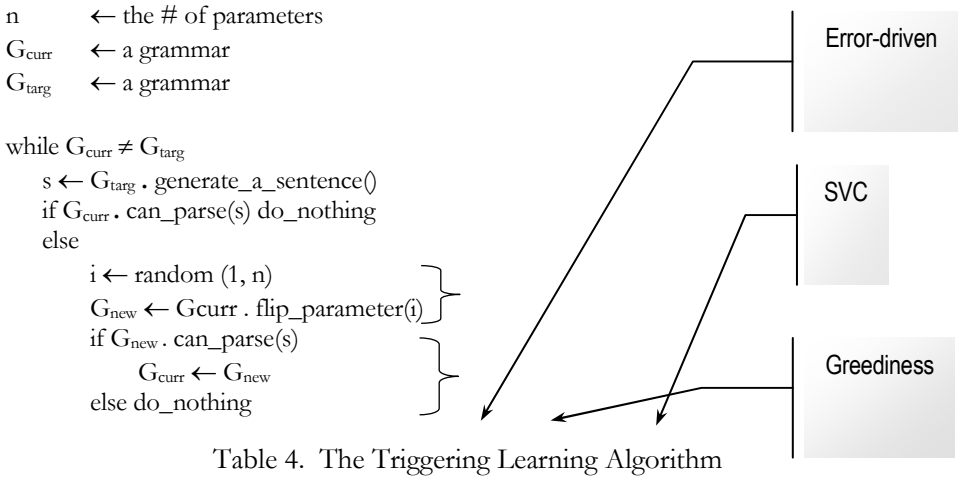


Table 4. The Triggering Learning Algorithm

The three constraints are intended to promote local hill-climbing in the state space of grammars. After each application of the parse test, Greediness and the SVC are meant to work in tandem to propel the learner's current hypothesis closer to the target grammar as learning proceeds. The idea is that Greediness provides a simple heuristic to evaluate grammatical 'betterness' and the SVC keeps the learner from wandering too far astray in the grammar space, losing gains it made in prior learning trials. The point at which each constraint comes into play is shown in Table 4.

Berwick and Niyogi (1996) and Berwick and Niyogi (1996), by running a simulation, demonstrate that these constraints do not afford the TLA any reliable advantage over an algorithm that picks grammars at random at least for the small three-parameter domain developed by G&W. From Berwick and Niyogi (1996):

> "The curve with the slowest rate … represents the TLA. The curve with the fastest rate … represents a random step algorithm with no Greediness Constraint or Single Value Constraint. Rates for Random Step with either the Greediness Constraint or the Single Value Constraint lie between these two and are very close to each other. The curves for cases where the target is some [other] language … are similar, though not depicted here; Random Step generally dominates the TLA."

They also give anecdotal examples where the constraints clearly steer the learner away from the target based on idiosyncrasies present in the language space being searched. In addition, I have demonstrated that although Greediness by itself (without the SVC) keeps the learner in the vicinity of the target language, it does so at the cost of requiring the learner to parse an inordinately large number of sentences before making a shift to a new grammar closer to the target (Sakas and Fodor, 1997).

What has remained unanswered is the question: how do the SVC and Greediness interact in general, especially in domains with more than three parameters? The sections that follow address this question.

3.3 A Probabilistic Formulation of the TLA

Figure 8, below, depicts the local state space of a TLA learner. The actions taken by the TLA are represented by the arcs and are based on the outcome of the parse test applied to a sentence from the target grammar. Following Niyogi and Berwick (1996),

a Markov structure can be created from the state space by determining the probability that any of the arcs is traversed. The structure can then be used to calculate the learning rate of acquisition. The approach presented here differs from Niyogi and Berwick's in that the probabilities are formulated incorporating broad factors that indicate the general shape of the domain as opposed to characteristics of a specific domain under study. (See Chapter 2 for discussion.)

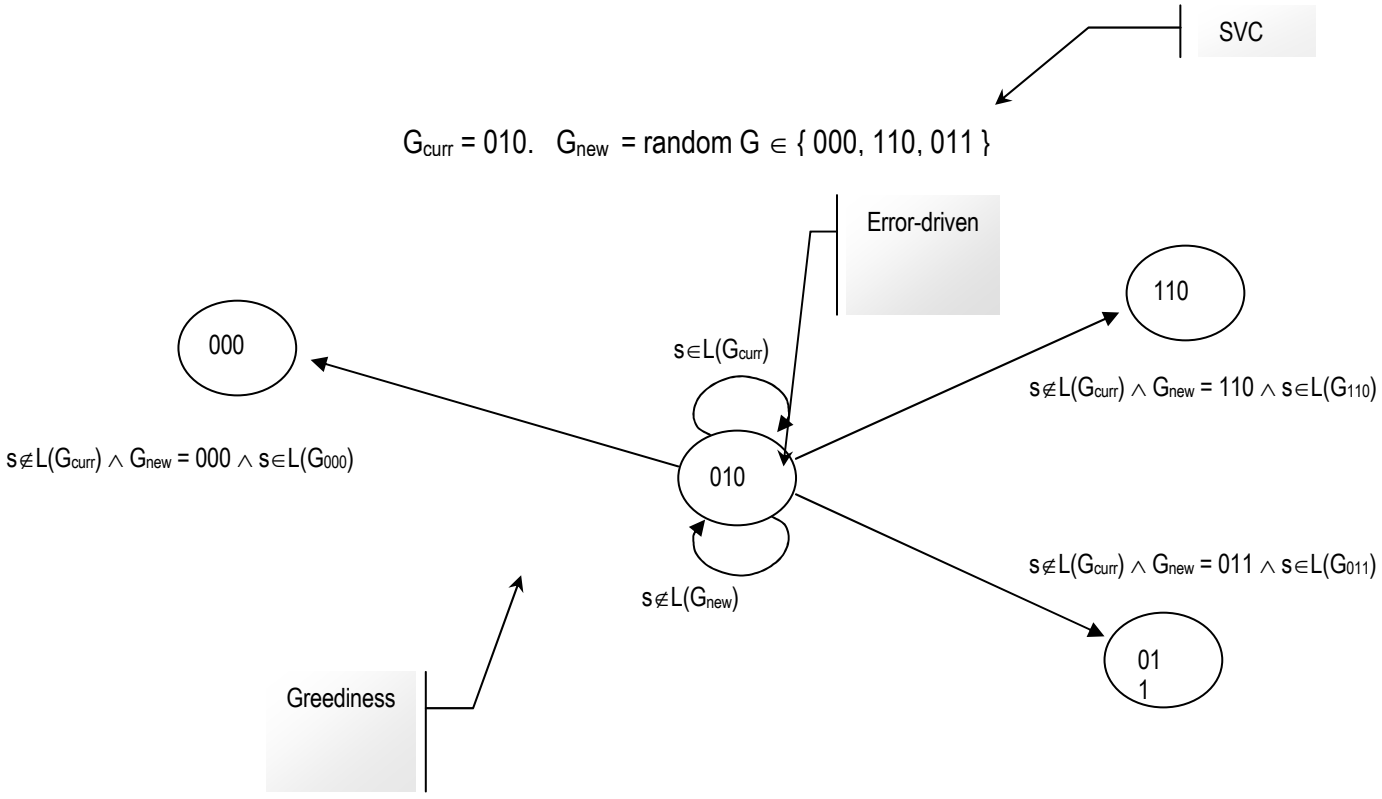


Figure 8. Local state space for the TLA. G_{curr} = 010.

- Let α_i denote the ***ambiguity factor*** of grammar G_i. α_i is the probability that G_i can parse an arbitrarily chosen sentence from the target language $L(G_{targ})$.

$$\Pr(s \in L(G_{targ}) \cup s \in L(G_i)) \text{ or } \Pr (s \in L(G_{targ}) \cap L(G_i))$$

- Let $\lambda_{i,j}$ denote the ***overlap factor.*** $\lambda_{i,j}$ is the probability that an arbitrarily chosen sentence from the target language is parsable by both G_i or G_j.

$$\lambda_{i,j} = \Pr (s \in L(G_{targ}) \cap L(G_i) \cap L(G_j)).$$

  Note that $\lambda_{i,j} = \lambda_{j,i}$ and when $j = targ$ then $\lambda_{i,j} = \alpha_i$.

- Let π_j denote the probability of ***picking*** or looking ahead at a new hypothesis grammar G_j from G_{curr}.[26] Following G&W, π_j is assumed to be constant across all parameters. That is, $\forall j$, $\pi_j = 1/n$, where n = the number of parameters. Note that *picking* is not the same as *moving*. The learner might or might not subsequently move to G_j depending on whether G_j can parse the current input.

Now, the probability that the learner moves from G_{curr} to G_{new} after receiving input sentence *s* is equal to the probability that G_{curr} can *not* parse *s*, times the probability that the TLA picks G_{new} in an attempt to parse *s* , times the probability that the G_{new} *can* parse *s* given that G_{curr} can not parse *s*.

---

[26] A more rigorous notation would be to incorporate two subscripts on π, the first denoting the current grammar. However the current grammar is always identifiable from the context and so, for readability, the corresponding subscript is not used.

$$Pr(G_{curr} \rightarrow G_{new}) = (1-\alpha_{curr})(\pi_{new}) \; Pr\,(G_{new} \text{ can parse } s \mid G_{curr} \text{ can't parse } s)$$

where

$$Pr(G_{new} \text{ can parse} \mid G_{curr} \text{ can't parse}) = (\alpha_{new} - \lambda_{curr\,,\,new}) \,/\, (1-\alpha_{curr})$$

So,

$$Pr(G_{curr} \rightarrow G_{new}) = (1-\alpha_{curr})(\pi_{new}) (\alpha_{new} - \lambda_{curr\,,\,new}) \,/\, (1-\alpha_{curr})$$

The $(1-\alpha_{curr})$ terms cancel leaving:

Equation 2:

$$Pr(G_{curr} \rightarrow G_{new}) = (\pi_{new}) (\alpha_{new} - \lambda_{curr\,,\,new})$$

And the probability that the learner retains the current grammar after consuming an input sentence is:

$$Pr(G_{curr} \rightarrow G_{curr}) = \alpha_{curr} + (1-\alpha_{curr}) \, \pi_k \, (1 - Pr(G_{new} \text{ can parse } s \mid G_{curr} \text{ can't parse } s))$$

$$= \alpha_{curr} + (1-\alpha_{curr}) \, \pi_k \, (1 - (\alpha_{new} - \lambda_{curr\,,\,new}) \,/\, (1-\alpha_{curr}))$$

$$= \alpha_{curr} + \pi_k \, ((1-\alpha_{curr}) - (1-\alpha_{curr}) (\alpha_{new} - \lambda_{curr\,,\,new}) \,/\, (1-\alpha_{curr}))$$

$$= \alpha_{curr} + \pi_k \, ((1-\alpha_{curr}) - (\alpha_{new} - \lambda_{curr\,,\,new}))$$

Equation 3:

$$Pr(G_{curr} \rightarrow G_{curr}) = \alpha_{curr} + \pi_k \, (1 - \alpha_{curr} - \alpha_{new} + \lambda_{curr\,,\,new})$$

's are sums over k, where $k \in \{ i : G_i \text{ differs from } G_{curr} \text{ by one parameter value} \}$

Retention of G_{curr} occurs if *s* can be parsed by the current grammar (occurs with probability α_{curr}) or if for all grammars G_k that differ by one parameter value from G_{curr}, the TLA picks G_k and G_k cannot parse *s*. Attaching probabilities to the arcs in

Figure 8 gives us the Markov state diagram of the local state space for the TLA (see Figure 9).

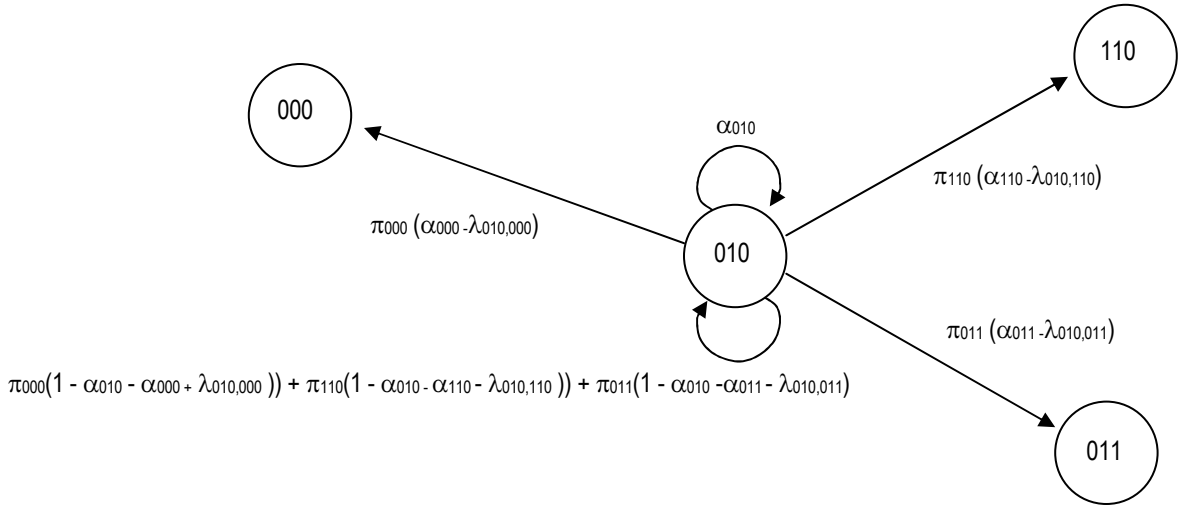


Figure 9. Local space of a Markov chain describing the outcomes TLA. The current grammar = 010.[27]

One detail remains to be sorted out. Since the TLA is error-driven and the input stream is generated entirely by G_{targ}, once the learner hypothesizes G_{targ} it will never leave. That is:

[27] For clarity, I use two arcs, 010 → 010, to separate the probabilities of 1) a successful parse by the current grammar (010), and 2) a failure by 010 together with a failure by an attempted grammar (one of 000, 010 and 011). Traditionally this would be a single arc with the probabilities added together.

Equation 4 $\Pr(G_{targ} \rightarrow G_{targ}) = 1$

This implies that there are no arcs leaving G_{targ}.

Equation 5 $\Pr(G_{targ} \rightarrow G_k) = 0 \quad \forall k, 0 < k < n$

Thus for every grammar G_i in $\mathcal{H}_n$, given that the values for π_i and α_i are known or can be calculated, one can determine the probability that the TLA will move from any grammar to any of its neighboring grammars. From these transition probabilities a transition diagram can be constructed and from the transition diagram we can construct a transition matrix. By calculating the fundamental matrix one can arrive at the numbers of sentences the TLA can be expected to consume during the acquisition process (see Chapter 2 for the mathematical details). The problem is that the number of states is exponential in the number of parameters, and hence the calculations are intractable for any reasonably sized domain.

3.3.1 G-Rings and Linguistic Smoothness

In order to obtain some insight into how the TLA performs in a domain of reasonable size, I will enforce a smoothness requirement. Recall that a G-Ring, $\mathcal{G}_i$, is a class of grammars such that each member of $\mathcal{G}_i$ differs from the target grammar by *i* parameter

values (see Chapter 2). Note that G-Rings strictly partition the grammars in the parameter space – every grammar belongs to one and only one G-Ring.

For example, $\mathcal{G}_0$ is the G-Ring that contains the target grammar. $\mathcal{G}_2$ is the G-Ring that contains all of the grammars that differ from the target grammar by two parameter settings. And $\mathcal{G}_n$ (n = the total number of parameters) is the G-Ring that contains the single grammar that has every parameter value opposite of the target.

By viewing the parameter space in terms of G-Rings, the TLA can be regarded as shifting (or not) from a grammar in one G-Ring to a grammar in one of the two neighboring G-Rings. Assume the TLA is entertaining some hypothesis $G_{curr} \in \mathcal{G}_i$ and receives a randomly selected string $s \in L(G_{targ})$. If s can be parsed by G_{curr}, the learner does nothing and thus remains in G-Ring $\mathcal{G}_i$ (following the error-driven constraint). Otherwise it changes one parameter value at random (the single value constraint or SVC) and attempts to parse s with the new grammar G_{new}. Importantly, G_{new} is strictly a member of either a G-Ring one closer to the target grammar, or one further away ($G_{new} \in \mathcal{G}_{i+1}$ or $G_{new} \in \mathcal{G}_{i-1}$). If G_{new} can parse s, G_{new} is adopted as the current hypothesis ($G_{curr} = G_{new}$); the learner has moved one G-ring closer to or further from G_{targ}. Otherwise G_{curr} remains unchanged (the Greediness constraint) and the TLA is ready to receive the next input sentence. In short, after receiving an input, the TLA will:

1) retain its current grammar and remain in the current G-Ring

2) move one G-Ring closer to the target, or

3) move one G-Ring away from the target

Note that staying in the current G-Ring means staying with the current grammar; the TLA's constraints do not permit shifting to a new grammar in the same G-Ring.

A conception of learning as passing through G-Rings is insufficient to tractably model TLA performance. The state space still consists of one state per grammar (the total number of grammars is 2^n). However, if Weak Smoothness is imposed (*Weak Smoothness Requirement* – All the members of a G-Ring can parse s with an equal probability, see Chapter 2), the state space can be collapsed; each *class* of grammars in a G-Ring is represented by a *single* state. Probabilities reformulated in terms of G-Rings and Weak Smoothness are substantially the same as those derived earlier in Section 3.3 . The derivation of the transition probabilities follows immediately below.

- α_i denotes the *ambiguity factor* of G-Ring *i*. α_i is the probability that a grammar $G \in \mathcal{G}_i$ can parse an arbitrarily chosen sentence from the target language. It can be viewed as the probability that an input can be parsed by an arbitrary grammar in a particular G-Ring.

- Let $\lambda_{i,j}$ denote the *overlap factor*. $\lambda_{i,j}$ is probability that an arbitrarily chosen sentence from the target language is parsable by both a grammar in G-Ring $\mathcal{G}_i$ and a grammar in G-Ring $\mathcal{G}_j$. Note that $\lambda_{i,0} = \lambda_{0,i} = \alpha_i$.

- π_{i-m} denotes the probability of *picking* or "looking ahead" at a new hypothesis grammar G_{new} that belongs to a G-Ring *m* rings closer to the target than ring *i*. Likewise, π_{i+m} denotes the probability of picking a new hypothesis grammar *m* rings further from the target. The SVC ensures that $m = \pm 1$ for all $\mathcal{G}$.

Now the probability that the learner moves from G-Ring $\mathcal{G}_i$ to $\mathcal{G}_{i+1}$, where $G_{curr} \in \mathcal{G}_i$ is:

Equation 6 $$\Pr(\mathcal{G}_i \rightarrow \mathcal{G}_{i+1}) = (\pi_{i+1})(\alpha_{i+1} - \lambda_{i,i+1})$$

Similarly, the probability that the learner moves from $\mathcal{G}_i$ to $\mathcal{G}_{i-1}$ is:

Equation 7 $$\Pr(\mathcal{G}_i \rightarrow \mathcal{G}_{i-1}) = (\pi_{i-1})(\alpha_{i-1} - \lambda_{i,i-1})$$

Finally, the probability that the learner retains the current grammar (which implies that the learner will stay in the same G-Ring) is:

Equation 8 $$\Pr(\mathcal{G}_i \rightarrow \mathcal{G}_i) = \alpha_i + \sum_{k \in \{i-1,\ i+1\}} \pi_k (1 - \alpha_i - \alpha_k - \lambda_{i,k})$$

This occurs if *s* can be parsed by the current grammar or if the TLA picks a grammar from the ring one value closer to the target, or one value further from the target, and

the new grammar cannot parse *s*. (The algebraic derivation of Equation 8 is identical to that of Equation 3, see Section 3.3 above.)

What remains is to give a definition for π_{i-1} – the probability that the TLA chooses to look ahead *towards* the target grammar and π_{i+1} – the probability that the TLA chooses to look ahead to a grammar in a G-Ring *away* from the target grammar. (Keep in mind that, as outlined at the end of Chapter 2, in what follows the target grammar is assumed to be the vector consisting of all 1's).

The SVC dictates that for each learning event the TLA can change the value of only a single parameter. If the TLA chooses to flip a parameter value from 0 to 1, G_{new} will be in a G-Ring closer to the target than G_{curr} (e.g. 111, see Figure 10 below). If it chooses to flip a parameter from 1 to 0, it will be in a G-Ring further from the target. If all parameters are equally probable as candidates for a flip, and *n* denotes the number of parameters that defines each grammar in the parameter space, then:

Equation 9 $\quad \pi_{i-1} = i / n$

This is simply the hamming distance from the target (i.e., the number of zeros in every grammar vector in G-Ring $\mathcal{G}_i$) divided by the total number of parameters.[28]

Similarly:

[28] Recall from Chapter 2, that the *hamming distance* is the number of bits (i.e. parameter values) by which two bit vectors (grammars) differ. In this case, since the target is equal to the bit vector made up of all ones, the hamming distance is calculated as the number of zeros.

Equation 10 $\pi_{i+1} = (n - i) / n$

This is the number of ones divided by the total number of parameters.

Using this framework I next construct Markov transition matrices and determine under what circumstances the SVC aids in the learning process and how strong is the effect of a smooth parametric domain on TLA learning. Of particular interest will be what happens as the domain increases in size.

3.3.2 An Example Application: The feasibility of TLA learning in $\mathcal{H}_3$

In order to illustrate the methodology employed in the experiments described throughout the remainder of the chapter, this section will fully work through an application of the framework for the TLA executing in a space of three parameters ($\mathcal{H}_3$, Figure 10) . Subsequent sections will present results for $\mathcal{H}_5$, $\mathcal{H}_{10}$, $\mathcal{H}_{20}$ and $\mathcal{H}_{30}$.

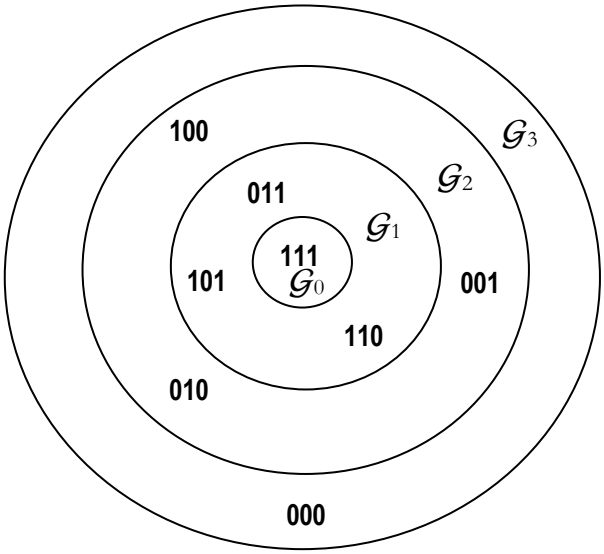


Figure 10. Parameter space $\mathcal{H}_3$ with G_{targ} = 111.

3.3.2.1 The Transition Probabilities

To be concrete, assume that there is a probability of 0.5 that a sentence can be parsed by grammars in $\mathcal{G}_1$; a probability of 0.3 that a sentence can be parsed by grammars in $\mathcal{G}_2$ and a probability of 0.2 that a sentence can be parsed by both a grammar in $\mathcal{G}_1$ and a grammar from $\mathcal{G}_2$.[29] Furthermore, assume that the learner is currently hypothesizing G_{110} (a grammar in $\mathcal{G}_1$) and has received a sentence that can not be parsed. Also assume that the probability that a grammar in $\mathcal{G}_3$ can parse an arbitrary input is .4 and that a sentence can be parsed by both a grammar in $\mathcal{G}_3$ and $\mathcal{G}_2$ is .1.

To summarize the assumptions that are in place:

$\alpha_3 = .4,\ \alpha_2 = .3,\ \alpha_1 = .5,\ \lambda_{0,1} = \alpha_1 = .5,\ \lambda_{1,2} = .2,$ and $\lambda_{2,3} = .1$

Now, what is the probability that the TLA will entertain a grammar in $\mathcal{G}_2$ or $\mathcal{G}_0$ after receiving an input sentence?

Applying Equation 6:

$$\Pr(\mathcal{G}_1 \rightarrow \mathcal{G}_2) = \pi_2 (\alpha_2 - \lambda_{1,2})$$

$$= ((3-1)/3)\ (0.3 - 0.2)$$

$$= 0.0667$$

[29] Note that the space is not strongly smooth. Recall from Chapter 2, that the *Strong Smoothness Requirement* mandates that the closer the G-Ring is to the target grammar, the higher the probability that a grammar in the ring can parse an arbitrary input sentence generated by the target grammar.

$$\Pr(\mathcal{G}_1 \rightarrow \mathcal{G}_0) = \pi_0 \, (\alpha_0 - \lambda_{0,1})$$

$$= (1/3) \, (1.0 - .5)$$

$$= 0.1667$$

Similarly, the probability that the TLA will stay in $\mathcal{G}_1$ can be established as follows using Equation 8.

$$\Pr(\mathcal{G}_i \rightarrow \mathcal{G}_i) = \alpha_i + \sum_{k = \{ i-1, \; i+1\}} \pi_k \, (1 - \alpha_i - \alpha_k - \lambda_{i,k})$$

$$\Pr(\mathcal{G}_1 \rightarrow \mathcal{G}_1) = \alpha_1 + \pi_0(1 - \alpha_1 - \alpha_0 + \lambda_{1,0}) + \pi_2(1 - \alpha_1 - \alpha_2 + \lambda_{1,2})$$

$$= .5 + 1/3 \, (1 - .5 - 1 + .5) + 2/3 \, (1 - .5 - .3 + .2)$$

$$.5 + 1/3 \, (1 - .5 - 1 + .5) + 2/3 \, (1 - .5 - .3 + .2)$$

$$= .7667$$

All possible transitions are generated to yield the following transition matrix. The shaded submatrix contains the probabilities of the transient (non-absorbing) states from which the fundamental matrix is derived (see Chapter 2).

	$\mathcal{G}_3$	$\mathcal{G}_2$	$\mathcal{G}_1$	$\mathcal{G}_0$
$\mathcal{G}_3$	0.8000	0.2000		
$\mathcal{G}_2$	0.1000	0.7000	0.2000	
$\mathcal{G}_1$		0.0667	0.7667	0.1667
$\mathcal{G}_0$				1.0000

From this, the corresponding fundamental matrix can be derived (see Chapter 2):

	$\mathcal{G}_3$	$\mathcal{G}_2$	$\mathcal{G}_1$
$\mathcal{G}_3$	8.5000	7.0000	6.0000
$\mathcal{G}_2$	3.5000	7.0000	6.0000
$\mathcal{G}_1$	1.0000	2.0000	6.0000

Adding the values of the first row gives the number of sentences consumed on average before convergence starting from state G_3. In this example: 8.5 + 7.0 + 6.0 = 21.5.

This example was presented to illustrate the step by step process by which the transition probabilities are derived and then employed in the framework that will be utilized in the rest of the dissertation. The initial assumptions were arbitrarily chosen to illustrate the technique involved. A note of caution is needed. Although it is tempting to induce regularities from single examples,[30] it is important to note that the effect of the λs and αs on the fundamental matrix is complex and requires experimentation with a wide range of values. Results from such experimentation are given in what follows.

3.3.2.2 Constraints on λ and α and Subset Avoidance

There are two constraints that need to be maintained on λ and α. One constraint needs to be in place due to the nature of probabilities. The other is imposed so as not to permanently block the learner from attaining the target grammar.

[30] One might induce that less inputs are consumed starting from grammars closer to the target because the sum of the third row (closest to the target) indicates that only 9 sentences will be consumed as versus 21.5 – the sum of the first row (further from the target).

(1) $\alpha_i + \alpha_j - \lambda_{i,j} < 1$

(2) $0 \leq \lambda_{i,j} < \alpha_i \leq 1$ and $0 \leq \lambda_{i,j} < \alpha_j \leq 1$

Constraint (1) indicates that the probabilities of two grammars being able to parse an arbitrary sentence must not be greater than the entire probability space. For example, if G_i can parse a sentence with probability of .7 and G_j can parse a sentence with probability of .6, then they both must be able to parse some of the same sentences. Otherwise the probability that either G_i *or* G_j can parse a sentence is 1.3 which is greater than 1.0 which is impossible. Another way of saying this is that the number of sentences in $L(G_i) \cup L(G_j)$ must be less than or equal to the number of sentences in $L(G_{targ})$.

Constraint (2) states that the probability of two grammars parsing a sentence must be *strictly* less than the probability that either single grammar yields a successful parse. $\lambda_{i,j} = \alpha_j$ implies that $L(G_j) \subseteq L(G_i)$. There is no set-theoretic reason not to allow such a subset relation, but if a learner's current hypothesis was G_j, the learner would be blocked from attaining the target grammar due to the error-driven constraint. Specifically, this occurs when an error-driven learner with Greediness is entertaining the grammar that generates the superset, in this example G_i. It will never hypothesize the subset G_j on a parse failure because all of the sentences in G_j are (by definition) included in the language generated by G_i; if a sentence cannot be parsed by G_i it cannot be parsed by G_j. Hence the learner will never encounter a sentence that: i) causes a

parse failure with G_i, and ii) can be resolved by attempting a parse with G_j. For the TLA executing in the G-Ring framework presented in this thesis, if this subset relation is encountered, it creates an insurmountable learnability problem since, at some point, the learner must pass through each G-Ring (due to the SVC) on the way to the target. Thus, constraint (2) guarantees that there is a possibility greater than zero of encountering a sentence that will allow the learner to pass through the current ring.[31]

3.3.2.3 Finding the Most Advantageous Space for Efficient TLA Learning

In the example given at the end of section 3.3.2.1, the TLA consumed 21.5 input sentences on average before it attained the target grammar, starting from 3 parameters away. In order to discover under what conditions the TLA functions most efficiently, different values for α and λ are applied to the Markov formalization of TLA learning described up to this point. I ran several experiments in spaces of various sizes. All were run on parameter spaces that were weakly smooth. The experiments involved:

- fixing the parameter space so that grammars in all G-Rings are equally likely to parse an input sentence. $\forall$ i,j $\alpha_i = \alpha_j$.

[31] In G&W's 3 parameter domain for testing the TLA, no language is a proper subset of any other. There is no mechanism or constraint built into the TLA that prevents superset hypotheses.

- fixing the parameter space so that it is strongly smooth in a linear fashion.[32] $\forall\, i > 0,\ i < \alpha_i - \alpha_{i+1}$ and $\forall\, i > 0, j > 0,\ \alpha_i - \alpha_{i+1} = \alpha_j - \alpha_{j+1}$

- finding optimal values for α and λ regardless of any strong smoothness relationship between G-Rings. This allows for 'irregular' spaces in which G-Rings are not related to adjacent G-Rings by strong smoothness.

Since the probabilities are real-valued and interact with each other and the constraints in complicated ways, searching for the most advantageous probabilities requires use of an optimization algorithm. The results that follow were arrived at by applying a standard non-linear constrained optimization algorithm: The Generalized Reduced Gradient (GRG) algorithm (Lasdon and Waren, 1978) as implemented by FrontLine Systems, Inc. I used Microsoft's Excel spreadsheet application as the front end to the optimizer.

The basic approach was to set the objective function of the optimizer to minimize the average number of sentences expected for the TLA to acquire the target grammar subject to the constraints outlined in the preceding section 3.3.2.2.

I use $\mathcal{H}_3$ as a concrete example to illustrate the methodology employed in choosing the *decision variables* (the quantities that the optimizer is free to manipulate) in order to

[32] A non-linear, but still strongly smooth, domain would entail an acceleration in the probability of parsing success in G-Rings approaching the target. For example where $\alpha_i = 2\ \alpha_{i+1}$.

achieve the *objective function* – a minimum number of input sentences. Results from larger spaces and discussion appear in the sections that follow.

I first ran five experiments by fixing α at .01, .1, .5, .9, and .99 for *all* G-Rings dictating a uniform (not strongly smooth) space and defining λ as a percentage of α (λ% or *percentage overlap*). Formulating λ as a percentage was done in order to enforce the subset avoidance constraint (constraint (2) outlined in Section 3.3.2.2 above). Given these fixed values of α, the optimizer was then able to find the most advantageous percentage overlap (λ%) – the value for λ% that minimized the average number of sentences required for the TLA to attain the target (subject to the appropriate constraints). I then had the optimizer find the lowest values for *both* α and λ%.

These six results for $\mathcal{H}_3$ are depicted in Table 5 below.

α	λ%	Avg # sentences until convergence
0.01	0.0001	310
0.10	0.0010	38
0.40	0.0002	19*
0.50	0.0002	20
0.90	0.8001	100
0.99	0.9801	1003

Table 5. Results from 6 feasibility experiments simulating TLA learning in $\mathcal{H}_3$ with all G-Rings equally likely to parse an input sentence. The optimizer found .4 as the value for α and .0002 for λ% that minimizes the number of sentences the TLA requires to attain the target grammar.

Note that the optimizer chooses values for λ that roughly correlate with α. This is consistent with the results that follow. In domains of high ambiguity the TLA

performs best when the languages are clustered together very tightly. An explanation and discussion is presented in the next section (Section 3.2).

I then set the αs so that the space was linearly smooth (not uniform). And again the optimizer minimized the average number of sentences required for convergence by picking λ%. Finally, I let the optimizer pick all values. As expected, the algorithm picked values that resulted in a lower number of required sentences. This is labeled *Optimal* in Table 6.

	λ%	α_3	α_2	α_1	# sents
Linear increase in smoothness	.99	0.25	0.50	0.75	**22**
Optimal	.03	0.00	0.28	0.44	**16**

Table 6. Effects of strong smoothness on TLA performance in $\mathcal{H}_3$.

The optimizer works by implementing an iterative hill-climbing strategy. Each step involves adjusting the decision variables in several directions and checking the rate of change of the objective function. After determining the best direction, the optimizer 'plugs in' those values and repeats the process. The process ends when the optimizer can no longer adjust the cells so that a better objective is obtained. Unfortunately there is no guarantee that the true optimal objective will be obtained. This is because there may be a locally optimum point (or several locally optimum points) that look to the optimizer as truly optimal; no adjustment of the decision variables improves the value of the objective function at that step in the process.

Thus, in order to determine the validity of the optimization, I executed the algorithm many times with randomly generated starting values for the decision variables. In theory, by starting off from many different points, if local maxima exist, each run should yield a different outcome. Fortunately I found the results consistent regardless of the starting values. There was some small variation. This leads to the conjecture that there are several small local maxima or 'bumps' in the vicinity of the true globally optimal value.

However it is important to stress that this is an empirical conclusion. Although the evidence is strong, there might be 'true' optimal values of λ% and the α's that are quite different than the presented results.[33]

A final limitation of the approach is the need to formulate λ as a percent of α rather than freely allowing arbitrary values for the λ's. When left free to chose the λ's, the optimizer consistently focused on (and tried to overcome) the constraint intended to avoid subset relations between G-Rings. Specifically it attempted to make every *other* ring a subset of its neighboring ring. Although a 'clever' strategy in that it effectively reduces the number of G-Rings by half, (some of these experiments actually produced numbers somewhat lower in terms of number sentences consumed by the TLA than

[33] TLA learning in a weakly smooth domain can be formally modeled as a *random walk in two dimensions*. There is abundant mathematical literature on random walks. Unfortunately a preliminary exploration of a formal approach became unwieldy without certain homogeneity assumptions in place (e.g., setting all λs equal to each other). Still, I would not rule out the possibility that such a formulation could constructively be developed.

specifying λ as a percent), the resulting parameter space is far too removed from linguistic plausibility to merit discussion.

3.4 The Feasibility of the TLA

3.4.1 Uniform probability of a successful parse

Table 7 presents results from twenty-four experiments simulating TLA performance in uniform parameter spaces larger than $\mathcal{H}_3$ – spaces in which the probability of a successful parse of (an input sentence) is uniform across G-Rings.

$\mathcal{H}_5$

α	λ%	Avg # sentences until convergence
0.0100	00.00	1,198
0.1000	00.00	151
0.3802	00.00	81*
0.5000	00.20	85
0.9000	89.00	428
0.9900	99.09	4,396

$\mathcal{H}_{10}$

α	λ%	Avg # sentences until convergence
0.0100	00.00	17,387
0.1000	00.00	2,772
0.2856	00.00	2,005*
0.5000	00.20	2,374
0.9000	89.00	11,882
0.9900	99.09	120,471

$\mathcal{H}_{20}$

α	λ%	Avg # sentences until convergence
0.0100	00.00	7,344,041
0.1000	00.00	1,793,571
0.1967	00.00	1,624,850*
0.5000	00.20	2,223,099
0.9000	89.00	11,120,586
0.9900	99.09	111,840,694

$\mathcal{H}_{30}$

α	λ%	Avg # sentences until convergence
0.0100	00.00	5,086,768,462
0.1000	00.00	1,593,264,558
0.1618	00.00	1,528,362,614*
0.5000	00.20	2,227,687,316
0.9000	89.00	11,141,678,763
0.9900	99.09	111,821,049,096

Table 7. Results from 24 feasibility experiments simulating TLA learning in parameter spaces of 5, 10, 20 and 30 parameters with all G-Rings equally likely to parse an input sentence. The asterisk indicates the value that the optimizer found which minimizes the number of sentences the TLA is expected to consume in order to attain the target grammar.

Even at optimal rates of ambiguity and language overlap (denoted by * in Table 7), parameter spaces with a uniform probability of a successful parse are not conducive to good TLA performance. As anticipated by Niyogi and Berwick, the number of sentences required in each of the learning situations studied, even at the optimal rate of ambiguity and overlap, is still notably larger than a 'blind' learner that simply chooses grammars at random without taking into account any relevant features of the input (for example success or failure of a parse test). Such a learner – which simply hypothesizes

a grammar at random whenever a sentence is received – would require 2^n (n = number of parameters) sentences on average to converge on the target grammar.[34] Figure 11 shows this exponential increase in the size of the input sample plotted against the four optimal values of α and λ from Table 7. Clearly this rate of performance is unsuitable for any psychologically plausible model of acquisition that is expected to converge on a target grammar in a parameter space of a reasonable size.

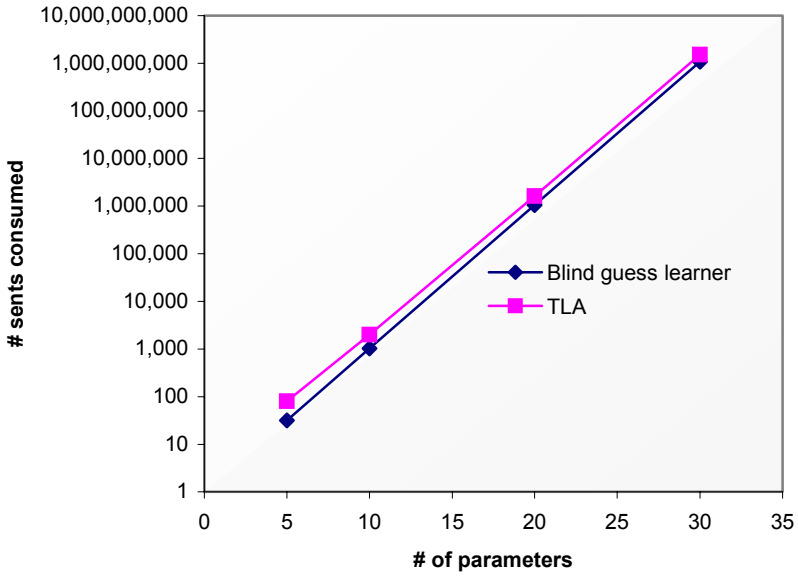


Figure 11. Logarithmic plot of the optimal number of sentences expected to be consumed before convergence against the number of grammars in the parameter space. The TLA at its best requires more sentences than a learner that blindly selects grammars at random.

[34] Sakas and Fodor (1997) discuss a variation of a purely random learner that retains the error-driven constraint – the *Unconstrained Error-Driven* learner (*UED*). They demonstrate that the number of sentences required on average for this learner is bounded from below by the purely random learner. In the optimal case, the UED requires exactly one sentence less than the random learner (= $2^n - 1$).

The extreme number of inputs consumed by the TLA is due to the fact that the TLA is an extremely conservative learner. The SVC keeps the learner from exploring grammars far away from the current grammar. And even within the circumscribed local space sanctioned by the SVC, Greediness allows a shift to a new grammar hypothesis only if the candidate grammar is 'good enough' to license the current input. For almost all combinations of ambiguity and overlap, the SVC and Greediness bias the TLA to remain in place, retaining G_{curr} rather than adopting a new hypothesis grammar.[35] For example, Table 8 below indicates the probabilities, for each G-Ring in $\mathcal{H}_{10}$, of: 1) shifting to a G-Ring further from the target, 2) retaining the current grammar and 3) shifting to a grammar in a G-Ring closer to the target grammar given the optimal settings (α =0.2856, λ% = 0; see Table 7).

G-Ring that contains G_{curr}	Away from G_{targ}	Retain G_{curr}	Towards G_{targ}
$\mathcal{G}_{10}$		0.720	0.280
$\mathcal{G}_{9}$	0.028	0.720	0.252
$\mathcal{G}_{8}$	0.056	0.720	0.224
$\mathcal{G}_{7}$	0.084	0.720	0.196
$\mathcal{G}_{6}$	0.112	0.720	0.168
$\mathcal{G}_{5}$	0.140	0.720	0.140
$\mathcal{G}_{4}$	0.168	0.720	0.112
$\mathcal{G}_{3}$	0.196	0.720	0.084
$\mathcal{G}_{2}$	0.224	0.720	0.056
$\mathcal{G}_{1}$	0.252	0.676	0.072

Table 8. Probabilities that the TLA will retain the current grammar or shift towards or away from the target grammar in $\mathcal{H}_{10}$ given α = 0.2856 and λ = 0.

[35] The one exception is when α = .5 and λ = 0. In this case the probability that the TLA shifts to a new grammar is exactly equal to the probability that it retains the current grammar.

Since the space is uniform with regard to α, attempting to force the TLA to move by lowering α won't work. Due to the Greediness constraint, the probability of retaining G_{curr} is determined by two factors: α (if G_{curr} can parse the current sentence, the TLA retains G_{curr}), and the probability of a parse failure in the event that G_{curr} can't parse the current input in which case the TLA attempts a shift by testing a candidate grammar. But the probability that the candidate grammar will license the input sentence is no better than the probability that G_{curr} will license the sentence.[36] Therefore low values of α dictate that the TLA will retain the current grammar due to parse failure when applying the candidate to the sentence, and high values of α will also force the TLA to retain G_{curr} due to a high rate of successful parsing by G_{curr}. The net effect is that values of α somewhat below 0.5 are the most conducive for TLA learning.[37]

Still, whatever the value of α, a high rate of grammar retention is exhibited by the TLA as depicted in Table 8 above. This might lead to the conclusion that such an unavoidably high rate of retention (given a homogeneous value of α across G-Rings) is the cause of the exponential workload that the TLA bears. This is not the case. To see

---

[36] This is so when the candidate is actually the target grammar, in which case the probability is 1.

[37] An interesting phenomenon to note is that the optimal value of α decreases as the number of parameters increases. This leads to the speculation that TLA performance will be optimal in large parameter spaces only at low ambiguity rates. This might be because the tradeoff between conservatism and exploration is too costly in large spaces. The Error-driven constraint compels conservatism if the ambiguity rate is high. But it will also result in aggressive performance in terms of attempting new hypotheses if the ambiguity rate is low. The more parameters there are, it appears the more exploration is required. Discussion along this line follows in subsequent sections for TLA performance in strongly smooth parameter spaces.

why this is so, it is useful to cast TLA learning in a parameter space as a special type of Markov process – an *absorbing Random Walk with a reflecting barrier.*[38] The idea is that the system can shift to one of two adjacent states except at one 'end' where there is only one adjacent state. For example, in the transition diagram depicted in Figure 12, the learner can leave $\mathcal{G}_3$ only to enter state $\mathcal{G}_2$. However, from the other non-absorbing states, the learner has a choice of two states to shift to.

Figure 12 exactly models TLA behavior in parameter space $\mathcal{H}_3$. Probabilities r_1, r_2, r_3, etc. represent the probabilities that the TLA will *retain* the current grammar and are determined by Equation 8. Probabilities p_1, p_2, p_3, etc. represent the probabilities that the TLA will shift *towards* the target grammar, and q_1, q_2, q_3 etc. that it will shift *away* from the target; these probabilities are determined by Equation 7 and Equation 6.

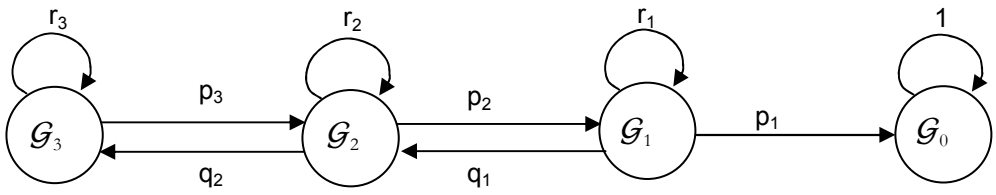


Figure 12. A Markov chain representing a random walk with one absorbing state $\mathcal{G}_0$ and one reflecting barrier at $\mathcal{G}_3$.

Surprisingly, the factor most affecting the TLA's performance is not the high rate of grammar retention; rather, it is the decreasing probability of moving towards the target (the last column in Table 8). To be concrete, in $\mathcal{H}_{10}$, if the learner incorporated an

[38] This is frequently referred to as the *Drunkard's Walk*, the bar serving alcohol being the reflecting state, and the drunkard's home being the absorbing state.

equal bias to move towards or away from the target (with modifications to accommodate the absorbing and reflecting states), and maintained the optimal rate of grammar retention (.72 from center column of Table 8) we would obtain Table 9 below.[39]

G-Ring that contains G_{curr}	Away from G_{targ}	Retain G_{curr}	Towards G_{targ}
$\mathcal{G}_{10}$		0.720	0.280
$\mathcal{G}_9$	0.140	0.720	0.140
$\mathcal{G}_8$	0.140	0.720	0.140
$\mathcal{G}_7$	0.140	0.720	0.140
$\mathcal{G}_6$	0.140	0.720	0.140
$\mathcal{G}_5$	0.140	0.720	0.140
$\mathcal{G}_4$	0.140	0.720	0.140
$\mathcal{G}_3$	0.140	0.720	0.140
$\mathcal{G}_2$	0.140	0.720	0.140
$\mathcal{G}_1$	0.140	0.500	0.360

Table 9. Probabilities, for a modified TLA that is **equally likely** to shift towards or away from the target grammar.

Applying the usual techniques we discover that the total number of sentences required for this system is only 315 compared against the 2,005 required by the TLA. Increasing the retention rate to 0.9 for the random walk algorithm yields an average number of inputs consumed of only 831. The random walk is surprisingly efficient compared to the TLA. This is because the number of inputs that can be expected to be consumed

[39] The transition probabilities surrounding the two end states are not equal. This follows from the fact that there is no G-Ring to the 'left' of the ring furthest from the target grammar so the probability of shifting to the right from $\mathcal{G}_{10}$ is doubled. Similarly, since the retention rate of $\mathcal{G}_0$ (= r_0 in Figure 12) is equal to 1.0, there will not be a parse failure when $\mathcal{G}_0$ is the candidate from $\mathcal{G}_1$. The corresponding probabilities (those emitting from $\mathcal{G}_1$) were derived using Equation 9 and Equation 10 and setting $\pi_{i+1} = \pi_{i-1}$ as described in the text that follows.

by the random walk learner (unlike the TLA) decreases as the learner approaches the target. To see why this is the case, consider the learner in a state two transitions away from the target. Although the system may wander away from the target, *on average*, over many trials, it will require fewer inputs to achieve the target from a state two transitions away than from a state further (say four transitions) away. This decreasing consumption is depicted in Figure 13.

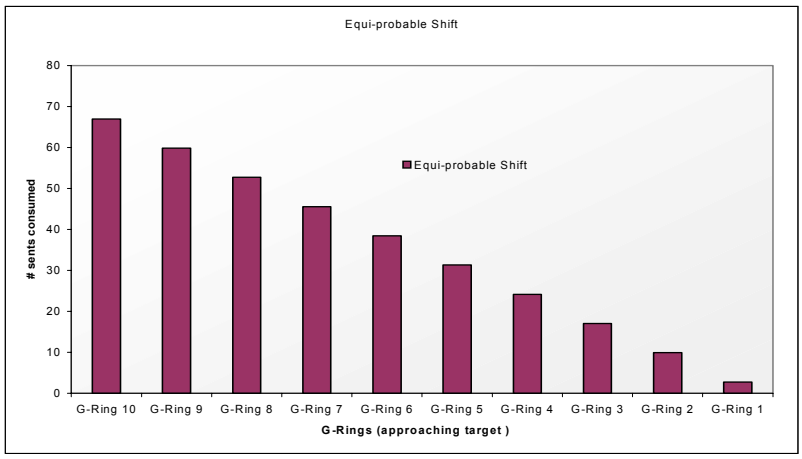


Figure 13. The number of sentences consumed in each state by a random walk learner.

It is informative to contrast the decreasing input consumption of a random walk with the input consumption of the TLA. This is shown in Figure 14 below.

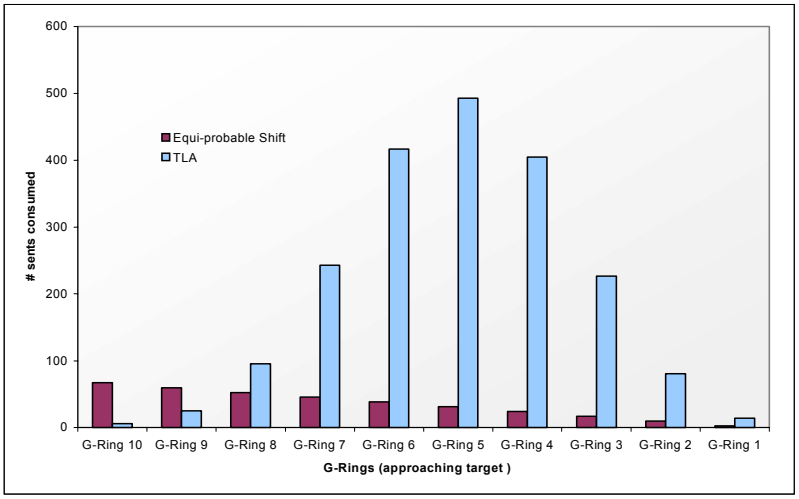


Figure 14. The average number of sentences consumed in each state for both the TLA and a learner that will shift towards or away from the target grammar with an equal probability.

The TLA spends the majority of its time consuming input sentences in G-Rings mid-distance from the target state. This is the result of the decreasing probability of moving towards the target as the target is approached. Since the values of α for all G-Rings are equal, the factors affecting this decrease are the probabilities π_{i+1} and π_{i-1} (see Equation 9 and Equation 10) of picking a grammar in a G-Ring one further or one closer to the target.

Recall:

Equation 9 $\pi_{i-1} = i / n$

Equation 10 $\pi_{i+1} = (n - i) / n$

where *i* is the (hamming) distance from the target and *n* is the number of parameters that need to be set. Basically these equations assert that the probability of choosing a

grammar in a G-Ring closer to the target is inversely proportional to the distance of G_{curr} from the target. The effect of these equations can be seen in the last column in Table 8.

It is important to note that these probabilities are unaffected by *any* characteristics of the input sample, they are strictly dictated by the learning algorithm – specifically by the SVC. Roughly, this is because as the learner approaches the target, most parameters are set correctly, so a change is more likely to shift the learner away from the target rather than towards it. (Whereas for the random walk learner, a shift is *equally* likely to shift in either direction.)

Since the SVC is the primary source of the poor performance exhibited by the TLA (in a uniform parameter space) one might conjecture that improvement would result from a simple modification to the TLA that replaces the SVC with a heuristic that guides the TLA with a 50/50 chance of moving to a G-Ring one further or one closer to the target (setting $\pi_{i+1} = \pi_{i-1} = .5$). In the examples presented here, where it is assumed for concreteness that the target is the grammar vector consisting entirely of 1's, such a strategy could be put into operation, and it would certainly dramatically increase learning efficiency. Unfortunately this heuristic is impossible to implement in the general case where the target could be any combination of parameter values. In this case, the learner would have no way of determining in which direction the target lies. For example, if the target were 00101 in $\mathcal{H}_5$, and the current grammar were 01111,

toggling the first 0 to a 1 would move the learner away from the target. Whereas if the target where 11101, toggling the first 0 would move the learner closer. Without the help of an oracle that points the way towards the target, the TLA is bound to the values of π as formulated above. Hence, the TLA is doomed to be overly conservative. As it approaches the target, in a uniform space, the probability of moving even closer decreases to a level that forces the TLA to spend an inordinate amount of time consuming sentences midway between the worst grammar hypothesis and the target.

It's clear that what is needed for feasible TLA learning is a factor which will 'pull' the TLA closer to the target, a factor which is strong enough to mitigate the impeding deceleration caused by the SVC. In the next section I show that Greediness can serve this function, but only in strongly smooth parameter spaces.

Summary of this section: In a parameter space in which the probability of a successful parse is homogeneous across all grammars, the number of input sentences required by the TLA rises exponentially with the number of parameters that need to be set, even at rates of ambiguity that are optimal for TLA learning. This poor performance is primarily due to the algorithm's SVC and the SVC is unaffected by the degree of ambiguity of the input sample. Notably, TLA performance is worse than both a learner which guesses grammars at random (with an optimal input sample) and a random walk learner.

3.4.2 Linear Strong Smoothness

Next I consider the feasibility of TLA performance in a linearly strongly smooth domain – a domain in which the probability of a successful parse correlates with the similarity of the current grammar to the target grammar. In this space the TLA is a feasible learner.

To test this, I fixed the values of α so that they were linearly increasing as shown in Equation 11:[40]

Equation 11 $$\alpha_i = (n - i + 1) / (n+1)$$

As usual, i = hamming distance from the target and n = # of parameters that need to be set.

I ran four experiments, one in each parameter space ($\mathcal{H}_5$, $\mathcal{H}_{10}$, $\mathcal{H}_{20}$ and $\mathcal{H}_{30}$), having the optimizer find the value of the overlap rate λ% that minimizes the number of sentences expected to be consumed by the TLA. Consistently, the optimizer found the optimal overlap rate to be as high as possible while respecting the constraint prohibiting subset relations between G-Rings (i.e. $\alpha < \lambda$). The results from experiments run on the four domains are depicted in Table 10 and plotted in Figure 15.

---

[40] This imposes strong smoothness such that the probability of a successful parse (α) is equally distributed across G-Rings from $1/(n+1)$ for $\mathcal{G}_n$ to 1.0 for $\mathcal{G}_0$. In general, a parameter space could be strongly smooth over any distribution on αs. All that strong smoothness requires is that $\alpha_{i+1} < \alpha_i$.

Although the plot indicates that the rate of increase in input consumption is greater than linear, the relatively small number of sentences required for the TLA to attain the target in a space of 30 parameters lends strong support to the claim that the TLA *is* a feasible learner in strongly smooth domains.

G-Ring	λ%	Avg # sentences until convergence
$\mathcal{H}_5$	99.99	69
$\mathcal{H}_{10}$	99.99	312
$\mathcal{H}_{20}$	99.99	1,522
$\mathcal{H}_{30}$	99.99	3,777

Table 10. Optimal number of sentences required on average for the TLA to acquire G_{targ} in a linearly smooth domain.

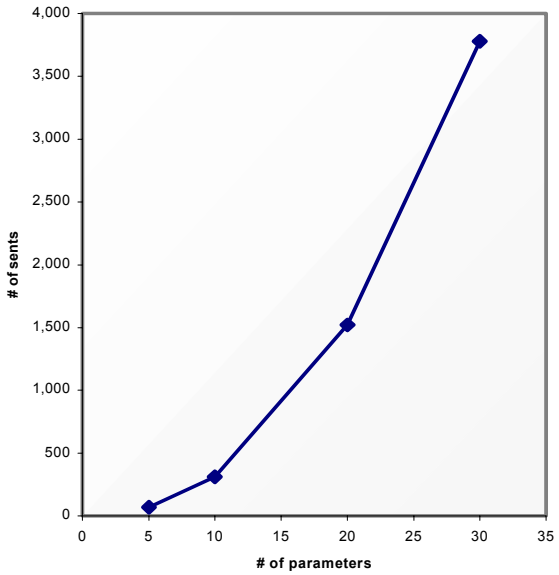


Figure 15. Plot of number of sentences required on average for the TLA to acquire G_{targ} in a linearly smooth domain against the number of parameters in the parameter space.

The learner is still conservative. Table 11 depicts the probabilities of retaining the current grammar, and moving towards or away from the target grammar in $\mathcal{H}_{10}$. As in the uniform space described in the preceding section, there is a high rate of grammar retention which increases as the TLA approaches the target The TLA also exhibits a decreasing probability of moving towards the target as it approaches it. However, when the space is linearly smooth, the effect of the Greediness constraint is able to overcome the conservativeness caused by the SVC. This can be seen by considering the first column of probabilities in Table 11.

G-Ring	Away from G_{targ}	Retain G_{curr}	Towards G_{targ}
$\mathcal{G}_{10}$		0.90908	0.09092
$\mathcal{G}_{9}$	0.00000	0.91816	0.08183
$\mathcal{G}_{8}$	0.00000	0.92725	0.07275
$\mathcal{G}_{7}$	0.00001	0.93633	0.06366
$\mathcal{G}_{6}$	0.00001	0.94541	0.05457
$\mathcal{G}_{5}$	0.00002	0.95450	0.04548
$\mathcal{G}_{4}$	0.00003	0.96358	0.03639
$\mathcal{G}_{3}$	0.00004	0.97266	0.02729
$\mathcal{G}_{2}$	0.00006	0.98174	0.01820
$\mathcal{G}_{1}$	0.00007	0.99084	0.00909

Table 11. Probabilities that the TLA will retain the current grammar, or will shift to a G-Ring away from or towards the target in a linearly smooth $\mathcal{H}_{10}$ domain.

Even though there is a very high retention rate, when the TLA *does* shift, there is an extremely low probability of shifting away from the target grammar (relative to the probability of shifting towards the target grammar). By definition, the probability of a successful parse is higher as the learner approaches the target. The TLA exploits this characteristic of a strongly smooth space. Since Greediness will not allow a shift to a

candidate G-Ring unless a successful parse is obtained, and the probability of a successful parse decreases as the learner moves further from the target, the probability of a shift away from G_{targ} is extremely low. The ultimate effect of this is that the SVC and Greediness quickly propel the TLA towards the target grammar from G-Rings far away. When the TLA is in the vicinity of the target, the conservative effect of the SVC kicks in but is not overwhelming since Greediness keeps the probability of the TLA shifting towards the target greater than that of shifting away from the target. This can be seen in Figure 16 which depicts the number of sentences consumed in each G-Ring as the TLA approaches the target.

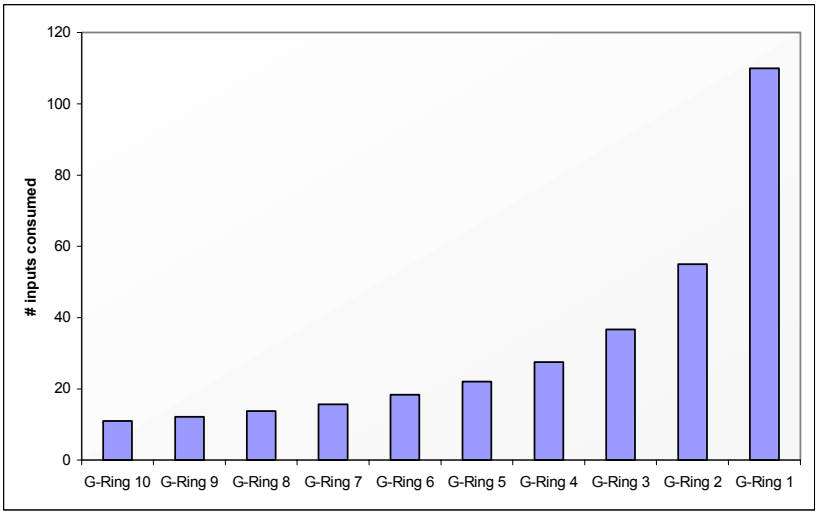


Figure 16. Number of sentences consumed by the TLA in each G-Ring of $\mathcal{H}_{10}$ in a strongly smooth domain.

In a strongly smooth domain, the TLA quickly gets near the target and remains there until it finally shifts to the target grammar.

Summary of this section: In a strongly smooth parameter space in which the probability of a successful parse is correlated linearly with the similarity of a hypothesized grammar to the target grammar, feasible learning is attained by the TLA. The relatively low number of sentences required to converge on the target is primarily due to the algorithm's Greediness constraint which biases the learner towards the target to a degree sufficient to offset the slowdown caused by the SVC as the learner nears the target grammar.

3.4.3 Accelerating Strong Smoothness

The final experiment run on TLA learning involved allowing the optimizer to freely pick the value of α (as well as λ%) for each G-Ring. The results are displayed in Table 12.

G-Ring	λ%	Avg # sentences Until convergence
$\mathcal{H}_5$	9887	53
$\mathcal{H}_{10}$	9953	264
$\mathcal{H}_{20}$	9958	1,408
$\mathcal{H}_{30}$	9999	3,658

Table 12. Optimal number of sentences required on average for the TLA to acquire G_{targ} in an accelerating smooth domain.

The optimizer consistently chose α's that accelerated the rate of a successful parse as the corresponding G-Ring approached the target G-Ring. The values for α are depicted for $\mathcal{H}_{10}$ in Figure 17 below.

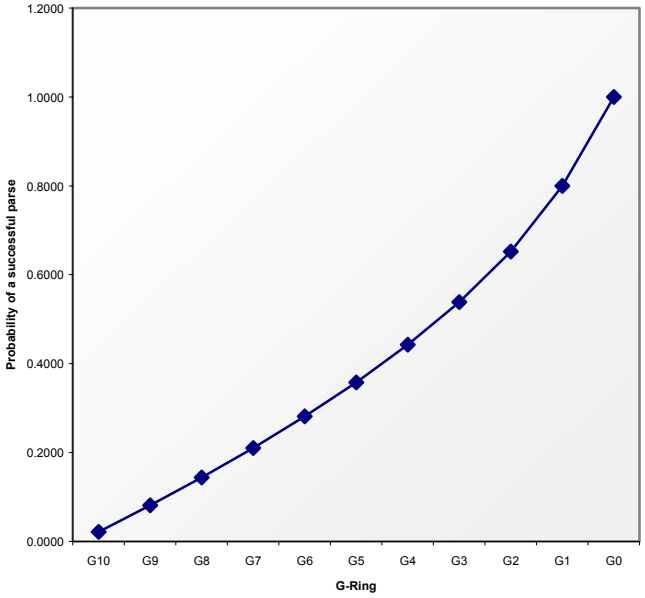


Figure 17. The values of α for each G-Ring picked by the optimizer for $\mathcal{H}_{10}$. This shape of the domain permits the most efficient TLA performance reported.

Clearly, the optimizer shaped the domain in order to increase the benefits afforded by the SVC and Greediness operating in a linearly smooth domain as described in the preceding section. The result is that TLA learning efficiency is increased by a modest amount over a linearly smooth space. This is depicted in Figure 18.

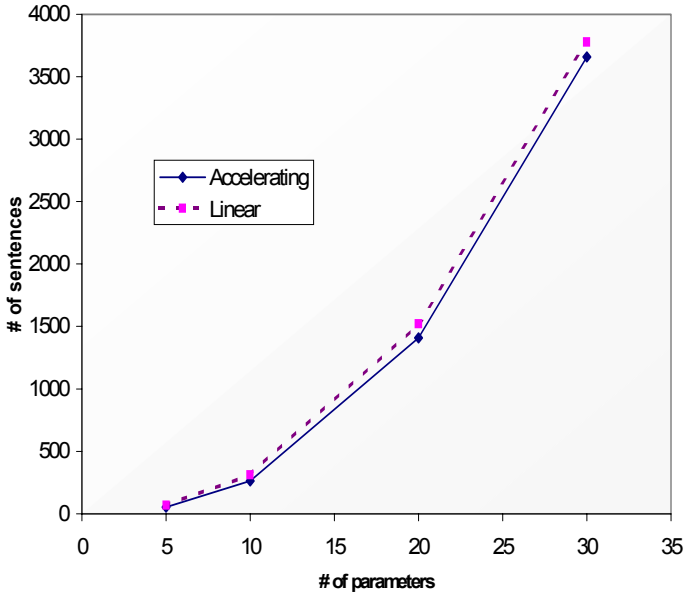


Figure 18. The number of sentences consumed by the TLA in spaces with accelerating smoothness and linear smoothness.

3.5 Discussion

The TLA is an attractively simple algorithm and by incorporating a parse test, Gibson and Wexler have made an advance on previously existing acquisition theories in two respects. First, the TLA overcomes Valian's parsing paradox (see Section 1.4.1). Second, it takes the task of recognizing usable input sentences seriously. In the classical instant-triggering model it was assumed that property detectors for each parameter were available that could deduce which parametric settings were necessary to license the current sentence based solely on the surface structure. As discussed in Section 1.4, the properties of natural languages make the task much harder than might

have been anticipated. Only through a parse test, which applies the grammar to the sentence at all levels of derivation, can parametric evidence be obtained.

But testing a grammar is work, and the TLA can only reasonably test one grammar at a time, and so it cannot scan the whole domain of grammars in order to select the one that best fits the input. The TLA's three constraints: the Error-driven constraint, Greediness and the SVC, have the effect of steering the TLA closer to or further from the target grammar, as inputs are consumed, based on characteristics of the input sample it receives. These characteristics are determined by the shape of the parametric domain in terms of which grammars are more or less likely to license the current input. For the TLA to be successful as a model of natural language acquisition, it appears that the domain being searched must adhere to a strict strong smoothness requirement – the grammars most parametrically dissimilar to the target must be most unlikely to license the current input sentence.[41] (This smoothness relationship can be linear or accelerating.)

Whether or not natural languages are strongly smooth is unknown. Chomsky has emphasized that small changes in parameter settings can have considerable effects on the languages generated, due to the rich interactions of principles, parameters and

[41] Or the number of parameters must remain low with a conducive rate of ambiguity. Very roughly, the number a child might be expected to hear between 6 and 7 million sentences (Hart and Risley, 1995). There are several areas of Table 7 that fall under this number.

lexical properties in sentence derivations. For example, he writes (Chomsky, 1988, p.63):

> "there is no simple relation between the value selected for a parameter and the consequence of this choice as it works its way through the intricate system of universal grammar. It may turn out that the change of a few parameters, or even of one, yields a language that seems to be quite different in character from the original."

If this is so, the degree of overlap of the parameter settings in two grammars (the number of parameter values they share) would *not* correlate highly with the degree of overlap of the languages they generate; the grammar/language relationship would not be a smooth one. Regardless of whether Chomsky's point establishes non-smoothness in general, it is clear that linguistically plausible parametric descriptions which delineate unsmooth domains can be easily arrived at. For example, in Gibson and Wexler's small domain of three word order parameters (see Table 2 and Table 3 from Chapter 3 above), the sentence pattern SVO is licensed by grammars that differ from each other maximally: every one of their parameters is set differently. SVO order is licensed by the grammar SV, VO and -V2, and by the grammar VS, OV and +V2 (among others).

Even if natural languages were strongly smooth, the TLA requires an excessive amount of between-language overlap in the domain (greater than 99%) between languages generated by grammars in neighboring G-Rings. For example, taking numbers displayed in Table 12 for $\mathcal{H}_{10}$, relaxing the overlap percentage from .9999 to .9000

(leaving the αs fixed in a strongly smooth fashion) increases the expected number of input sentences consumed from 312 (for .9999) to 1,181. Decreasing the percentage to .8 yields a result of 2,259, to .7 yields 3,233. These values are plotted in Figure 19 below.

To be concrete, if all languages in $\mathcal{H}_{10}$ consist of 10,000 sentences, these numbers indicate that 9,999 out of 10,000 must lie in the intersection between languages generated by grammars in neighboring G-Rings, if learning speed is to be optimal. Even more striking is that the *maximal* difference between any two languages in the domain is 10 sentences in order for learning speed to be optimal. That is, the language generated by the grammar in $\mathcal{G}_{10}$ can differ from that of a grammar in $\mathcal{G}_1$ by, at most, 10 sentences. Such a dense space of languages seems to go against linguistic fact.

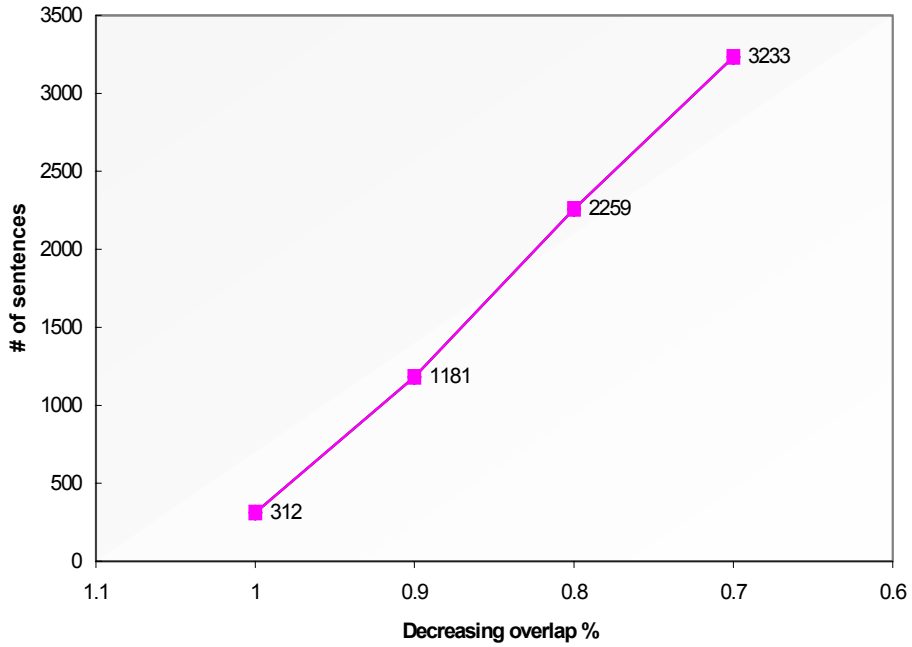


Figure 19. Sensitivity of TLA performance to the overlap between neighboring G-Rings in $\mathcal{H}_{10}$. The parameter space is linearly strongly smooth.

Although these two factors: strong smoothness and a high overlap rate, keep the domain manageable for TLA learning, there is still an exponential increase of the number of sentences that the TLA consumes with respect to the number of parameters in the domain. A high overlap rate and strong smoothness are useful in that they keep TLA performance in check for parameter spaces up to around 30. But as the number of grammars in the space approaches a truly gargantuan number, induced by only a modest increase in the number of parameters (2^{50} = 1,125,899,906,842,620), TLA performance might very well be significantly worse than the performance in the smaller domains depicted here despite the positive effect of strong smoothness and high overlap. In the next Chapter a learning model is presented that sets parameters

individually in an attempt to avoid the exponential explosion that ensues from evaluating collections of parameters as whole grammars.

4 The Structural Triggers Learner

4.1 The Parametric Principle

The TLA assigns values to parameters, but it does not incorporate the central insight of parameter theory – what we have called the *Parametric Principle – set individual parameters; do not evaluate whole grammars* (Fodor 1998, Sakas and Fodor, in press; Sakas and Fodor, 1998). This is what distinguishes a true parameter setting device from learning systems of other kinds, and it is the source of the enormous simplification of the learning task for which the principles and parameters model is renowned.

The essential point is that: if there are *n* binary parameters and the learning procedure is able to establish the value of each one independently of the others, only *n* successful learning events need occur for convergence on any one of 2^n grammars. By contrast, any learning device that evaluates grammars rather than individual parameter values faces a task that is correlated with the number of grammars – exponential with respect to the number of parameters. Gibson and Wexler originally tested the TLA on an artificially small domain of eight languages defined by three binary parameters, where the extreme difference in workload between setting parameters and selecting grammars did not become apparent. But it is clear from the results presented in Chapter 3 that if the estimate of 30 syntactic parameters for natural language is more realistic, then the

disparity is between setting 30 parameters and checking more than a billion grammars, at least for domains that meet a reasonable smoothness requirement.

A useful way to look at the difference between testing grammars and testing parameters is to see convergence as the elimination of all grammars other than the target, and to consider how effectively different procedures manage to eliminate grammars. A grammar-by-grammar test can eliminate them one at a time at best, so even if the learner kept track of the fate of every one, in the worst case it could take up to $2^n - 1$ eliminative steps to rule out all but the target. In fact, the TLA never eliminates any possible grammars at all, because it does not take the trouble to record negative outcomes of its parse tests. Presumably, this is because the slight reduction in the search space that results from eliminating individual grammars does not compensate for the cost of record-keeping on such a vast scale. Therefore all grammars remain in the pool from which the TLA selects a grammar to test, with the consequence that some incorrect grammars may be tried out many times. More importantly, the pool of potential grammars does not get any smaller as learning proceeds.

By contrast, true parameter setting permits a very rapid reduction of the pool of possible grammars. Each time a parameter is permanently set, one parameter value is eliminated. And since half of all grammars have that parameter value, that eliminates

from consideration half of the candidate grammars remaining.[42] 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. This is how the Parametric Principle makes such a great difference to the scale of the learning problem.

Very few existing learning models abide by the Parametric Principle. Statistical weighting systems such as those proposed by Valian (1994) and Kapur (1994), postpone setting a parameter for some time while evaluating the evidence, but do eventually settle on a value for each and set it permanently. The Structural Triggers Learner described later in this Chapter also is designed to obey the Parametric Principle. It is surprising that a parameter-based learning model would *not* take advantage of the reduction of the learning problem that the Parametric Principle makes possible. There are no compensating advantages to be gained by searching through the vast space of grammars. At best, clever search strategies may make it less punishing (cf. Clark 1992, Gibson and Wexler 1994, among others). It remains to be seen, of course, but at present it seems unlikely that any improvement would rival that due to the Parametric Principle.

[42] For simplicity I make the common assumption here that exactly half of the remaining grammars are eliminated by each parameter that is set, but this would not be so if there were co-occurrence restrictions on parameter values or constraints on parameter accessibility such that a parameter does not freely admit of either value in combination with all other parameter values.

The sole reason for violating the Parametric Principle, it appears, is that obeying it is too difficult. The literature on language learnability does not make this clear; the point is rarely addressed explicitly. Only Clark (1994) considers it, and he judges that the computational costs of respecting the Parametric Principle "are too great to be acceptable." If true, this is a very consequential fact. Being forced to give up the idea of 'instant' triggering does some damage to Chomsky's original conception of parameter setting, but to give up the Parametric Principle would be to abandon its whole essence.

In this chapter, I present an analysis of an approach that permits true parameter setting in accord with the Parametric Principle and demonstrate that Clark was correct to a degree – the Parametric Principle does entail an exponential computational cost in general. However, the analysis also indicates that there are specific domains in which the learner is able to achieve the target consuming a reasonable number of inputs. In these domains, the factors that contribute to the exponential cost are kept to modest levels early on in the time course of acquisition to enable successful parameter setting events to occur. As more and more parameters are successfully set, the generally exponential factors are held in check; they never reach the point of inducing a combinatorial explosion.

4.1.1 Unambiguous triggers

A parametrically-principled learner must be able to establish a parameter value with sufficient confidence to be prepared to rule out forever all grammars in which that parameter takes the opposite value. If there is no noise in the input (which is the assumption throughout this dissertation), there is no reason not to set a parameter permanently, and permanently discard its contrary value, as soon as clear evidence of its value is received. However, a stochastic learning device such as the TLA cannot do this because it does not *know* when it has received clear evidence for a parameter setting, i.e., evidence that some sentence of the target language cannot be licensed (parsed) without that value. The SVC isolates the contribution of an individual parameter value in the parse test, so the TLA knows exactly which value potentially earns support from the success of the parse. But the support is only potential, not reliable confirmation, because the sentence might have been ambiguous and the parse assigned to it might have been the wrong one. In that case the positive outcome of the parse test provides no evidence at all for that parameter value; for all the learning system can know, the sentence is equally compatible with the opposite value.

Is there any way out of this uncertainty? The uncertainty would not arise if there were no parametric ambiguity in the domain and if the learner knew that were so; but that is not realistic for natural language. Alternatively, the uncertainty due to ambiguity could be avoided by the learner if it could establish which inputs were parametrically

ambiguous and refrain from setting parameters in response to them. Learning would be based solely on unambiguous triggers.[43]

4.2 The STL Algorithm

For the TLA, a trigger is a *sentence* that causes the learner to change or flip a parameter value. In this chapter I introduce a model – *The Structural Triggers Learner* (*STL*) (Fodor, 1995; 1998) – that takes a significantly different view of the triggering process.

The surface word order of an input string provides insufficient evidence to know for certain that a particular parameter setting is correct. Recall from Sections 1.2 and 1.4.1 that the underlying order (importantly, the order relevant to uncover the correct parameter settings) can be altered by movement operations. Hence, the true parameter values are not readily apparent in the strings that comprise the input sample; only after determining which transformational operations have been applied, can the correct values be identified. This is a difficult task given that one surface structure (word string) might be derived from several (underlying) base structures; i.e. the input sentence could be ambiguous.

Fodor presents an ingenious approach that makes uses of the human parsing mechanism to detect parametric ambiguity. In her *STL* model, triggers and parameter

[43] Of course this strategy will fail as a cognitive model of language acquisition if the domain of natural languages contains no unambiguous triggers. Whether or not this is the case is an important determination that remains to be made. See Chapter 5 for discussion.

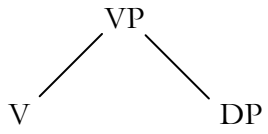
values are the kinds of things that are ingredients of grammars *and* ingredients of trees. Suitable as ingredients of grammars, they are all combined into one large grammar (termed a *supergrammar*) which the parser applies to the input in exactly the same way as any other grammar. No unusual parsing activity is needed, yet all parameter values are evaluated at once. Parameter values suitable as ingredients of trees, are detected in the parse trees output by the parser, so that the learning device is able to see which of them had contributed to parsing an input sentence and would know which to adopt.

A subtree consisting of just a few nodes and/or feature specifications serves both purposes. A trigger and the parameter value it triggers are then be 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. As trigger, a treelet is detected in the structure of input sentences. As parameter value, it is then adopted into the learner's current grammar, and is available for licensing new sentences.[44]

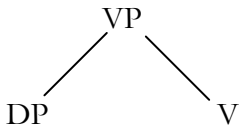
Consider an example. For the Complement-final value of the V-before-O parameter discussed in Section 1.4.1, the structural trigger (parameter value) might be a VP

[44] See Sakas and Schmeidler (in press, 1997) for a description of a computer implementation of an STL parser that can handle the small, 3-parameter domain of G&W described in Section 3.1 .

subtree with the verb preceding the object. For the Complement-initial value, the treelet would be the mirror image with the object preceding the verb. See Figure 20.



Treelet that the grammar of English contains



Treelet that the grammar of Japanese contains

Figure 20. Two structural triggers or parameter values for the V-before-O parameter. Although a determiner phrase (DP) is chosen for this example, any tree-based linguistic description of object position could have been used.

A treelet such as one of these reflects underlying order in any language with that value. Assuming that the parser's output is a surface structure tree after movement transformations have been applied, the parameter value treelets depicted in Figure 20 will reflect underlying order as long as its terminals are not constrained in any way; when the underlying structure has been transformed, either NP or V or both could be traces. Thus the 'English' parameter value treelet in Figure 20 would contribute to parsing not only *I despise decaf* but also *Decaf, I despise* with the O moved out of VP, and not only *I have some change* but also *Have you any change?* with the V moved out.

The structural triggers employed by the learner are exactly those elements, whatever they are, that UG specifies as the sources of possible cross-language variation. These structural triggers make it possible to attain the goal of efficient error-free learning.

Suppose that the STL attempts (like the TLA) to parse each input with the current grammar G_{curr} first. If that fails, it tries again with the supergrammar consisting of G_{curr} with all UG-provided triggers/parameter values folded into it (or more precisely: all those not yet definitively disconfirmed by previous learning). Unlike the TLA model, the STL makes crucial use of the structural analyses assigned to strings by the parsing routines. To say that a parse with G_{curr} fails is not to say that there is no structural output at all. The parser will build as much of the tree as G licenses before it is forced to a halt. At this point, it needs a new parameter value in order to proceed. It will then draw on the pool of additional treelets provided by UG in an attempt to find one (or more) to complete the parse tree. Thus the learning device does not attempt to directly identify the trigger treelets in input sentences. Rather, it *contributes* the triggers to the input, when parsing cannot proceed without them; a parameter value treelet must be in the target grammar if it finds that that treelet and no other can enable an otherwise blocked parse.

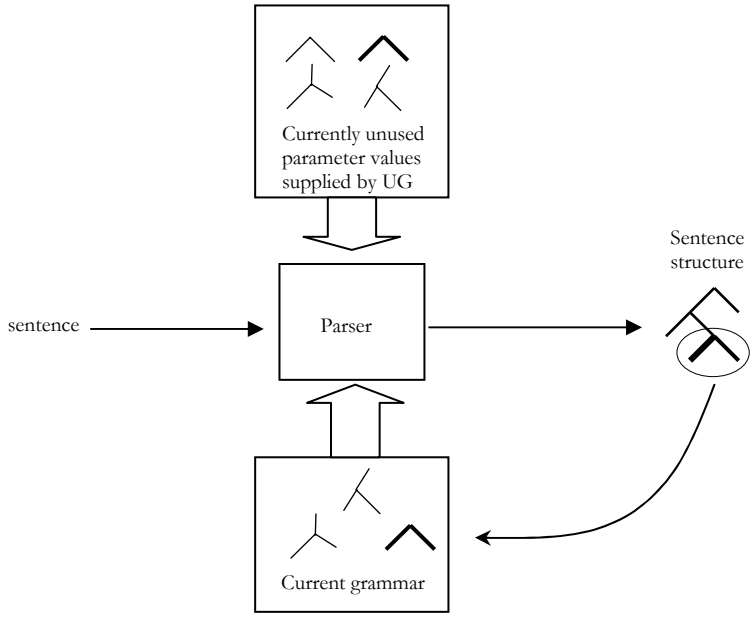


Figure 21. An example of how the STL adopts a new parameter value. The figure depicts a sentence that cannot be parsed by the current grammar but can be parsed by the supergrammar (the current grammar plus all currently unused parameter values). Assuming no choice points were encountered during the parse, the parameter value treelet that was used in the complete parse tree can be adopted into the current grammar.

But what if an input sentence is parametrically ambiguous? If it is, the supergrammar will define more than one parse tree for it. At some point in the parsing process, therefore, the parser will be faced with a choice between two (or more) analyses. So to detect parametric ambiguity, the parser needs to note when a choice point arises in parsing with the supergrammar, a point at which two (or more) analyses present themselves. If there is no such choice point, the input has just one supergrammar parse. It is parametrically unambiguous, and every parameter value present in the parse-tree is correct; the learner should adopt all of them that are not already in G_{curr}.[45]

[45] Note that there is no need in this system for an SVC-like constraint.

The same applies to any parameter values involved in the analysis of the sentence prior to a choice point in the parse, since the sentence is unambiguous up to that point. If and when a choice between alternative analyses does arise, there are several strategies the learner might adopt (cf. Fodor 1998). The one I address in this thesis is called the *waiting-STL*. I will use the terms *waiting-STL* and simply *STL* interchangeably to refer to the waiting-STL unless specifically stated otherwise.

A waiting-STL employs a type of serial parser, known as a *flagged serial parser* (Inoue and Fodor, 1995) which squares with a widely help view of adult sentence processing. When the parser notes a choice point in 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.[46]

Most importantly, the waiting-STL obeys the Parametric Principle and reaps all of its benefits (outlined above). Because the waiting-STL does not take chances, it can set parameters with confidence, so it can be confident enough to discard incorrect values, and cut the size of the subsequent learning problem in half. Note that the waiting-STL is a true parameter setting device which obtains separate evidence (in the output parse tree) for each parameter involved in the derivation of a sentence. It does not test

[46] The waiting-STL that is formalized below does not take advantage of unambiguous beginnings of sentences. It is possible that a waiting-STL that is able to use unambiguous beginnings might be somewhat more efficient.

grammars as wholes. The successive halving of the learning problem manifests itself as a reduction in the number of alternative potential parses subsequent sentences will have. The number of potential parses will be high to start with, because all the UG parameter values are available. Thus, many alternative sentence structures can potentially be derived. As more and more parameters are set, the number of alternative structures will shrink. Thus there is *progressive disambiguation* of the input as learning proceeds.

Eventually, at the point of convergence on the adult (target) grammar, all sentences will be parametrically unambiguous (the only ambiguity that remains will be any structural ambiguity inherent in the target grammar itself). For the waiting-STL, the advantage of eliminating wrong parameter values is that the proportion of sentences that are fully unambiguous parametrically will increase as learning proceeds. A sentence that was unusable for acquisition due to ambiguity early on in the learning process could become usable subsequently.

I argued in Section 4.1 that the Parametric Principle is a viable way to defeat the potential exponential complexity of the learning task. Since the STL is a true parameter-setting device which respects the Parametric Principle, this hypothesis can be can put to the test. In the next section I investigate whether the waiting-STL does indeed reduce the learning problem to one whose workload is linear in the number of parameters, as envisaged on the basis of Chomsky's development of P&P theory. The

surprising answer is that the Parametric Principle does not, in general, entail a linear increase in workload as the number of parameters increases. However, there are specific domains in which the learner is able to achieve the target consuming a reasonable number of inputs.

4.3 A probabilistic formulation of the STL

Since the STL abides by the Parametric Principle, a Markov formulation useful to analyze performance does not require states which represent the grammars of the parameter space. Instead, each state of the system will depict the number of parameters that have been set, ***t***, and the state transitions will represent the probability that the STL will adopt some number of *new* parameter values, ***w***, on the basis of the current state and whatever usable parametric information is revealed by the current input sentence.

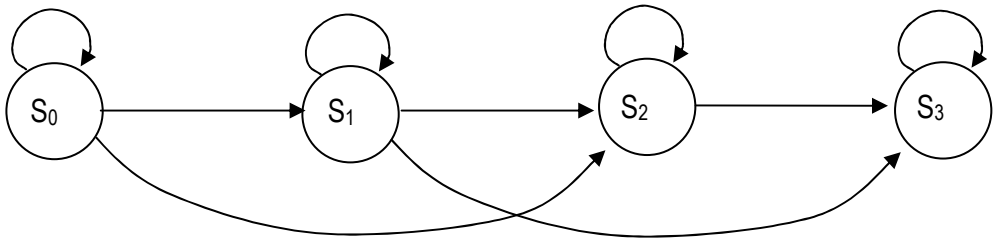


Figure 22. A possible state space for the STL performing in parameter space $\mathcal{H}_3$. Nodes represent the current number of parameters that have been correctly set and arcs represent a possible increase in the number of parameters set. Once the learner enters state 3, it has converged on the target. For the purposes of this example, each input expresses a maximum of 2 parameters. I.e. an input may express 0, 1 or 2 *new* parameters (w = 0, 1 or 2).

In other respects, the approach is identical to that used to model TLA performance: First one determines the transition probabilities of the Markov system that corresponds to the process of parameter-setting by the STL (i.e. attach transition probabilities to the arcs of Figure 22). Then one calculates the fundamental matrix, which will yield the expected number of sentences required for the learner to converge on the target (see Chapter 2 for details).[47]

The following factors (described in detail below) determine the transition probabilities:

- the number of parameters that have already been set ($\boldsymbol{t}$)
- the number of parameters *relevant* to the target language ($\boldsymbol{r}$)
- the *expression rate* ($\boldsymbol{e}$)
- the *effective expression rate* ($\boldsymbol{e'}$)

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, $\boldsymbol{r}$, is the total number of

[47] What follows is a presentation of one method for arriving at the expected size of the input sample consumed by the waiting-STL. As stated, this approach is identical to that discussed in earlier chapters. 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. See this chapter's appendix for the relevant formulae.

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.[48] The expression rate, ***e***, for a language, is the average number of parameters expressed by its input sentences. For now, I will assume that e is the same for all target language sentences. (This is surely not a linguistically realistic assumption – see section 4.4.1 for discussion and alternatives). 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 mean proportion of expressed parameters that are expressed unambiguously (i.e., $e' = (e - a)/e$).

The learner starts out in a state where no parameters are set (S_0) and then moves closer and closer to the target as more and more parameters are definitively set. The probability of the transition from one state to another is determined by the parse test of the current input.

---

[48] Of course, the parameters for relative clauses and questions will (typically) be relevant to the target language as a whole.

The learner does **not** change state if:

- the input is ambiguous, or
- all the parameters expressed by the input have already been set.

The learner **will** change state if:

- The input is unambiguous, **and**
- the input expresses parameters that have not yet been set.

The background is now in place to present a derivation of the number of inputs the STL can be expected to consume before converging on the target grammar.

Let the probability that the system will change from an arbitrary state S_t to state S_{t+w} be depicted as $P(S_t \rightarrow S_{t+w})$; this is the transition probability that the STL will move from a state in which $\boldsymbol{t}$ parameters have been set (S_t) to one in which $t+w$ parameters have been set (S_{t+w}). This probability is derived by first calculating 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. Then the probability that the STL does not have to discard that sentence because it contains an ambiguity is derived. Combining these probabilities, a formula is arrived at which yields the probability that the learner will adopt $\boldsymbol{w}$ new parameter values ($0 \leq w \leq e$) on the basis of a given input sentence, given that $\boldsymbol{t}$ parameters had already been set.

To present the derivation, it is useful to set ambiguity aside for a moment. In order to set all *r* parameters, the STL has to encounter enough batches of *e* parameter values, possibly overlapping with each other, to make up the full set of *r* parameter values that have to be established. Let *H(w|t,r,e)* be the probability that an arbitrary input sentence expresses *w* new (i.e., as yet unset) parameters, out of the *e* parameters expressed, given that the learner has already set *t* parameters (correctly), for a domain in which there are *r* total parameters that need to be set.[49]

H(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 Equation 12:

Equation 12.

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

Now, to deal with ambiguity I bring the effective rate of expression, *e′*, into play. Recall that *e′* is the proportion of expressed parameters that are expressed unambiguously.

[49] Readers familiar with basic probability theory will recognize the probability of choosing *w* of a kind, without replacement, from a batch of *e* items from *r* total items as being hypergeometrically distributed., hence the function name *H*.

Thus the probability that any single expressed parameter is expressed unambiguously is also e'. The probability that all of the expressed but as yet unset parameters are expressed unambiguously is $(e')^w$. That is, the probability that an input is effectively unambiguous and hence usable for learning is equal to $(e')^w$.

We are now in a position to give the formula for the probability $P(S_t \rightarrow S_{t+w})$, that the STL will set w additional parameters after encountering an arbitrary input sentence given: t parameters had previously been set, a total of r parameters need to be correctly set to acquire the grammar, the sentence expresses e parameters and the ratio of unambiguous expressed parameters to the total number of expressed parameters is e'. For clarity, I will sometimes write $P(w|t,r,e,e')$ to mean exactly $P(S_t \rightarrow S_{t+w})$.

For values of w other than 0, the probability of setting w new parameters is simply the probability that w of the e parameters expressed are currently unset (= $H(w|t,r,e)$), times the probability that all w parameters are unambiguously expressed (= $(e')^w$).

The probability of setting 0 parameters ($w = 0$) is the probability that all e parameters the sentence expresses have already been set (= $H(0|t,e,r)$), plus, the sum of the probabilities that the sentence expresses a certain number of parameters, all *unset* (= $\sum_{i=1}^{\min(e,r-t)} H(i|t,r,e)$), but all expressed ambiguously (= $(1 - (e')^i)$). See Equation 13 below.

Equation 13.

$$P(w|\ t,\ r,\ e,\ e') \ = \ \begin{cases} H(w|t,r,e)\ (e')^{w}\ , & 0 < w \le e \\ H(0|t,r,e)\ +\ \sum_{i=1}^{\min(e,r-t)} H(i|t,r,e)\ \ (1-(e')^{i})\ , & w = 0 \end{cases}$$

Equation 14 gives the probabilities for all possible transitions of a Markov system that models STL performance (that is, $P(S_t \rightarrow S_{t+w})$ for all $0 \le t < r,\ 0 \le w < e,\ t+w \le r$). In the next section I work through the details of a specific example of modeling the efficiency of STL learning in $\mathcal{H}_5$.

4.3.1 An Example Application: The feasibility of STL learning in $\mathcal{H}_5$.

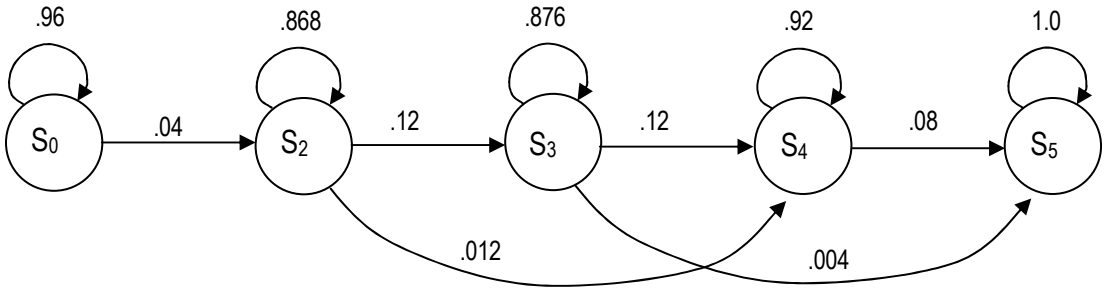


Figure 23. A Markov transition diagram for an STL learner, where $r = 5$, $e = 2$ and the ambiguity rate is 80% – on average, 1.6 parameters out of the 2 expressed, are expressed ambiguously. That is: a=1.6, $e' = 0.2$.

Figure 23depicts a transition diagram for STL learning where $r = 5$, $e = 2$ and $e' = 0.2$. Each arc represents a possible transition and the probability of each transition is derived from an application of Equation 15 in the preceding section.

For example, assume that during learning the STL depicted in Figure 23 has already set 3 parameters. After receiving an input it:

i. may not be able to set any new parameters,

ii. may be able to set one additional parameter or perhaps

iii. sets two new parameters.

The probability of the STL changing state from having set 3 parameters to having set 5 (following iii above) is $P(S_3 \rightarrow S_5) = P(2 \mid 3, 5, 2, 0.2)$. This is the probability of setting 2 new or previously unset parameters given that 3 have been set and that the input expresses 2 out of 5 relevant parameters that need to be set with a 20% chance that each parameter is expressed unambiguously. Here is the exact calculation:

$$P(S_3 \rightarrow S_5) = P(2 \mid 3, 5, 2, 0.2)$$

$$= H(2 \mid 3, 5, 2)\,(0.2)^2$$

$$= .1\,(0.04)$$

$$= 0.004$$

As a second example, the probability of shifting from state S_2 to state S_3 is:

$$P(S_2 \rightarrow S_3) = P(1 \mid 2, 5, 2, 0.2)$$

$$= H(1 \mid 2, 5, 2)\,(0.2)^1$$

$$= .6\,(0.2)$$

$$= 0.12$$

As a final example the probability that, if the learner is in state S_4, it stays there, is calculated as follows:

$$P(S_4 \rightarrow S_4) = P(0 \mid 4, 5, 2, 0.2)$$

$$= H(0|4,5,2) + \sum_{i=1}^{min(2,\,5-4=1)} H(i|4,5,2)\;(1-(0.2)^i)$$

$$= 0.6 + H(1|4,5,2)\,(1-(0.2)^1)$$

$$= 0.6 + 0.4\,(0.8)$$

$$= 0.92$$

By using Equation 13 to derive probabilities for all the transitions depicted in Figure 23 a transition matrix can be constructed.

	S_0	S_2	S_3	S_4	S_5
S_0	0.9600	0.0400			
S_2		0.8680	0.1200	0.0120	
S_3			0.8760	0.1200	0.0040
S_4				0.9200	0.0800
S_5					1.0000

The fundamental matrix is then generated from the transition matrix.

	S_0	S_2	S_3	S_4
S_0	25.0000	7.5758	7.3314	12.1334
S_2		7.5758	7.3314	12.1334
S_3			8.0645	12.0968
S_4				12.5000

The sum of the first row of the fundamental matrix yields the average number of inputs required for the STL, starting in S_0 (no parameters have been set), to enter the absorbing state S_5 (all parameters have been set). That sum here is equal to 52.0406. That is, the STL consumes approximately 52 sentences on average to set 5 parameters under the conditions specified.

There is at least one other approach that can be used for establishing the number of sentences consumed by the STL. See appendix of this chapter, Section 4.5 for details.

4.4 The Feasibility of the STL

Table 13 below presents numerical results derived by fixing different values of *r, e*, and *e′*. In order to make assessments of performance across different situations in terms of increasing rates of ambiguity, a practical measure of ambiguity, ***a′***, is employed which is directly derived from *e′*: $a' = 1 - e'$, and is presented in Table 13 as a percentage (the proportion is multiplied by 100).

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**	15	22	29	36
	40	34	46	59	73
	60	144	176	210	245
	80	3,300	3,466	3,666	3,891
10	**20**	14	18	23	28
	40	174	187	203	221
	60	9,560	9,621	9,727	9,878
	80	9,765,731	9,766,375	9,768,376	9,772,740
15	**20**	28	32	37	41
	40	2,127	2,136	2,153	2,180
	60	931,323	931,352	931,479	931,822
	80		..... over 30 billion		
20	**20**	-	87	91	95
	40	-	27,351	27,361	27,383
	60	-	90,949,470	90,949,504	90,949,728
	80	-		..... over 95 trillion	

Table 13. Average number of inputs consumed by the waiting-STL before convergence. Fixed rate of expression.

The numbers in the table exhibit some interesting patterns. First, note that the expected number of inputs needed increases quite slowly with the number of

parameters (compared to the number that the TLA requires; see Figure 15 in Chapter 3). The STL does not search the space of grammars, but sets parameters one by one systematically, in accord with the Parametric Principle. However, the impact of ambiguity is much sharper. At low ambiguity and at low expression rate, the sample size for convergence is small – even when *r* is large. But increasing ambiguity and expression raises the sample size needed to several hundreds of thousands of sentences, and then into billions, when both ambiguity and expression are high. Figure 24 displays this striking effect of ambiguity on performance. The chart represents the number of sentences that the waiting-STL can be expected to consume as a function of increasing ambiguity in a domain where *r* is fixed at 20, and *e* is fixed at 10. Even on a logarithmic scale, an accelerating increase in the size of the input sample can clearly be seen. Plots incorporating different fixed values of *r* and *e* show a similar increase.

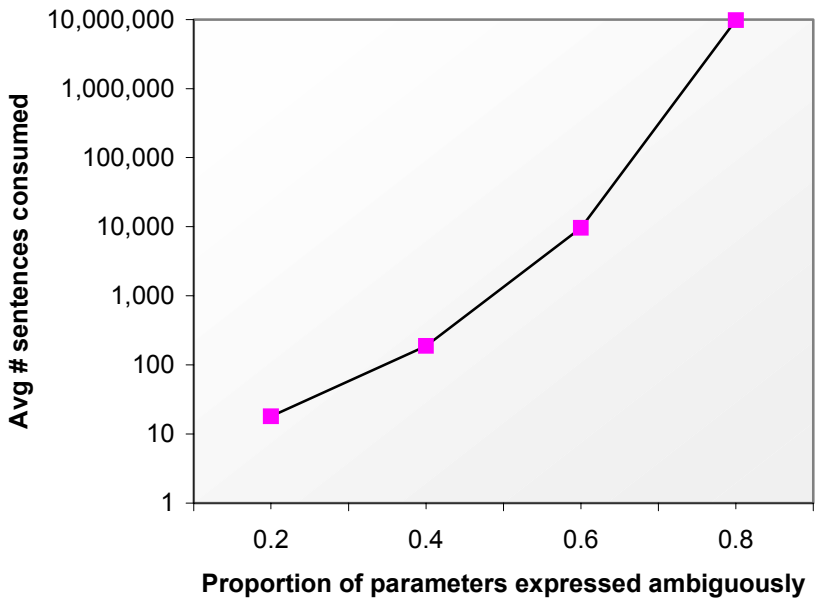


Figure 24. Performance of the waiting STL plotted on a logarithmic scale of inputs consumed as ambiguity increases. r is fixed at 20, and *e* at 10.

Not as striking, but perhaps more surprising, is the effect of ambiguity on the effect of *r*. As ambiguity increases, the rate of increase in the number of sentences consumed increases with respect to the number of parameters. This is depicted in Figure 25. The basic result it that, at a high degree of ambiguity, as the domain size scales up, the number of sentences consumed by the STL escalates at a rate that is greater than linear in the number of parameters to be set.

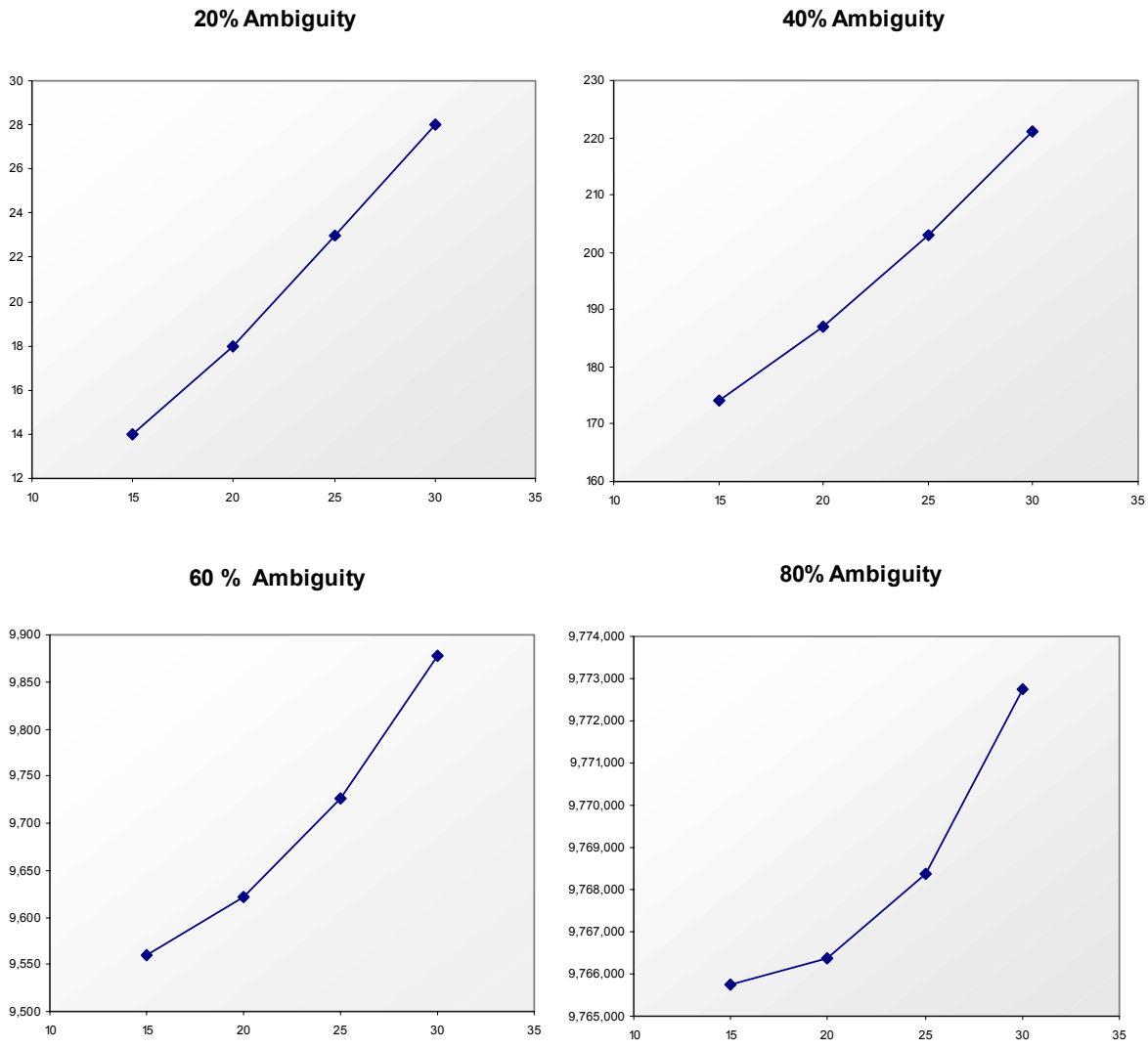


Figure 25. The effect of increasing ambiguity as domain size increases. The number of parameters increases along the x-axis. The number of sentences consumed increases along the y-axis. *e* is fixed at 10. Note that the scale of the y-axis differs in each chart.

In a highly ambiguous domain the STL discards a devastatingly large number of inputs waiting for reliable, unambiguous ones (Figure 24). The basic effect of ambiguity is understandable, although its extent is remarkably extreme. But the fact that the parametrically principled STL doesn't keep the complexity of the learning task linear in

the *number of parameters* (Figure 25) runs contrary to the primary motivation for the model – the Parametric Principle (set parameters, do not evaluate whole grammars).

Both of these results seem not to bode well for the ultimate success of the waiting-STL as a feasible model of syntax acquisition. However, as I explain in what follows, incorporating a reasonable assumption about the input sample into the probabilistic analysis, the expected performance of the STL improves significantly. But first, it is informative to discuss why the framework as presented so far leads to predicting that the STL will consume an extremely large number of sentences at rates of ambiguity and expression approaching natural language.[50]

By far the greatest amount of damage inflicted by ambiguity occurs at the very earliest stages of learning. This is because before any learning takes place, the waiting-STL must (by definition) wait for the occurrence of a sentence that is fully unambiguous. Such sentences are bound to be extremely rare if the expression rate and the degree of ambiguity is high. For instance, a sentence with 20 out of 20 parameters unambiguous will virtually never occur if parameters are ambiguous on average 99% of the time (the probability would be $(1/100)^{20}$).[51]

---

[50] See Section 1.2 for brief discussion on across language natural language ambiguity.

[51] This can be derived from Equation 2 when w = e = 20. In this case, the $(e')^{w}$ term in the waiting-STL transition formula (Equation 13) becomes $(e')^{e}$. That is, the proportion of unambiguously expressed parameters (e') is raised to the total number parameters expressed (e).

After learning gets underway, STL performance increases dramatically – the generally damaging effect of ambiguity is mitigated. Because the STL sets parameters in accordance with the parametric principle, with every successful learning event the number of parameters still to be set is decreased. Hence, the expression rate for *unset* parameters decreases as learning proceeds. And to be usable by the STL, the only parameters that need to be expressed unambiguously are those that have not yet been set. For example, if 19 parameters have already been set, and $e = r = 20$ as in the example above, the probability of encountering a usable sentence if parameters are ambiguous on average 99% of the time and the input sample consists of sentences expressing 20 parameters, is relatively high: $(1/100)^1 = 1/100$.[52]

Clearly, the probability of seeing usable inputs increases rapidly as the number of parameters that are set increases. All that's needed, therefore, is to get parameter setting started, so that it can then begin to pull the learner down into more comfortable regions of parametric expression. Once parameter setting is underway, the waiting-STL is extremely efficient.

Take as another example the situation that ensues when the ambiguity rate is 80%, e = 10 and r = 30. Table 13 indicates that 9,772,740 sentences can be expected to be consumed before all 30 parameters are correctly set. However 9,765,625 are consumed before a single parameter is set. This is because, at the outset of

[52] This can be arrived at by plugging into Equation 13: w = 1, t = 19, e = 20, and r = 20. This is equal to H(1|19,20,20) (1/100)1 = (1) (1/100).

learning, there are only two transitions that the system can make: $S_0 \rightarrow S_e$ or $S_0 \rightarrow S_0$. That is, the waiting-STL must wait for a fully unambiguous trigger before parameter setting commences, otherwise the system remains in the state where no parameters are set. When that trigger finally arrives, the STL is able to correctly set *all* of the *e* parameters that are expressed. But before that moment, the STL is discarding input after input. The exact number that are discarded can be determined by setting *e* (the number of parameters expressed) to *w* (the number of *new* parameters that the learner can be expected to set) in Equation 13 which will yield the probability of shifting out of S_0: $H\ (10, 0, 30, 10)\ (.2)^{10} = (1)\ (.2)^{10} = 0.0000001024$. Since S_0 is the starting state with only two transitions $S_0 \rightarrow S_e$ and $S_0 \rightarrow S_0$, full Markov technique need not be applied. The waiting time for a successful event, out of an event space consisting of success and failure, is equal to the reciprocal of the probability of success. In this case it is 1/0.0000001024 which is exactly 9,765,625. After the first 10 parameters have been set, learning proceeds extremely rapidly, only 17,115 more sentences, on average, are needed to set the remaining 20 parameters.

Figure 26 displays the mean number of sentences expected to be consumed by the STL in each state of the Markov system after the first fully unambiguous trigger is encountered (and causes the system to shift from S_0 to S_{10}). It should be noted that the number of sentences depicted in Figure 26 reflects an average that can be expected in the limit over all possible scenarios. For a given learner and input stream it is

indeterminate exactly how each arbitrarily encountered batch of 10 expressed parameters may lead to setting new parameters. In some cases 5 new parameters will be set, in other cases 3 or perhaps 1 or zero may be set. This explains the large peak when the learner is in state S_{10}. All possible input streams (which contain at least one fully unambiguous trigger) will eventually cause the learner to attain state S_{10} whereas different selections of sentences may cause the learner to 'pass by' states after S_{10}. (For example the learner might or might not enter state S_{17} depending on the sequence of sentences that have been encountered.)

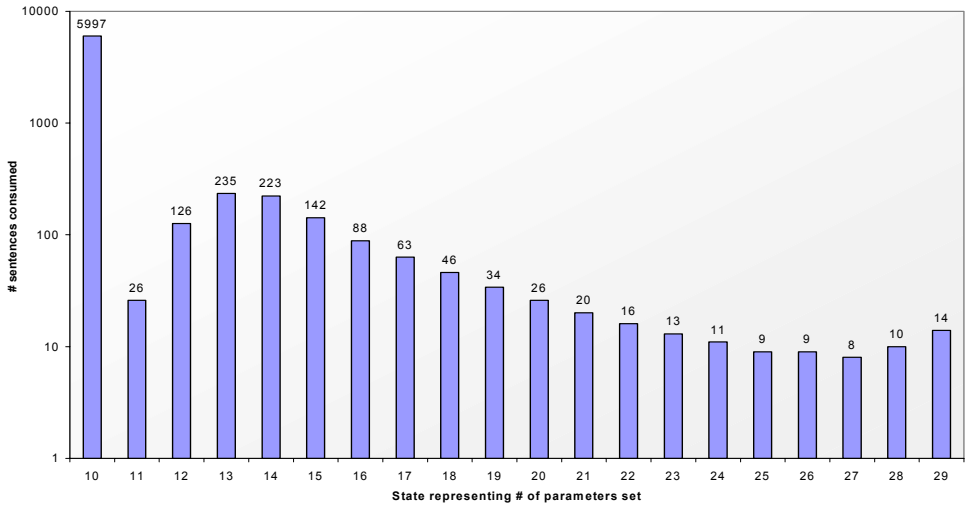


Figure 26. The logarithm of expected number of sentences consumed by the waiting-STL in each state after learning has started. $e = 10$, $r = 30$, and $e' = 0.2$.

This also explains the S-shape of the curve. H (from Equation 13) represents the distribution of the probability of seeing w new, currently unset parameters regardless of whether they are expressed ambiguously or not. The distribution of H is not uniform (it is hypergeometric, see footnote 49 above). In the beginning of learning, just after

the first fully unambiguous trigger, the probability that the STL will encounter a sentence expressing only one new parameter is significantly lower than the probability that it will encounter a sentence expressing 5 or 6. The effect of this can be seen in the peak at S_{13}. By the same argument, learning slows down near the end, just before the last few parameters are set. If there are 29 parameters set, and 30 total *to be* set, more sentences are consumed waiting for a usable sentence which expresses exactly that one last parameter value than are consumed in the middle of the time course where forward progress can be made because a potentially usable sentence (that expresses *any number* of unset parameters) is more likely to be encountered.

For the waiting-STL model of acquisition to be considered feasible, its near paralysis at the outset of learning (under the likely assumption that natural language exhibits relatively high degrees of expression and ambiguity, see Section 1.2) must be overcome. In fact, there exists a refinement of the probabilistic framework that allows for a demonstration of faster learning under what may prove to be a more realistic abstraction of true human language acquisition. It puts in place a method for incorporating distributional assumptions about the shape of the input sample encountered by the learner. When there is a uniform distribution of expression, rather than the expression being fixed, the waiting-STL is an extremely efficient learner. Of course, it remains to be determined whether or not this assumption of variable expression rates across sentences is characteristic of natural languages.

4.4.1 Overcoming the Problem of Early Ambiguity

4.4.1.1 Assume a Distribution of Expression Across the Input Sample.

So far e has, for convenience, been taken to be fixed across all sentences of the target language. In that case, when $e = 10$ the learner will have to wait for a sentence with exactly 10 unambiguously expressed parameters in order to get started on learning, and as discussed above, that can be a very long wait. But if we take the value of e to vary across sentences, (rather than fix e), then the learner will encounter some sentences which express fewer than 10 parameters, and which are correspondingly more likely to be fully unambiguous.

Any distribution of e can be applied to the framework that models STL performance. Let $\mathcal{D}_I(x)$ denote the probability distribution of expression of the input sample; that is, the probability that an arbitrarily chosen sentence from the input sample $\mathcal{I}$ expresses x parameters. For example, if $\mathcal{D}_I$ imposes a *uniform*[53] distribution, then $\mathcal{D}_I(x) = 1/e_{max}$ where every sentence expresses at least 1 parameter and e_{max} is the maximum number of parameters expressed by any sentence.

Given $\mathcal{D}_I$ the new transition probability $P'(S_t \rightarrow S_{t+w}) = P'(w \mid t,r,e_{max},e')$ is calculated as:

Equation 16
$$P'(w \mid t,r,e_{\max},e') = \sum_{i=w}^{e_{\max}} D_I(i)\ P(w \mid t,r,i,e')$$

[53] A uniform distribution is one in which all outcomes are equally likely. For example, the distribution of the outcomes of a single die roll is uniform because each possible outcome is equally likely (= 1/6).

where P is defined as in Equation 13 above and e_{max} represents a *maximum* number of parameters that a sentence may express instead of a fixed number for all sentences.

To see why Equation 16 is valid, consider that to set exactly w new parameters at least w must be expressed in the current input sentence. Also usable (to set w new parameters) are sentences that express more (than w) parameters (w+1, w+2, w+3, …, e_{max}.) because the parameters expressed by the current input sentence may overlap with the parameters that have already been set. Thus, the probability of setting w new parameters is simply the sum of the probabilities that a sentence expressing a number of parameters, i, from w to e_{max} is encountered by the STL (= $\mathcal{D}_I(i)$), times the probability that the STL can set exactly w additional parameters given i. Table 14 presents numerical results derived by fixing r and a' and allowing e to vary uniformly from 0 to e_{max}. As in Table 13 above, a practical measure of ambiguity, $\boldsymbol{a'}$, is employed which is directly derived from e': $a' = 1 - e'$, and is presented in Table 14 as a percentage (the proportion is multiplied by 100).

e_{max}	a′(%)	r=15	r=20	r=25	r=30
1	**20**	124	180	238	300
	40	166	240	318	399
	60	249	360	477	599
	80	498	720	954	1198
5	**20**	28	40	53	67
	40	46	65	86	107
	60	89	124	161	199
	80	235	324	417	511
10	**20**	17	24	32	40
	40	40	55	70	86
	60	102	137	173	209
	80	323	430	538	648
15	**20**	15	21	27	33
	40	46	62	77	93
	60	134	176	219	262
	80	447	586	726	868
20	**20**		20	26	32
	40		74	91	109
	60		223	275	327
	80		755	931	1108

Table 14. Average number of inputs consumed by the waiting-STL before convergence. Uniformly distributed rate of expression.

Clearly, the waiting-STL can be expected to feasibly attain the target consuming a reasonable number of sentences given the values for expression, relevance and ambiguity presented in Table 14. Note particularly that increasing e_{max} has far less effect than increasing e in Table 13. When $e_{max} = 20$, the mean value of e is equal to 10. Hence, the lowest block (of rows) of Table 14 can be usefully compared with the middle block of Table 13 (in which the expression rate, e, is fixed at 10). Variable expression is clearly beneficial.

However, the effect of ambiguity on learning efficiency is still substantially greater than linear. Figure 27 plots STL performance as a function of increasing ambiguity on a logarithmic scale.

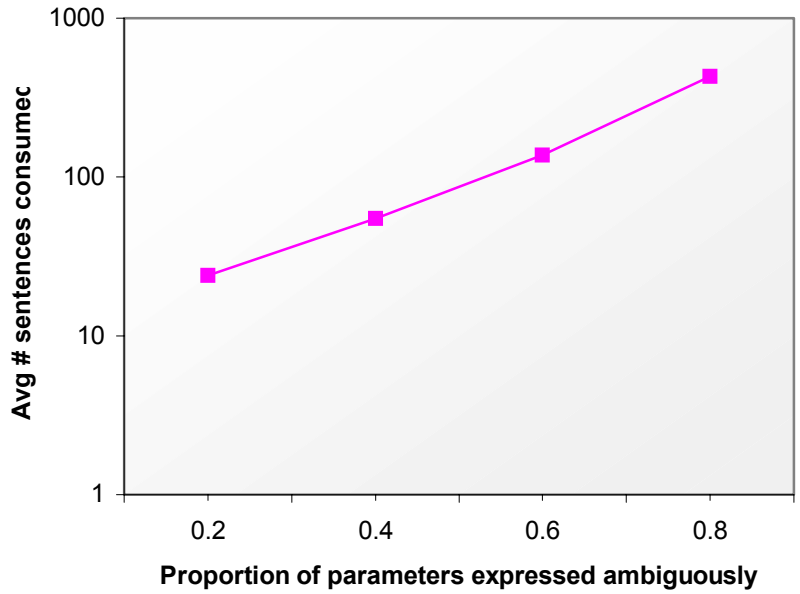


Figure 27. Performance of the waiting STL plotted on a logarithmic scale of inputs consumed as ambiguity increases. r is fixed at 20, and e is uniformly distributed from 0 to 10.

Compared with the fixed distribution consumption for the same values of expression and relevance depicted in Figure 24, a slower rate of increase is indicated, but still the increase in performance is exponential in the ambiguity rate.

Given this sharp increase, it is necessary to perform the calculations on more extreme values, in order to determine the upper bound on ambiguity that delineates a manageable input domain for the waiting-STL model.

e_{max}	a′(%)	r=30	r=40	r=50	r=60
20	90	2,767	3,656	4,549	5,446
	99.9	33,157	43,825	54,532	65,269
	99.99	337,348	445,886	554,832	664,064
	99.999	3,379,281	4,466,537	5,557,877	6,652,071
	99.9999	33,798,616	44,673,047	55,588,325	66,532,147
	99.99999	337,991,967	446,738,153	555,892,814	665,332,907

Table 15. Average number of inputs consumed by the waiting-STL before convergence. Uniformly distributed rate of expression. Note that results for r=40, r=50, and r=60 were not presented in previous tables.

Table 15 shows the average number of inputs consumed as ambiguity reaches extremely high levels when the expression rate per sentence uniformly varies from 0 to 20. Given the exponential effect of ambiguity, the model performs surprisingly well at relatively high levels of ambiguity.

For example, when there is a 99.999% chance of a parameter being ambiguous, (equivalently a probability of 1/100,000 of a parameter being expressed **un**ambiguously) and 60 parameters need to be set, the waiting-STL requires a manageable number of sentences (between 6 and 7 million) that is very roughly in line with the number a child might be expected to hear (Hart and Risley, 1995). Perhaps even more notable is that there is a linear (non-exponential) increase in the number of sentences consumed, even at a high rate of ambiguity. This is depicted in Figure 28.

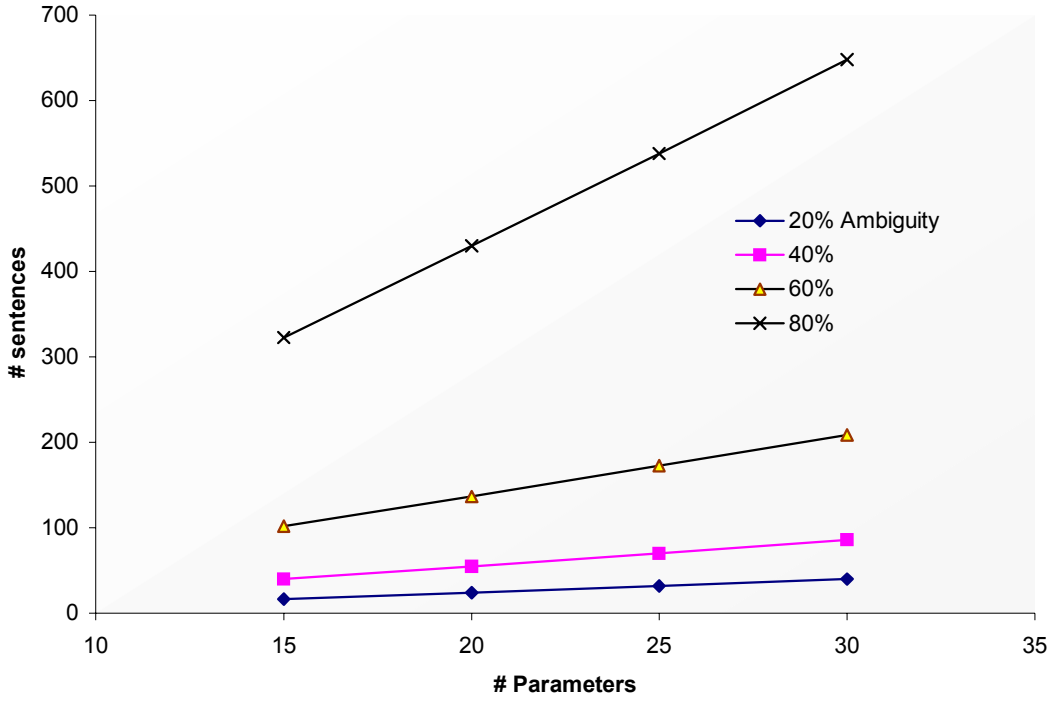


Figure 28. The effect of increasing ambiguity as domain size increases. *e* is distributed from 0 to 10.

In summary: To avoid the inefficiency due to making and correcting errors, a waiting-STL waits for fully unambiguous input to learn from. I have shown that this can result

in very slow rates of learning when parametric expression is fixed. But, this problem is not equally severe across the board. The generally damaging effect of ambiguity is absent at lower expression rates. It turns out 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 occasionally 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 an improvement in learning efficiency due to a reduction of either ambiguity rate or total number of parameters to be set. But it does seem possible, even likely, that the expression rate may vary across the sentences of natural languages. Those which express a low number of parameters should have a beneficial effect on learning speed, particularly at the early stages of learning where the degree of parametric ambiguity is at its greatest.

Whether there are such sentences that a child might encounter, which exhibit low expression rates must be determined by empirical research on child-directed language. But it seems reasonable to suppose that this is true. At least some of the 4 or 5 word sentences that early learners encounter surely do not exhibit every syntactic phenomenon in the language. 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.

4.5 Appendix

An alternative method for determining the number of sentences expected to be consumed by the STL is to directly sum the number of sentences consumed in each state. Let $E(S_i)$ represent the expected number of sentences that will be consumed in state S_i. E is given by the following recurrence relation:

$$E(S_0) = \frac{1}{(e')^e}$$

$$E(S_n) = \frac{1}{1 - P(S_n \rightarrow S_n)} \sum_{i=n-e}^{n} P(S_i \rightarrow S_n) E(S_i)$$

where $P(S_i \rightarrow S_j)$ is given by Equation 13 earlier in the chapter. The expected total is simply:

$$E_{tot} = E(S_0) + \sum_{i=e}^{r-1} E(S_i)$$

which is equal to the expected number to be consumed before any parameters have been set (= $E(S_0)$) plus the number expected to be consumed after the first successful learning event (at which point the learner will be in state S_e) summed with the number

of sentences expected to be consumed in every other state up to the state just before the target is attained (S_{r-1}).

Although not in closed form, it is easy to solve E_{tot} using dynamic programming.

To derive the formula that allows for a distribution to be placed on e, replace P by P' as given in Equation 16.

5 Discussion and Directions

The results presented in the dissertation provide indirect evidence that for two very different models, learning performance differs widely in domains with different distributions of ambiguity. Two factors emerge that exponentially affect the performance of syntax acquisition:

- the size of the parameter space, and
- the degree of parametric ambiguity[54]

For the TLA the size of the parameter space poses a major hurdle in the case that ambiguity is distributed evenly throughout the domain – the TLA performs notably worse than a learner that blindly guesses grammars at random, even at rates of ambiguity most favorable to TLA performance. The number of sentences required to converge on the target is hence greater than 2^n where n equals the number of parameters that need to be set. However, when the ambiguity is distributed so that a smooth domain ensues (i.e. the probability of a successful parse is not uniform, it is higher the closer a grammar is to the target grammar), the damaging effect of ambiguity is curtailed. At least as long as the number of parameters is modest, the TLA can acquire the target grammar consuming a reasonable number of input sentences.

[54] Including the effect of expression rate on ambiguity.

Two potential concerns are not alleviated by this result. First, although the absolute number of sentences consumed by the TLA is reasonable, an exponential trend is exhibited. If the number of parameters that is required to accurately describe human language is greater than 30 (the largest domain tested in the dissertation), further research would be needed to determine if the number exceeds the point at which an exponential explosion occurs. It could be, for instance, that the TLA is a feasible model in smooth domains of up to 39 parameters, but a clearly infeasible one in a domain of 40. Second, the result (for strongly smooth domains with 30 or fewer parameters) is contingent on there being a language domain in which languages that are parametrically similar share an extremely large number of sentence types. That is, there needs to be a high degree of overlap between the languages generated by neighboring grammars in the grammar space. Again, if the TLA is to be considered a feasible model, future research would be required to determine if natural languages conform to the experimental boundaries within which efficient TLA performance was observed. In respect to language overlap, however, this seems unlikely.

For the waiting-STL, the Parametric Principle prevents the effect of the domain size from dominating performance. However, a high degree of parametric ambiguity is crippling. That is, as the rate of ambiguity increases, the STL requires an exponentially increasing number of sentences in order to attain the target grammar. As the degree of cross-language ambiguity in natural language rises above a modest amount, the number of sentences consumed by the STL rapidly escalates at an unmanageable rate.

However, as for the TLA, there is a mitigating circumstance in which the STL can be expected to consume a reasonable number of inputs. When the number of parameters that are expressed per sentence can vary across the sentences of the input sample so that there are some sentences with little or no ambiguity, there is a striking improvement in STL performance. So, even at extremely high rates of ambiguity the STL can be considered a feasible learning model.

Still, as with the TLA, there are several potential concerns. First, it may be that the degree of parametric ambiguity in human languages approaches 100%. That is, there may be very few (if any) sentences that belong to the target language and the target language alone. If this is indeed the case, the STL is effectively paralyzed at the outset; the first parameters would be nearly impossible to set. Second, although varying parametric expression rate across sentences allows the Parametric Principle to keep the effect of the domain size in check (relative to the effect of domain size on TLA performance), there is still an accelerating increase in the number of sentences that are consumed in domains where the expression rate is fixed and the ambiguity rate is high. If psycholinguistic research reveals that this is indeed the case – that in human language, the number of parameters expressed fluctuates by very little and the ambiguity rate is high – the study presented here would have to be tuned to match the psycholinguistic data in order to determine if the STL is feasible under these conditions.

A twofold picture can be constructed from the STL and TLA feasibility studies. First, given the wide breadth of factors that create hospitable learning environments for the different learners (e.g. smoothness for the TLA, varying expression for the STL) it seems very unlikely that any one parameter setting algorithm could be devised that performs well in all possible domains. In other words, the evidence suggests that there is no acquisition model that exhibits superior performance across a broad range of possible learning scenarios. I call this the ***No Best Strategy Conjecture: No learning strategy is generally the most effective for setting syntactic parameters across all ambiguous domains.***[55]

Second, the effect of ambiguity on learning performance is *extremely sensitive to the* ***way*** *in which the ambiguity emerges*. Any computational framework that is employed to simulate language acquisition requires that certain variables (e.g. α, λ, *a*, *e*, etc.) be in place that define the input sample presented to the learner. The precise formulation of these variables shapes the distribution of ambiguity, which in turn affects learning performance. Certain distributions prevent feasible learning while others allow it. Within the class of all possible learning situations, both the STL and the TLA have *sweet spots* - where the *shape of ambiguity* is favorable to learning performance. It is noteworthy

[55] This follows the spirit of Schaffer's Law of Conservation of Generalization (1994) which applies to classification learners. Roughly, the result indicates that identifying the correct category of a datum that the learner has not been trained on (by generalizing from other data) is a "zero-sum enterprise" — for every learner, positive performance in some class of learning situations is offset by equally poor performance in others. Of course, the No Best Strategy Conjecture is concerned with efficiency, not accuracy.

that these situations are narrowly circumscribed; small changes in the relevant variables result in large changes in performance outcomes.

Together, this sensitivity and the No Best Strategy Conjecture strongly suggest that knowledge of successful performance in one artificial domain can not be used to argue for an algorithm's viability as a model of true natural language acquisition. Attempts at discovering the mechanism of human language acquisition through computational modeling must be coupled with a detailed analysis of the shape of ambiguity in input samples typically encountered by children. The results presented here contribute to a growing body of research indicating that parameter setting is a difficult and subtle enterprise. They also underline the fact that whether a particular acquisition model ultimately succeeds or fails will depend on the exact conditions under which the model performs well and the extent to which those favorable conditions are in line with the facts of human language.

References

Angluin, D. (1980) Inductive inference of formal languages from positive data, *Information and Control* 45: 117-135.

Berwick, R.C. and Niyogi, P. (1996) Learning from Triggers. *Linguistic Inquiry*, 27(2), 605-622.

Bertolo, S., Broihier, K., Gibson, E., and Wexler, K. (1997) Cue-based learners in parametric language systems: Application of general results to a recently proposed learning algorithm based on unambiguous 'superparsing'. In M. G. Shafto and P. Langley (eds.) *Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society*, Lawrence Erlbaum Associates, Mahwah, NJ.

Brent, M. R. (1996) Computational Approaches to Language Acquisition, MIT Press, Cambridge, MA.

Brill, E. (1993) *A Corpus-Based Approach to Language Learning*. Ph.D. Thesis, University of Pennsylvania.

Bresnan, Joan. (1998) Optimal syntax. To appear in *Optimality Theory: Phonology, Syntax, and Acquisition*, Joost Dekkers, Frank van der Leeuw and Jeroen van de Weijer (eds). Oxford University Press, Oxford, UK.

Brown, R., and C. Hanlon. (1970). Derivational complexity and order of acquisition in child speech. In J. R. Hayes, (ed.) *Cognition and the Development of Language.* Wiley, New York..

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

Chomsky, N. (1957) *Syntactic Structures*, The Hague, Mouton.

Chomsky, N. (1965) *Aspects of the Theory of Syntax*, MIT Press, Cambridge, Mass.

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

Chomsky, N. (1988) *Generative Grammar: Its Basis, Development and Prospects. Studies in English Linguistics and Literature*, Special Issue, Kyoto University of Foreign Studies.

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

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

Clark, R. (1994) Finitude, boundedness and complexity. In B. Lust, G. Hermon and J. Kornfilt (eds.) *Syntactic Theory and First Language Acquisition: Cross-linguistic Perspectives.* Vol. 2, *Binding, dependencies, and learnability*. Lawrence Erlbaum, Hillsdale, NJ.

Clark, R. (in press) Information Theory, Complexity, and Linguistic Descriptions. To appear in S. Bertolo (ed.) *Parametric Linguistics and Learnability: A Self-contained Tutorial for Linguists*, Cambridge University Press, Cambridge, UK.

Clark, R. (1994) Finitude, boundedness and complexity. In B. Lust, G. Hermon and J. Kornfilt (eds.) *Syntactic Theory and First Language Acquisition: Cross-linguistic Perspectives*. Vol. 2, *Binding, dependencies, and learnability*. Lawrence Erlbaum, Hillsdale, NJ.

de Marcken, C. (1996) *Unsupervised Language Acquisition*. Ph.D. Ddssertation, MIT.

Elman, J. L. (1995). Language as a dynamical system. In R.F. Port & T. van Gelder (eds.), *Mind as Motion: Explorations in the Dynamics of Cognition*. MIT Press, MA.

Elman, J. L., Bates, E. Johnson, M. A., Karmiloff-Smith, A. Parisi, D. and Plunkett. K. (1996). *Rethinking Innateness: A Connectionist Perspective on Development*. MIT Press, Cambridge, MA.

Feldman, J.A., Lakoff, G., Stolcke, A., and Weber, S.H (1990) Miniature Language Acquisition: A touchstone for cognitive science, Technical Report TR-90-009, International Computer Science Institute, Berkeley.

Finch, S. (1993) *Finding Structure in Language*. Ph.D. dissertion, University of Edinburgh.

Fodor, J. D. (1995) Fewer but better triggers. *CUNYForum* 19, 39-64.

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

Frank, R. and Kapur, S. (1996) On the Use of Triggers in Parameter Setting. *Linguistic Inquiry* 27(2), 623-660.

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

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

Grimshaw, J. (1997). Projection, Heads, and Optimality. *Linguistic Inquiry* 28(3), 373-422.

Hart, B. and Risley, T. (1995) Meaningful Differences in the everyday experience of young American children. Paul H. Brookes, Baltimore.

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

Jackendoff, R. (1977) *X-Bar Syntax*. MIT Press, Cambridge, MA.

Jurafsky, D. (1996) A Probabilistic Model of Lexical and Syntactic Access and Disambiguation. *Cognitive Science* 20: 137-194.

Kapur, S. (1994) Some applications of formal learning theory results to natural language acquisition. In B. Lust, G. Hermon and J. Kornfilt (eds.) *Syntactic Theory and First Language Acquisition: Cross-linguistic* Perspectives. Vol. 2, *Binding, dependencies, and learnability*. Lawrence Erlbaum, Hillsdale, NJ.

Kremer, S. (1996). *A Theory of Grammatical Induction in the Connectionist Paradigm*. Ph.D. dissertation, University of Alberta.

L.S. Lasdon, A. Waren, A. Jain and M. Ratner. (1978) Design and Testing of a Generalized Reduced Gradient Code for Nonlinear Programming. *ACM Transactions on Mathematical Software 4:1.*

Marcus, G. F. (1993). Negative evidence in language acquisition. *Cognition* 46: 53-85.

Marcus, G.F. (in press) *The Algebraic Mind.* MIT Press, Cambridge, MA.

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

Pearlmutter, N. J., Daugherty, K., MacDonald, M. C. & Seidenberg, M. S. (1994) Modeling the use of frequency and contextual biases in sentence processing. *Proceedings of the 16th Annual Conference of the Cognitive Science Society*: 699-704.

Pinker, S. (1979) Formal Models of Language Learning. *Cognition* 7: 217 - 283.

Pullum, G. (1996) Learnability, hyperlearning, and the poverty of the stimulus. In J. Johnson, M. L. Juge, and J. L. Moxley (eds.) *Proceedings of the Twenty-second Annual Meeting: General Session and Parasession on the Role of Learnability in Grammatical Theory* : 498 - 513. Berkeley Linguistics Society, Berkeley, CA.

Prince, A. and Smolensky, P. (1993) Optimality Theory: Constraint Interaction in Generative Grammar. RuCCS Technical Report 2, Rutgers Center for Cognitive Science, Rutgers University, Piscataway, NJ, and Department of Computer Science, University of Colorado at Boulder. To appear, *Linguistic Inquiry Monographs*, MIT Press, Cambridge, MA.

Reiger, T. (1996) *The Human Semantic Potential: Spatial Language and Constrained Connectionism*, MIT Press, Cambridge, MA.

Roberts, I. (in press) Language change and learnability. To appear in S. Bertolo (ed.) *Parametric Linguistics and Learnability: A Self-contained Tutorial for Linguists*, Cambridge University Press, Cambridge, UK.

Sakas, W. G. and Fodor, J. D. (1997) Triggering, Hill-Climbing and the Conservative Learner: Can a stochastic trigger-based learner afford Greediness as a constraint?. Paper presented at the Conference on Computational Psycholinguistics. Berkeley, California, August 10-12, 1997.

Sakas, W. G. and Schmeidler, S. (1997) Parsing triggers: A parameter-based algorithm. The Tenth Annual CUNY Conference on Human Sentence Processing, University of Southern California, Santa Monica, March 20–22.

Sakas, W. G. and Fodor, J. D. (1998) Setting the first few syntactic parameters: A computational analysis. *Proceedings of the Twentieth Annual Conference of the Cognitive Science Society* Lawrence Erlbaum Associates, Hillsdale, NJ. Sakas and Fodor, 1997.

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

Sakas, W. G. and Schmeidler, S. (in press) Parsing triggers: A multi-lingual parameter-based mechanism for determining sentence structure. CUNY*Forum,* 20. New York: City University of New York Graduate Center.

Schaffer, C. (1994) A conservation law for generalization performance. In Proceedings of the 1994 International Conference on Machine Learning Morgan Kaufmann, Schaffer, C. 1994. A conservation law for generalization performance. *Proceedings of the 1994 International Conference on Machine Learning*. Morgan Kaufmann, San Mateo, Ca.

Schunn, C. D., Crowley, K., & Okada, T. (1998). The growth of multidisciplinarity in the Cognitive Science Society. *Cognitive Science*, 22.1: 107-130.

Tesar, B. and Smolensky, P. (1998) Learnability in optimality theory. *Linguistic Inquiry* 29.2, 229-268.

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

Valian, V. (1994). Children's postulation of null subjects: Parameter setting and language acquisition. In B. Lust, G. Hermon, & J. Kornfilt (eds.) *Syntactic theory and first language acquisition: Cross-linguistic perspectives. Vol. 2: Binding, dependencies, and learnabilitY*: 273-286). Erlbaum, Hillsdale, NJ.

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

Yang, C.D. (1999) A selectionist theory of language acquisition. *Proceedings of the 37th Annual Meeting of the ACL*, Morgan Kaufmann, Orlando, FL