**Carnegie Mellon University**

# A Non-Deterministic, Success-Driven Model of Parameter Setting in Language Acquisition

**Eric Harold Nyberg III**

A Dissertation

Submitted to the Graduate School

in Partial Fulfillment of the Requirements

for the Degree

Doctor of Philosophy

in

Computational Linguistics

Department of Philosophy

Carnegie Mellon University

Pittsburgh, Pennsylvania 15213

May 1992

Thesis Committee:
Jaime Carbonell (chairman), School of Computer Science, Carnegie Mellon University
Robin Clark, Department of Linguistics, University of Geneva
Dan Everett, Department of Linguistics, University of Pittsburgh
Kevin Kelly, Department of Philosophy, Carnegie Mellon University

# Contents

# List of Figures

# Chapter 1

# Introduction

"Languages are the best mirror of the human mind." – Leibniz

Researchers in linguistics and psychology have long been interested in the empirical facts of language use as a source of insight into the processes of the mind. The process of language learning has been a particular source of fascination, since the adult state of the speaker seems vastly underdetermined by the available data. Somehow the language learner acquires precisely the grammar of his native language just by listening to the words and sentences spoken around him, and without benefit of direct instruction or reliable correction. In this sense, learning language would seem much more difficult than the acquisition of other cognitive skills (like mathematics or chess), for which there exist axiomatic descriptions and explicit, effective curriculi. Nevertheless, all children achieve mastery of their native language in a short time span, regardless of their ability to acquire other cognitive skills.

It is difficult to discern the states of knowledge that exist within the mind as language is processed and understood. It has been the goal of generative linguistics to hypothesize what knowledge structures exist in the adult mind in order to explain observed linguistic phenomena. There have been analogous efforts to explain the developing states of knowledge in the mind of the child learning language; the primary goal of this thesis is to show how a computer program can be used to simulate the learning process, and that a computational simulation of the learning process can be a valuable source of insight in theory construction and refinement.

This thesis adopts a particular language learning paradigm, called the *principles and parameters* approach (Chomsky, 1981, 1985), which attempts to capture what is invariant across all languages and what must be learned for each language. I summarize the basic assumptions of this paradigm and mention previous work in this area (in a broad way) in Section 1.1.

Only finite sets of languages can be hypothesized in the parameter setting framework, and it is both feasible and interesting to implement parameter-setting models using computational systems. Since it is possible to test the resulting systems on a large variety of data, building and testing a computer implementation of a model can help to support or refute hypotheses about the nature of language learning, and shed light on areas that are still poorly understood. I give a brief overview of the basic elements of parameter-setting learning programs in Section 1.2.

Optimal programs for solving various learning problems are usually judged by criteria such as correctness, efficiency, and robustness. Interestingly, the problem of language acquisition imposes an additional set of constraints for judging the success of a learning program, since a cognitive model of language development must also exhibit certain well-known characteristics associated with child development. In Section 1.3, I discuss the cognitive criteria that have been proposed by Pinker (1979) and show how they relate to the evaluation of parameter-setting models.

In addition to the present work, there have been two recent efforts to construct computational models of parameter setting (Clark, 1990a; Dresher and Kaye, 1990). I summarize and discuss these models in Section 1.4.

In Section 1.5 I focus on the particular problems to be addressed in this thesis, thus defining the scope of the present work.

## 1.1 The Parameter-Setting Paradigm

There has been a trend in the theory of grammar away from the representation of grammar as an exhaustive list of phrase structure rules and towards a representation of grammar as a core set of structural principles plus a lexicon. This tendency reflects the notion that the study of grammar should be cross-linguistic in origin, and that the eventual goal of the study of grammar should be a description of the grammatical principles that hold in all languages, with some explanation of how those principles differ in their embodiment from language to language. Such cross-linguistic grammatical principles are often referred to as *universal grammar*, or UG (Chomsky, 1985).

In a paper entitled "Principles and Parameters in Syntactic Theory" (1981), Chomsky introduced the notion that the basic framework of language, shared across all languages, could be captured as a set of basic principles or axioms, each of which might vary slightly in predictable ways from language to language. The notion of a *parameter* was developed to capture the variation in a grammatical principle; the values of a particular parameter control the behavior of a principle from language to language. It is not the goal of this thesis to argue pro or con the various particulars of Chomsky's model of grammar, but rather to investigate the general problem of learning within such a framework.

For example, consider the simple parameterized system proposed in (Nyberg, 1987). The basic word orders SVO, SOV, VOS and OVS are derived by two parameters, one that controls the principle of Argument Attachment and one that controls the principle of Specifier Attachment (Gibson, 1987). If **P1** has the value `:left`, then arguments attach to the left of the verb; if it has the value `:right`, then arguments attach to the right of the verb. The parameter **P2** controls Specifier Attachment in a similar fashion, dictating whether the subject of a phrase appears to its left or its right. It is a straightforward task to derive the four possible constituent orders from the values of **P1** and **P2** (see Figure 1.1)[1].

| **P1** | **P2** | **Word Order** |
|---|---|---|
| `:left` | `:left` | `SOV` |
| `:left` | `:right` | `SVO` |
| `:right` | `:left` | `OVS` |
| `:right` | `:right` | `VOS` |

Figure 1.1: **Example: Word Order Parameters**

The goal of the learner would be to choose the correct values of these two parameters for the language being learned. For example, if sentences from a VOS language were presented to the learner, we would expect the learner to choose `:right` as the value for both parameters. Linguists generally assume that some sentences act as "triggering data" which cause the learner to reset one or more parameter values when they appear in the learner's input. For example, a sentence like *Ate the rocks Ted* could act as a triggering example for word order VOS, since the verb appears at the beginning of the sentence.[2]

In the subsections that follow, I mention some linguistic research on principles and parameters in the domains of syntax and phonology. I also discuss in more detail some of the underlying assumptions within the paradigm which concern both the input to the learner and constraints upon hypotheses made by the learner.

[1]There are certainly other surface constituent orders, which derive not from these attachment parameters but from processes of movement; see (Nyberg, 1987) for more details.

[2]The role of triggering data in learning is crucial, especially with respect to certain problems in learning (e.g., recovering from overgeneralization, learning in noisy environments); cf. Section 1.1.4 and Chapter 5.

### 1.1.1 Principles and Parameters in Syntax

Several researchers have proposed parameterized accounts of adult language variation in different (and often quite narrow) areas of syntax:

- *Parameters and the Binding Theory*. Much suggestive work has been done on cross-linguistic variation in binding theory, particularly with respect to the bounding domain for anaphors and pronouns and the presence/absence of long-distance anaphors (such as Icelandic *sig* and Japanese *zibun*). Work reported in the literature includes that done by Wexler and his colleagues (Wexler and Manzini, 1987; Borer and Wexler, 1987; Wexler, 1987), Solan (1987) and Lust (1986a; 1986b). Binding theory parameters are of particular interest because they allow successively more general binding domains for anaphors and pronouns, hence producing a series of languages that are supersets of the previous language. This creates a situation that has provided linguistic motivation for investigation of the *Subset Principle*, a constraint on hypotheses that prevents the learner from overgeneralizing in a situation where the data will not allow retraction of an overly general, incorrect hypothesis (Angluin, 1978; Berwick, 1985; Wexler and Manzini, 1987).[3]

- *Parameters and Word Order*. There has been some work on how surface word order results from deeper principles of syntax, and how these principles might be parameterized to account for cross-linguistic variation in surface word order. Among those who have investigated word order parameters are Travis (1984), Koopman (1983), and myself in previous work (Nyberg, 1987). Parameters that account for word order are also of interest from a computational point of view, because they do not generate subset/superset languages as their values vary. This type of non-subset parameter can pose particular problems when combined together with subset-type parameters (such as those associated with the binding theory) in a single learning problem (Clark, 1988).

[3]The Subset Principle will be described more fully in Section 1.1.4.

- *Parameters and Null Subject Phenomena.* Another line of research has been concerned with the presence of null subject sentences in child language, and the restructuring that occurs in the child grammar in order to distinguish between true null subject languages (such as Italian) and non-null subject languages (such as English) (Hyams, 1986; Kazman, 1988).

- *Other Types of Parameters.* Other work on parameterized accounts of syntactic variation include investigation of the nature of empty categories (Nishigauchi and Roeper, 1987) and parameters and second language acquisition (Flynn, 1987; Phinney, 1987; Archibald, 1990).

Although many syntactic parameters and parameter values have been developed, few researchers have considered the concrete problem of how their parameters are actually set by a child learning language, presumably in response to some triggering data. Although most researchers are careful to note what might constitute adequate triggering data, only a handful have discussed the problems that might arise in a learning system that actually sets parameters in response to data. The work of Ken Wexler and his colleagues has focussed in some part on the development of constraints and principles that must be satisfied by grammatical systems to ensure learnability (Wexler and Culicover, 1980; Wexler and Manzini, 1987; Borer and Wexler, 1987).

### 1.1.2 Principles and Parameters in Phonology

Although work in generative phonology in the last decade has been much concerned with cross-linguistic variation (with some efforts to provide parameterized accounts), there have been only two fairly comprehensive parameterized models in phonology, both of them models of stress rule acquisition:

- *Halle and Vergnaud.* In *An Essay on Stress* (1987), Halle and Vergnaud propose a parameterized system for metrical phonology that builds metrical constituent structures and associates stress patterns with words and phrases depending on the values of a small set of parameters. Their work is concerned not only with the formal specification of such a system, but also in testing the coverage of their system against a wide variety of data from different languages. However, they have not attempted to build a computational implementation of their system.

- *Dresher and Kaye.* Dresher and Kaye have proposed a parameterized account of stress assignment, and they have also attempted to build a computer model of stress learning that sets parameters and can learn the stress rules of different languages (Dresher and Kaye, 1990). It remains an open question whether their system is more comprehensive in terms of coverage than that of Halle and Vergnaud.[4]

Although it is not possible to review all facets of the work that has been done in the parameter-setting paradigm, it is necessary to review three areas of inquiry that are crucial to the formulation of the model presented in this thesis: the nature of the learner's input data, the Subset Principle, and the theory of markedness.

### 1.1.3 The Nature of the Input Data

In general, there are at least two kinds of evidence that may be presented to or used by a learning program: positive evidence and negative evidence. Assume that the language to be learned is $L$. Sentences $s \in L$ are *positive evidence* for $L$; sentences $s \notin L$ are *negative evidence* for $L$. The learner may or may not know for each $s$ that it encounters whether $s$ is a piece of positive or negative evidence. The language learning environment will not only contain positive evidence (grammatical sentences from the language to be learned), but also negative evidence (ungrammatical sentences, performance errors, etc.). Since sentences encountered by the child learner do not carry a "tag" indicating whether they are positive or negative evidence, the learner must somehow determine which examples are grammatical and which are to be filtered out.

[4]The Dresher and Kaye model will be discussed in detail in Section 1.4.1 and Chapter 3.

In addition, we must consider the effect of the child's own utterances as a source of feedback during learning. Since the child's utterances at a given stage in learning presumably reflect the current state of his or her grammar, feedback about the grammaticality of his utterances could provide crucial information to the learner. However, it has been widely assumed in the field of language acquisition that the child has no consistent access to information about the correctness of his early utterances. At best, the child receives sporadic but useful corrections, and at worst, corrections made by adults can actually be misleading (Brown, et al., 1969; Brown and Hanlon, 1970; Newport, et al., 1977)[5].

I assume that the child has no way of detecting negative evidence (ungrammatical examples), and no reliable access to correction. Given the circumstances, the simplest hypothesis is that all inputs are assumed to be grammatical by the learner. Of course, the actual data presented to the learner will contain a measure of ungrammatical or "noisy" examples. An algorithm that can learn under these conditions will most likely also be able to learn given some access to correction or negative examples; but the reverse is not necessarily the case (that a learner which can learn with correction and negative examples can learn without them). I will make the weaker assumption about the learner's knowledge concerning the input data, namely, that the learner cannot access negative evidence, and that there is a possibility of ungrammatical examples in the input.

An additional learning strategy that can be used by a learner is to make use of *indirect negative evidence*. Consider the case where a learner has encountered several thousand examples, and not once has it encountered an example $s$ of a certain type derived by a particular value of some parameter (for example, a sentence containing a

[5]Although some children may have the benefit of direct feedback on their utterances in the form of adult correction, it is clear that not all children have this advantage, and that a model that learns without access to correction is a stronger model than one that requires correction in order to learn. As pointed out by Jaime Carbonell (personal communication), machine learning systems that can test hypotheses (on the external environment or by asking an oracle) converge much faster in the learning process (Gross, 1989). It is left to future work to consider how a parameterized system could enhance its learning performance by receiving feedback on specific inputs.

long-distance anaphor, which would indicate a certain value of the anaphoric bounding domain parameter (Wexler and Manzini, 1987)). If after a period of time the system concludes that it will *never* encounter such an example, then it has made use of indirect negative evidence. It is important to note that the use of indirect negative evidence requires some notion of counting or a threshold (i.e., how long should the learner wait before deciding $s$ will never appear?), and is therefore prone to horizon effects (e.g., $s$ may appear just a few examples after the learner decided that it never would). In general, a system that can learn without indirect negative evidence makes a weaker assumption about the processing power of the learner and its ability to remember examples over time. The learner presented in this thesis does not utilize indirect negative evidence.

### 1.1.4 The Subset Principle

Consider the case of two languages, $L_1$ and $L_2$. Suppose that a learner must choose either $L_1$ or $L_2$ when presented with some set of example sentences drawn from either language. Suppose also that the learner must be able to recover if it picks the wrong language by mistake.

If $L_1$ and $L_2$ are disjoint, then the decision is an easy one; the presence of a single sentence from $L_1$ will provide enough evidence to choose $L_1$, and the presence of a single sentence from $L_2$ will provide enough evidence to choose $L_2$. Even if the learner makes a mistake, and picks the incorrect language, it will soon encounter example sentences not in the language it has chosen, which will thus provide evidence that should allow the learner to correct its mistake.

If $L_1$ and $L_2$ intersect, then there will be some set $C$ that contains sentences that are in both $L_1$ and $L_2$. In this case, the learner must wait until it detects the presence of some sentence in $L_1$ or $L_2$ that is not in $C$ in order to choose the correct language. If the learner makes a mistake, and chooses the incorrect language, it will still be able to correct its mistake when further evidence is encountered; namely, some sentence not

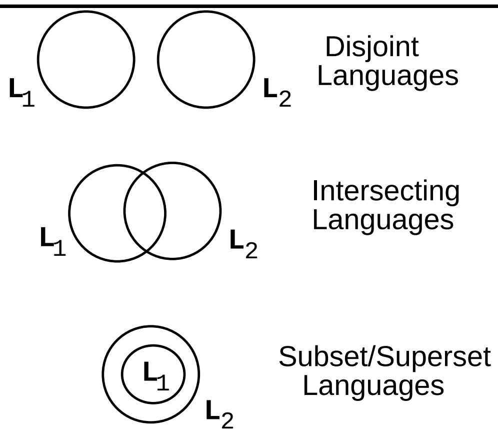

Figure 1.2: **Disjoint, Intersecting, and Subset/Superset Languages**

in $C$ or in the incorrect language but in the language to be learned.

Consider the case where $L_1$ is a subset of $L_2$; that is, every sentence in $L_1$ is also in $L_2$. The learner can correctly choose $L_2$ if it encounters some sentence that is in $L_2$ but not in $L_1$. If the learner never encounters such a sentence, then it could choose $L_1$. However, consider the case where the learner makes a mistake and chooses $L_2$ when the correct language is $L_1$. In this case, all the subsequent sentences that the learner will encounter, taken from $L_1$, will also be sentences in $L_2$ — so the learner will have no way of determining that it has made a mistake, since all the remaining data will be consistent with its selection of language $L_2$. This type of error, which cannot be detected by the learner on subsequent positive-only data, is referred to as an *unrecoverable error*.

The formal problem of learning subset languages with positive-only data has been discussed by Angluin (1978), Wexler and Culicover (1980), and Berwick (1985). In order to avoid unrecoverable errors when learning subset languages, they adopt the *Subset Principle*, a constraint on the ordering of hypotheses made by a learner:

> *The Subset Principle*. If there are two languages, $L_1 \subset L_2$ and the learner encounters data that are consistent with both languages, then it selects the

smaller language, $L_1$.

In a parameter-based system, the Subset Principle can be formulated with respect to parameter values; in particular, if a parameter $P$ has two values, such that $L_1$ with $P$ set to one value is a subset of $L_2$ with P set to the other value, then the first value of $P$ should be chosen if the learner encounters input data that are consistent with both values[6].

I return to a discussion of the Subset Principle and its treatment in the model presented here, in Section 2.4.

### 1.1.5 Markedness and Parameter Values

The Subset Principle provides us with one constraint on the ordering of parameter values. The term *markedness* has been used to describe the ordering of languages or parameter values according to some criteria. At least three different criteria can be used to order languages or grammatical constructions:

- *Relative Frequency*. If a language type or grammatical construction is rare among the world's languages, than it can be said to be *marked* with respect to the world's languages;
- *Difficult for Humans to Learn*. A grammatical construction that is relatively difficult for children to learn can be said to be *marked* with respect to learnability[7];
- *Difficult for Computers to Learn*. A language that is relatively difficult for a computer program to learn can be said to be *marked* in a computational learning algorithm. For example, a superset language can be relatively difficult to learn,

[6]Using the Subset Principle to enforce a strict ordering on parameter values during learning depends on other considerations, most notably the Independence Principle; see (Wexler and Manzini, 1987) for further details.

[7]For example, Ochs (1985) has noted that some word orders in adult Samoan appear much later than more commonly attested orders, presumably because they are harder for children to learn.

because the learner must be certain that some subset of that language is not the correct language. The Subset Principle is essentially a statement about the relative markedness of two languages, the subset and the superset, and holds that the superset language is more marked.

For the purposes of constructing a learning program, we are concerned with the notion of markedness as it relates to the order of acquisition. It is necessary to decide which value of each parameter should be the "default" value that is tried first by the learning program. For this reason, I use the term *marked* to refer specifically to any parameter value that is not the default value. The question of *which* value should be the default value is not easy to answer. In general, most decisions of this nature seem to be based on a mixture of intuitions concerning relative frequency and learnability constraints like the Subset Principle (Dresher and Kaye, 1990; Wexler and Manzini, 1987; Berwick, 1985). A more detailed investigation of these issues is beyond the scope of this thesis; in fact, I simply replicate most of the assumptions made by Dresher and Kaye with respect to default and marked parameter values. However, the Subset Principle and the ordering of parameter values during learning are discussed further during the evaluation of other learning models (Section 1.4.1 and Section 1.4.3).

## 1.2 Computational Parameter-Setting Models

In grammatical theory, parameters are often represented as binary (yes/no) answers to particular questions about the grammar (e.g., "Does the grammar allow null pronominal subjects?" (Hyams, 1986)). The state of binary parameters is easily captured by associating them with a single bit in a computer program, with a value of 0 corresponding to the "no" answer and a value of 1 corresponding to a "yes" answer. Thus, for any particular set of parameter values, those values can be represented in a computer program as a *bit string*, or sequence of 0's and 1's[8].

Each bit string corresponds to a single unique way of setting all the parameters in the grammar. Each bit string *encodes* the language corresponding to that particular grammar, as defined by the principles of the universal grammar. The class of languages to be learned is therefore finite, and corresponds to the set of all possible bit strings. This class is of size $2^n$, where $n$ is the number of binary parameters in the grammar.

```
0000                                ; No bits set.
1000 0100 0010 0001                 ; One bit set.
1100 1010 1001 0110 0101 0011       ; Two bits set.
1110 1011 1011 0111                 ; Three bits set.
1111                                ; Four bits set.
```

Figure 1.3: **Possible Bit Strings,** $n = 4$

Parameter learning can therefore be cast as a search problem: for a given set of sentences, find the bit string that encodes the correct grammar for those sentences. Under the assumption that learning takes place incrementally, as the child processes more and more sentences in the linguistic environment, this search should be linked to the actual processing of the input sentences and the current state of the grammar. In other words, the learner processes each sentence using its current hypothesis about the grammar (encoded as a particular bit string), and perhaps modifies that hypothesis (selecting another bit string) if it cannot successfully assign an interpretation to the input sentence. Constructing this type of model requires that a computational implementation of the language processor itself (for example, a syntactic parser) must also

[8]Some researchers have introduced the possibility of multi-valued (non-binary) parameters (e.g., the bounding domain parameter (Wexler and Manzini, 1987)). In this thesis I consider only binary parameters; furthermore, I assume that any multi-valued parameter can be suitably encoded as a sequence of binary parameters, whose values taken together indicate the value of the parameter in question.

be constructed before a learning algorithm can be tested. The basic architecture for this type of system is illustrated in Figure 1.4.

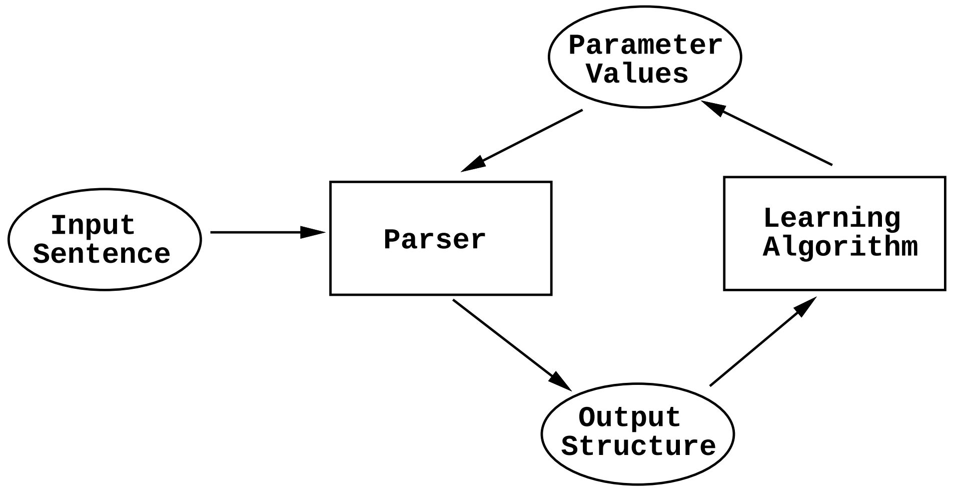

Figure 1.4: **Basic Parameter-Setting Architecture**

In this architecture, the language processing system (parser) and the learning algorithm are distinct. The architecture is basically a simple feedback loop: the parser assigns a structure to each input sentence based on the current grammar, and the learning algorithm has a chance to modify the current hypothesis based on that output structure. For example, if the parser cannot assign a legal structure to the input given the current parameter values, the learning algorithm may decide to select one parameter at random and flip its value, and then go on to the next input. In this thesis, I use the term *activation* to describe the action of a learning algorithm when it considers a new hypothesis.[9]

In addition to some sort of incremental feedback to the parser, the learning algorithm may also have a termination condition, upon which processing ends and a

[9]In this architecture, learning is implicitly incremental, i.e., the learner may make a decision after each and every input example. This is in contrast with batch learning architectures, like the one found in YOUPIE (cf. Section 1.4.1).

particular hypothesis is decided upon. For example, the learner might process the input until a certain number of input examples are parsed correctly with no errors, and then select the current hypothesis as the final grammar.

The particular parameter-setting model described in Chapters 2 and 3 of this thesis presents just one possible instantiation of this basic architecture[10].

## 1.3 Evaluating Learning Models

Constructing a computational learning model for a set of parameters forces the theorist to be explicit about the assumptions made by the theory. Formalizing a parameter model requires a concrete definition of the input, output, internal data structures, and algorithm to be used by the learner. A parameter model that is implemented as a computer program is amenable to exhaustive testing under a wide variety of conditions. Such testing can and often does lead to unexpected or unforeseen results, and can help to motivate theory refinement. An important reason for formalizing a learning theory is that it can help to validate any cognitive claims that are made. Predictions that are made about the relative difficulty of certain types of processing or learning in a computer model can be tested and confirmed by the collection and analysis of a wide range of test results.

An additional reason for implementing a learning theory in a computational model is that it forces the theorist to be explicit about the underlying theory of language processing. The architecture shown in Figure 1.4 cannot be instantiated without a clear computational embodiment of the linguistic theory represented by the Parser module, whether it is a syntactic parser or phonological stress assigner[11]. In constructing

[10]In addition to the parameter-based stress assignment parser presented in this thesis, there have been at least two computational implementations of parameter-based models of syntactic parsing; cf. (Gibson, 1991; Fong, 1990).

[11]It is possible to abstract away from the actual implementation of the parser module using some convenient abstraction of the output structure and the parsing algorithm itself. See (Clark, 1990b) for an abstraction of syntactic parsing. The phonological learner presented in this thesis contains a full

a testable learning model, the theorist may uncover shortcomings of the linguistic theory he chooses to implement. He may also find support for a particular theory, if it fits well into a theory of learnability; ultimately, a complete theory of language must account for the phenomenon of learnability.

In the remainder of this section, I will summarize the formal criteria, computational criteria, and cognitive criteria by which a learning model can be evaluated. The specifics of the particular learning algorithm adopted here are presented in Chapter 2.

### 1.3.1 Formal Criteria

Formal criteria for successful learning can be based on the notion of a *learning paradigm* (Osherson, et al., 1986), which provides a formal framework for describing particular learning problems.

To formalize a particular learning paradigm, we must consider:

1. the learning algorithm (learner);
2. the set of languages to be learned (search space);
3. the set of possible data presentations in which a particular language to be learned is exhibited to the learner (learning environment);
4. the succession of hypotheses conjectured by the learner given a particular data presentation (learning behavior).

Items 1, 2 and 3 are defined by the particular learning problem and the algorithm to be evaluated; Item 4 is derived by considering the behavior of the chosen learner in the chosen learning environment. Formal criteria for successful learning are formulated in terms of the learner's behavior in different learning environments. Note that a particular learning behavior might be judged successful by one set of criteria and

implementation of the parsing module.

unsuccessful by another. For example, suppose that learning algorithm A produces an answer after only a considering a finite number of data examples, but with only a probability of .9 that the chosen answer is the correct one. Suppose that learning algorithm B produces an answer with a probability of 1 that the chosen answer is the correct one, but does so "in the limit" — after considering a potentially infinite set of data examples (see the definition of convergence, below). In such a case there is no guarantee of a correct answer in finite time. Although algorithm B might be preferred on theoretical grounds, since it guarantees success, a practical learning system (i.e., one that is intended to solve useful problems in finite time on available computing hardware) might prefer algorithm A on practical grounds.

It is worth emphasizing that defining the learning paradigm has two major implications for computational simulation of language learning:

- It provides a framework for a precise definition of the learning problem;
- Given a precise definition of the learning problem, it is possible to formulate criteria for acceptable behavior of the chosen learning algorithm.

In general, formal learning theory is concerned with the study of different learning paradigms. Slight changes in the nature of the learner, the result to be learned, the environment, etc. can have a dramatic effect on the difficulty of the learning problem. For example, presenting clearly identified negative examples to the learner can reduce the difficulty of choosing a correct generalizing hypothesis, since overgeneralization can be corrected by the appearance of a single appropriate negative example; a learner with no access to negative examples must be more cautious, since overgeneralization can only be detected otherwise by the absence of additional confirming data over a long (statistically meaningful) sequence of input examples.

The learning framework presented in this thesis is based on the the *language identification* paradigm (Osherson, et al., 1986). There is a set of hypotheses available to the learner, corresponding to the set of possible grammars. The learner is presented with an infinite stream of data examples. The learner considers each piece of data, possibly changing its hypothesis after reading a piece of data. After a certain period of time, the learner should select the correct hypothesis, and hold that hypothesis infinitely afterwards. If this occurs, we say that the learner has *converged* to the correct hypothesis in the limit.

As mentioned before, it is possible to modify this paradigm such that the learner chooses some hypothesis as a "final" hypothesis after a finite number of input examples have been processed. Finite-time learners can stop processing after some pre-determined number of data examples, or they can stop learning when they have chosen a hypothesis that satisfies some evaluation metric. Osherson, Weinstein and Stob (1986) discuss the notion of "self-monitoring" learners, which use some sort of evaluation metric or termination condition to stop learning when a hypothesis meeting some criterion has been selected. For instance, a learner might be given both a data presentation *and* the hypothesis to be learned; the termination condition would be to stop learning when the current hypothesis matches the hypothesis to be learned. In this case, the evaluation metric is that the current hypothesis matches the target hypothesis. Other possible evaluation metrics include selecting hypotheses that account for a certain number of consecutive data examples, etc.

There are two important characteristics of self-monitoring learners which we consider here:

- Learning can take place in finite time on real hardware;
- The learner becomes prone to horizon effects.

The term *horizon effects* is used in the machine learning literature to refer to situations where a learning algorithm makes an indelible decision based on a finite data sample, where consideration of further data would lead it to choose otherwise. In some cases this can make a difference between successful and unsuccessful learning.

It is important to note that the combination of learning algorithm, search space and learning environment combine to determine whether a self-monitoring learner can exhibit acceptable behavior. Consider, for example, a learning paradigm with the following characteristics:

- When holding some hypothesis $h$, the learner selects a new hypothesis $h'$ only when the current data example $s \notin L(h)$;
- The learner is presented only with examples $s \in L(h_{target})$.

Since the learner only changes its hypothesis when it encounters an example not in the currently-hypothesized language, it will never give up the correct hypothesis once it has been selected (since all the data examples are in that language). In this case, a self-monitoring learner can be considered an appropriate computational idealization of an infinite learner, since the infinite learner's hypothesis would never change after the first time the correct hypothesis was selected (i.e., there is no potential for horizon effects).

On the other hand, consider another paradigm with the following characteristics:

- When holding some hypothesis $h$, the learner selects a new hypothesis $h'$ only when the current data example $s \notin L(h)$;
- The learner is presented both with examples $s \in L(h_{target})$ and $s \notin L(h_{target})$.

In this paradigm, a self-monitoring learner cannot be used to idealize the behavior of an infinite learner. Since the learner selects a new hypothesis when some $s \notin L(h_{target})$ is encountered, it may change its hypothesis *after* the correct hypothesis has been chosen. Since in general this may happen again and again (each time noise is encountered in the data presentation), an infinite learner might oscillate between the correct hypothesis and some other hypotheses and never converge, even though a self-monitoring learner would stop the first time it conjectured the correct hypothesis.

In the model presented here, self-monitoring learning is used to limit the length of each computational simulation; since thousands of trials were used to test the learner, this was the only feasible alternative. Nevertheless, the problem of horizon effects when learning with noisy data remains.

In the chapters that follow, I will discuss the performance of the learning algorithm in both a paradigm without noisy examples (Chapter 3) and a paradigm with noisy examples (Chapter 5). I will return to a discussion of noisy data and self-monitoring learning in Chapter 5.

### 1.3.2 Computational Criteria

As mentioned in the foregoing section, evaluation of a learning algorithm from a computational point of view has various practical concerns that are not always considered by formal criteria. Clearly, a formally sound algorithm that guarantees correct results in the limit is less interesting as a computational simulation of learning if it cannot be computed in finite time. In general, it is desirable for a learning algorithm to be *accurate* (always converging to the desired knowledge state in any given learning situation), *robust* (converging even when given a learning situation containing some incorrectly classified examples) and *efficient* (minimizing the convergence time and size of the hypothesis set during learning).

Nevertheless, when we consider models of child language learning, an algorithm that "gets the right answer" isn't good enough. For example, Osherson et al. (1986) have shown that any finite class of grammars is learnable when given only data examples $s \in L(h_{target})$. In some finite search space (like those determined by a finite set of parameter values), it would be trivial to construct an exhaustive search algorithm that tries each hypothesis in the learning space until it selects the correct one; since the space is finite, the learner would have to find the correct hypothesis after a finite amount of time. Since children do not seem to follow an exhaustive enumeration of potential grammars, this kind of solution seems unsatisfactory. The goal of language learning research should be to construct a *cognitive* model of language acquisition, one that is intended to model with empirical fidelity the learning processes of children learning language. Given this goal, a learning algorithm must also obey an additional set of cognitive constraints, which make the problem of learning in a finite parameter space considerably more interesting.

### 1.3.3 Cognitive Criteria

In this thesis, I adopt the cognitive evaluation criteria for language learning models set forth by Pinker (1979). These criteria are used both in the evaluation of recent work in parameter setting and in describing the focus of the present work.

1. *The Learnability Condition*. Children uniformly succeed in learning their native language, regardless of their ability to learn chess, calculus, and other complex cognitive skills. A learning model should therefore be able to acquire the appropriate language to be learned, in a wide variety of learning environments like those experienced by children.

2. *The Equipotentiality Condition*. Children learn whatever natural language they are exposed to as infants. A learning model should be able to learn any of the natural languages (i.e., it must not posit any learning mechanism that is particular to the constructions of a certain language, or that explicitly excludes one or more naturally occurring languages).

3. *The Time Condition*. A learning model must learn the language it is given to learn within the time span normally required by children learning the language.

4. *The Input Condition*. A learning model must not require types or quantities of input information that are not available to the child language learner.

5. *The Developmental Condition*. A learning model should make predictions about the intermediate stages of acquisition that agree with the empirical findings concerning the developmental stages of the child language learner.

6. *The Cognitive Condition*. A learning model should be consistent with what is known about the cognitive faculties of the child, including any known limitations on perceptual discrimination, conceptual ability, memory capacity, and attention span.

Each of these conditions has particular consequences for a computational learning model. The Learnability Condition implies that the learning algorithm must always select the correct grammar when presented with a plausible set of input data for the language. The Equipotentiality Condition implies that the learning algorithm should be able to learn any of the languages within the search space delimited by its parameter values. The Time Condition implies that an algorithm must derive the correct grammar in finite time (rather than in the limit), and do so after an empirically plausible number of examples has been processed. The Input Condition states that a learning algorithm must not make use of data unavailable to the child; for example, negative evidence, indirect negative evidence, and instantaneous presentation are all ruled out given the nature of the child's learning environment and cognitive capacity. The Developmental Condition implies that a learning algorithm must hypothesize a sequence of intermediate grammars that are analogous to the developing states of the child's linguistic ability. Finally, the Cognitive Condition limits the algorithmic complexity and memory used by the learning algorithm to what is feasible given our understanding of the child's processing ability and memory capacity.

## 1.4 Previous Models

In order to focus on the particular problems to be addressed in this thesis, I now turn to a discussion of two recent parameter setting models, and evaluate the strengths and weaknesses of each with respect to Pinker's evaluation criteria.

### 1.4.1 The Dresher and Kaye Model

In their 1990 *Cognition* article, Dresher and Kaye present a model of parametric variation in metrical phonology. They propose a system of 11 parameters which account for variations in the construction of metrical constituents and the assignment of primary and secondary stresses at the level of the phonological word. They also present YOUPIE, a learning system that sets the values of these parameters correctly when given sets of stressed words as input. I will briefly describe YOUPIE before turning to an evaluation and discussion of the system's characteristics; for a full explication of the details, see (Dresher and Kaye, 1990)[12].

#### Parameter Model

The basic premise of metrical phonology is that the stress patterns assigned to words are determined by metrical structures that describe the higher structure of syllables, analogous to the way that syntactic structures describe the internal grouping of words in a sentence. The basic unit of structure is the metrical *foot*, which groups adjacent syllables into larger units. For example, the repeating stress pattern in the English pronunciation of the word *Apalachicola* is described by the metrical feet shown in Figure 1.5. The syllables are grouped into binary feet, and the first syllable of each foot receives emphasis (noted by an asterisk). The extra stress on the final foot is derived by a word stress rule that stresses the final foot in the word.[13]

The rules of foot construction, and stress assignment at the foot and word level are described by a set of parameters whose values determine all the ways in which stress patterns are assigned to words[14]. The goal of the learner is to therefore determine which set of stress patterns (as determined by the values of the parameters) can account

[12]Since the Dresher and Kaye learning domain is replicated as part of this thesis, further details concerning metrical phonology and metrical parameters can be found in Section 3.

[13]For more details concerning the theory of metrical phonology, see also (Liberman and Prince, 1977; Prince, 1983; Hayes, 1981; Levin, 1985; Halle and Vergnaud, 1987; Everett, 1988, 1989).

[14]I will delay a full description of these parameters until Chapter 3.

```
((a* pa) (la* chi) (co** la))
```

Figure 1.5: **An Example of Metrical Structure**

for the observed stress assigned to words in the learner's input.

**Input Data**

The input data to YOUPIE are words with their associated stress patterns (see Chapter 3 for examples). For each language learned by the system, the input consists of a small set of example words, which are considered simultaneously by the learner (i.e., all at once rather than one at a time). YOUPIE is therefore a "batch" learner rather than an incremental learner.

**Learning Algorithm**

YOUPIE is a *cue-based* learner. The system relies on a set of cues, or pattern-action rules, which match certain stress patterns and cause the learner to change its parameter values. An example cue might state the following: *If the first syllable in every word is always unstressed, then set $P_7$ to 1.* The system learns by considering a small set of example stress patterns all at once. YOUPIE begins with all of its parameters in a default or unmarked state, and then applies each of its cues to determine the correct settings of its parameters.

The cues themselves match particular parts of stress patterns (e.g., an initial stressed segment followed by an unstressed segment), and when matched they determine the correct setting of one or more parameters in the grammar. The use of cues has two related consequences. First, there must exist a distinct and unambiguous cue (identifying context) for each parameter value (or set of related values) to be derived.

Second, the application of cues is order-dependent – some cues must be evaluated before other cues are tried, because of certain assumptions about the current state of the grammar. YOUPIE cannot learn unless cues are ordered in a certain way.

The input to the learner is a small set of representative stressed words from the language to be learned. The sample contains no ungrammatical examples or performance errors. The learning algorithm assumes that each stress pattern is a positive example. In addition, the learning algorithm makes arbitrary cross-word comparisons across the entire set of input data, in order to determine (in some cases) whether a particular stress pattern ever appears. Thus, the learning algorithm also makes use of indirect negative evidence.

The learner is a single-hypothesis, deterministic learner. It holds only a single hypothesis at a given time, and does not perform backtracking.

**Evaluation**

Like any learning algorithm, the Dresher and Kaye model makes certain assumptions about the learning paradigm in order to demonstrate learnability. Although the work presented in (Dresher and Kaye, 1990) represents an important step forward in the construction of parameter-based learning systems, there are certain assumptions made by the system that are undesirable with respect to Pinker's evaluation criteria:

- *The Learner Has Access to Indirect Negative Evidence*. As discussed in Section 1.1.3, a learner that makes use of indirect negative evidence must "remember" a large number of examples (or process a large number of examples at one time), in order to determine whether a certain type of input example ever occurs. The use of indirect negative evidence in a model of child language learning seems inappropriate, since remembering or processing a large number of examples requires a large memory. Since the child's memory capacity during development is quite limited, the use of indirect negative evidence seems to violate Pinker's Cognitive Condition: the model is not consistent with what is known about the cognitive faculties of the child. Since it utilizes information not available to the child learner, YOUPIE could also be said to violate the Input Condition.

- *The Learner's Intermediate Stages are Predetermined*. The empirical validity of a learning model can be judged by how well it predicts the intermediate stages attained by children learning language. YOUPIE can acquire parameters only in a rigid order, which is predetermined by the nature of the learning algorithm itself rather than by the characteristics of the language being learned. In fact, it is quite possible that the order of parameter setting differs across languages. For this reason, it seems doubtful that this type of cue-based learner can satisfy Pinker's Developmental Condition, since the predictions made about the intermediate stages of the learner are fixed and idiosyncratic to the learning algorithm itself.

- *The Learner is Not Robust*. YOUPIE assumes that each input example is grammatical, and that ungrammatical examples or performance errors never occur. In fact, YOUPIE cannot learn when exposed to ungrammatical or "noisy" examples[15]. Since the learning environment faced by children undoubtedly does contain noisy examples, YOUPIE cannot meet the Input Condition, since it requires unrealistic input data that are not available to the child language learner.

### 1.4.2 Learnability Problems and Single Hypothesis Learning

Several learning algorithms (including YOUPIE) which have been proposed as models of language learning are single-hypothesis, deterministic, error-driven systems. This type of learner is characterized by the following attributes:

- The learner maintains a single active hypothesis $h$, initially the least-marked hypothesis (all parameters are set to 0);

[15]Dresher, personal communication.

- If the learner encounters an example $s \notin L(h)$, then it changes the value of a single parameter from 0 to 1;
- The learner cannot set a parameter from 1 to 0 (backtracking is not allowed);
- Each hypothesis selected by the learner obeys the Subset Principle[16].

For example, in the early model proposed by Wexler and Culicover (1980), the learner maintains a single hypothesis, consisting of a set of transformation rules, which is changed only when the current example sentence (surface structure) cannot be derived from the example base structure using the existing set of transformations. In Berwick's model (1985), the learner maintains a single hypothesis, consisting of a set of phrase-structure rules, which is modified only when the current example sentence cannot be parsed with the current grammar. In the recent model proposed by Dresher and Kaye (1990), the learner maintains a single hypothesis, consisting of a set of parameter values, which is modified when some example stress pattern or patterns satisfies the trigger condition of a particular learning cue.

There are two characteristics of this type of model that cause undesirable behavior in certain situations, leading to unrecoverable errors:

- Flipping a parameter when a single example $s \notin L(h)$ is encountered;
- Maintaining only a single hypothesis.

**Parameter Flipping**

Consider a learning algorithm that flips a parameter when some example sentence $s \notin L(h)$ is seen in the input. Such an algorithm will always choose a new hypothesis when a sentence outside its language is encountered. If we assume that the input data contain only grammatical example sentences from the target language, then the

[16]Note that this is vacuous, if the the least-marked value of a subset parameter is always the subset value.

learner can safely assume that the presence of some $s \notin L(h)$ in the input means that the current hypothesis $h$ is incorrect and that $L(h)$ is not the target language. In reality, the problem faced by child language learners is not so ideal. The input data processed by children contain at least occasional performance errors, restarts, ungrammatical sentences, etc. It is unlikely that an algorithm that always flips a parameter in the presence of some $s \notin L(h)$ will be able to select the correct hypothesis when learning in a noisy environment.

| **Input Example** | **Hypothesis** |
|---|---|
| $\vdots$ | $\vdots$ |
| $s_m \in L_{target}$ | $L_{target}$ |
| $s_{m+1} \in L_{target}$ | $L_{target}$ |
| $\vdots$ | $\vdots$ |
| $s_n \notin L_{target}$ | $L_j$ |
| $s_{n+1} \in L_{target}$ | $L_k$ |
| $s_{n+2} \in L_{target}$ | $\vdots$ |
| $\vdots$ | $\vdots$ |

Figure 1.6: **Fluctuation Between Incorrect Hypotheses**

Consider the example learning scenario illustrated in Figure 1.6. At some time $m$, the learner encounters a piece of data $s_m$ that is in the target language, and hypothesizes the target language, $L_{target}$. At this point it will maintain the correct hypothesis for any number of subsequent input examples that are in $L_{target}$. However, suppose that at time $n$ the learner encounters some ungrammatical example sentence $S_n$ that is not in the target language $L_{target}$. Since the learning algorithm flips a parameter whenever it sees some $s \notin L(h)$, it must change its hypothesis away from the correct hypothesis. Suppose that the learner chooses some hypothesis $L_j$. Since backtracking isn't allowed, the learner will never regain the correct hypothesis once it has abandoned it. Even if backtracking were allowed, the learner would fluctuate between the correct hypothesis and some other hypotheses, since every $s \notin L_{target}$ would cause it to select some $L' \neq L_{target}$.

**Single-Hypothesis, Deterministic Learning**

In a single-hypothesis, deterministic learning system, the learner may hold only a single hypothesis at any one time (corresponding to a single set of parameter values), and it may not reset a parameter that it has already set. In a single-hypothesis deterministic system, changing the value of the wrong parameter can lead to an unrecoverable error in certain situations. As a result, the learner selects an incorrect hypothesis.

The example learning scenario illustrated in Figure 1.7 contains two parameters, which derive 4 possible hypotheses[17]. Let us assume that the first parameter is a subset parameter; as a result, $L(H_1) \subset L(H_3)$ and $L(H_2) \subset L(H_4)$. Suppose that the learner starts with $H1$, the least-marked hypothesis (both parameters are set to 0), and that $L(H_2)$ is the target language. Since $L(H_1)$ is not the target language, the learner will soon encounter some $s \notin L(H_1)$. It will then flip one of its parameters. The question is, which one? There are two possibilities; setting the first parameter will yield $\langle 1, 0 \rangle$, or $L(H_2)$, and setting the second one will yield $\langle 0, 1 \rangle$, or $L(H_3)$. A single-hypothesis deterministic learner must pick just one of the two possibilities, since it can hold only a single hypothesis at one time. Suppose that the learner picks not $H_2$, but $H_3$. It will soon encounter example sentences not in $L(H_3)$. Since backtracking is not allowed, the learner must then set the second parameter, yielding $\langle 1, 1 \rangle$, or $L(H_4)$. Note that since $L(H_2) \subset L(H_4)$, there is no way for the learner to reach the correct hypothesis without backtracking. However, even if backtracking were allowed, the learner would never encounter data that would prompt it to select $L(H_2)$ once $L(H_4)$ is selected. Since all subsequent input examples $s \in L(H_2)$ are also in $L(H_4)$, the superset language, the learner would (falsely) assume that it had selected the correct language.

[17]This example is drawn from Clark's discussion of the Causality Problem (Clark, 1988).

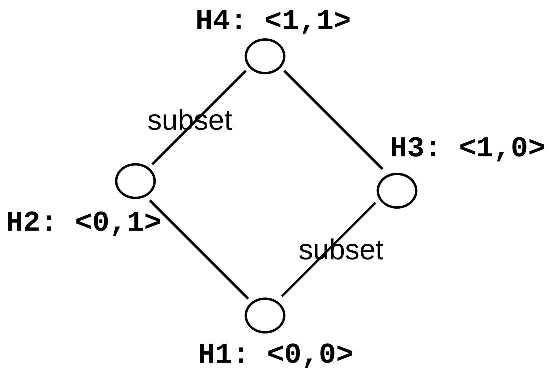

Figure 1.7: **A Simple Hypothesis Ordering Problem**

**Subset Shift**

As noted in (Clark, 1990a) and (Nyberg, 1990), it is often difficult for a strictly deterministic learner like YOUPIE to determine which parameter to reset. This has been called the Causality Problem by Clark, and is related to the Credit/Blame Assignment Problem in mainstream machine learning and Duhem's Problem in the philosophy of science[18]. Because certain parameters can interact in a way that makes it very difficult to formulate coherent, unambiguous cues, a cue-based learner can encounter situations where it cannot select the correct parameter to reset.

Clark (1990a) notes that parameters which are not subset parameters themselves can combine to create superset/subset situations, resulting in a parameter configuration he calls *subset shift*. For the sake of illustration, consider the examples from English and Irish presented in (Clark, 1990a).

English is a language that allows Exceptional Case Marking; that is, certain verbs

[18]cf. (Quine, 1960).

may subcategorize for a subordinate clause that contains a non-finite verb:

(1) *John considers [Bill to be a fool].*

On the other hand, Irish is a language that allows Structural Case Marking; that is, the subject of a non-finite clause may receive case marking by virtue of its structural position:

(2) *Is cuimneach leo [iad a bheith ar seachrán].*
(They remember they to be lost).

Clark notes that these possibilities for case-marking can be captured by two parameters:

- **ECM**: A verb may subcategorize for an infinitival IP.
- **SCM**: The subject of IP may receive abstract Case by virtue of its structural position.

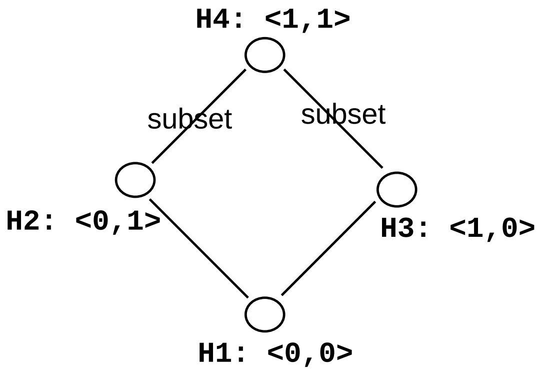

Figure 1.8: **The Subset Shift Problem**

The Subset Shift situation presents a significant problem for a single-hypothesis deterministic language learner. Consider the scenario illustrated in Figure 1.8. $H_1$

represents a language with neither Exceptional Case Marking nor Structural Case Marking ($\langle 0, 0\rangle$); $H_2$ and $H_3$ represent English and Irish, respectively ($\langle 0, 1\rangle$, $\langle 1, 0\rangle$); and $H_4$ represents a language that allows *both* Exceptional Case Marking and Structural Case Marking ($\langle 1, 1\rangle$).

Assume that the learner begins with hypothesis $H_1$. Assume also that the language to be learned is Irish, or $L(H_3)$. The learner will soon encounter some example sentence $s \notin L(H_1)$, and pick a new hypothesis. Unfortunately, on the basis of a single example that exhibits case marking of the subject of a non-finite clause, it is impossible to determine whether case marking is due to ECM or SCM[19]. A strictly deterministic learner that flips a single parameter at a time must select either $H_2$ or $H_3$. Assume that the learner picks $H_2$, making the (incorrect) assumption that the language it is learning is ECM rather than SCM. Further examples that are not in the language will lead the learner to select $H_4$; since it cannot backtrack, it must set the first parameter once it has chosen $H_2$, resulting in selection of $H_4$, an incorrect hypothesis. Even if backtracking were allowed in this situation, the learner could not recover from this error, since $H_4$ is a superset of $H_3$, and further positive-only evidence would support the choice of $H_4$. The learner has "shifted" into a superset hypothesis that it cannot retract, even though each of its local steps satisfied the Subset Principle. It should be clear from the example of Subset Shift that interactions between parameters can cause situations that cause a strictly-deterministic error-driven learner to make an unrecoverable error.

### 1.4.3 The Clark Model

In an attempt to address the shortcomings of single-hypothesis, error-driven parameter-setting models, Clark abandons determinism in favor of a genetic algorithm model of parameter-setting (Clark, 1990a, 1990b). Clark's learner is a non-deterministic, fitness-driven model based on the Darwinian notion of natural selection. The model, called

[19]As shown by Clark, this distinction can only be determined on further consideration of other structural properties of the language being learned.

DARWIN, has been implemented in a computer program that learns abstract syntactic parameters[20].

### Learning Algorithm

DARWIN assumes the existence of a syntactic parser that builds a representation of each input example, leaving one or more *nodes*, or constituent structures, in its processing buffer. If the parser constructs a single node, this is considered to be a successful parse. If more than one node results from a parse, the learner assumes that the parser failed to create a single, coherent structure for the input sentence. The learner uses a *fitness metric* to judge the fitness of each active hypothesis relative to a given input example. Hypotheses that are more fit have a better chance of being represented in the next population.

Learning proceeds as follows. Hypotheses are represented as bit strings (sequences of 0's and 1's). The learner randomly selects one possible bit string as its initial (default) hypothesis. The input to the learner is a set of example sentences from the language to be learned. The learner considers the examples one at a time. A given hypothesis (bit string) is used to set the parameters in the parser before processing the input example. Then the fitness of the result is judged by the number of parse nodes that are constructed. The learner adjusts its population after calculating the fitness of each hypothesis. The learner continues to process input examples until it selects the most fit hypothesis.

In addition to the use of a fitness metric to strengthen fit hypotheses and prune weak ones, Clark's model also makes use of a *crossover* mechanism, which is used to "mate" fit hypotheses to produce "children" in the next population of hypotheses. For example, if there are two highly fit hypotheses 010111 and 011000, the learner might split each of these hypotheses into two smaller units (e.g., $A = 010 + 111$ and $B = 011 + 000$). The smaller units are then "crossed over", as shown in Figure 1.9,

[20]For an overview of Genetic Algorithms, see (Booker, Goldberg and Holland, 1990).

producing two new hypotheses that are "hybrids" of $A$ and $B$.

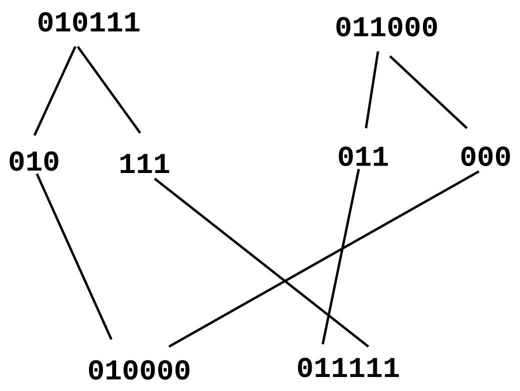

Figure 1.9: **Crossover**

DARWIN also makes use of *mutation*, a mechanism common to genetic algorithm learners, which introduces a small random factor that occasionally changes one bit in an existing hypothesis to produce a new hypothesis[21].

## Evaluation

As learners, genetic algorithms have several desirable properties. They are resistant to noise in the input sample, and they can rapidly localize search to smaller areas of high fitness within a very large search space (Goldberg, 1989). Using a genetic algorithm to set parameters can avoid some of the problems inherent in single-hypothesis, error-driven learners. In particular, DARWIN is not prone to the problems of noisy input, overgeneralization, subset shift, etc., which can lead deterministic learners like YOUPIE to an unrecoverable error. DARWIN also does not rely on indirect negative evidence. Nevertheless, genetic algorithms have certain characteristics that make them less desirable as models of child language learning:

[21]Clark's model also makes use of a negative fitness factor that penalizes superset hypotheses in order to avoid overgeneralization and unrecoverable errors. A full discussion of DARWIN's implementation is beyond the scope of this thesis; for full details, see (Clark, 1990b).

- *The Intermediate Stages of the Learner are Unrelated*. The effects of crossover and random mutation allow the genetic learner to sample a huge search space; however, the learner predicts that the child fluctuates wildly in his choice of grammar, resetting several parameters at once. Given the abstract nature of parameters, this can result in two successive languages that are completely unalike. This violates Pinker's Developmental Condition, since child language acquisition proceeds slowly and steadily through well-defined stages.

- *Non-Determinism is Unconstrained*. There is no intrinsic upper bound on the number of active hypotheses in DARWIN. In fact, to be successful DARWIN must maintain larger and larger populations when there are more parameters to be learned. Since the learner must process the input example and evaluate a complicated fitness metric for each active hypothesis, it seems unlikely that DARWIN could satisfy Pinker's Cognitive Constraint when large numbers of hypotheses are active, since this would require memory and processing faculties unavailable to the child[22].

**Discussion**

In general, YOUPIE is a learner that is too constrained and rigid in its hypothesis selection; it fails to satisfy the Developmental Condition because it has a single, idiosyncratic order of acquisition. On the other hand, DARWIN is too unconstrained and free in its hypothesis selection; it fails to satisfy the Developmental Condition because it jumps about too wildly in the hypothesis space.

DARWIN makes far weaker assumptions about the learner, learning in the presence of noisy examples and without access to indirect negative evidence. However, its hypothesis selection mechanism is too unconstrained to satisfy the Developmental

[22]Clark (personal communication) notes that although currently there exists no proof of the upper bound on the number of hypotheses in DARWIN, there are reasons to believe that such a bound does exist, and is related linearly to the number of parameters.

Condition, and there is no inherent limit to the non-determinism present in the model.

In the following section, I discuss the work to be presented in this thesis that improves on both the YOUPIE and DARWIN models in significant ways.

## 1.5 The Methodology and Scope of the Thesis

The work presented here is based on the following general ideas:

- Linguistic theory and parameter-setting strategies in particular are often presented in incomplete or unimplemented form;
- Computational implementation of a linguistic theory and a learning theory can provide feedback concerning the level of completeness and coherence of each theory;
- Computational simulation is a valuable tool for evaluating the behavior of linguistic models and learning algorithms;
- Empirical results from computational simulation can determine problematic areas in linguistic theory and motivate theory refinement.

### 1.5.1 The Simulation Model

A particular language learning simulation requires the following:

1. An algorithmic description and implementation of the linguistic principles and parameters of the given domain;
2. An algorithmic description and implementation of the learning algorithm for the above;
3. A comprehensive set of meaningful test data;

4. A procedure to test and gather results on the combined linguistic/learning system given the test data.

The process of simulation raises the following questions about the components of the model:

- *Linguistic Theory*. Does the theory propose principles and parameters that are computationally plausible?
- *Computational Tractability*. Is the model, taken as a whole, computationally tractable?
- *Cognitive Tractability*. Does the observed behavior of the learning model satisfy cognitive constraints?
- *Test Data*. How comprehensive and realistic are the test data? Are noisy data allowed?
- *Test Results*. Do the test results reflect empirical fidelity?

### 1.5.2 Goals

This thesis presents a parameter setting algorithm with the following characteristics:

- *No Use of Indirect Negative Evidence*. Unlike YOUPIE, the proposed model does not make use of indirect negative evidence during learning. This is crucial in order to satisfy the Cognitive Condition (and, indirectly, the Input Condition), since children can't remember all the examples they have heard;
- *Robustness*. Unlike YOUPIE, the proposed model can learn in the presence of ungrammatical examples. Since real learning environments contain a certain amount of noise, this is necessary in a realistic learning model;

- *Not Prone to Hypothesis Ordering Problems*. Unlike YOUPIE, the proposed model is non-deterministic and can avoid hypothesis ordering problems. Since current linguistic theories have proposed parameterizations that give rise to such problems, it is the goal of this thesis to account for them;

- *Limited Non-Determinism*. Unlike DARWIN, there is a well-defined bound to the amount of non-determinism allowed in the proposed model. This is important in satisfying the Cognitive Constraint, since it is not plausible that a child entertains an unlimited number of hypotheses about his grammar at any given time;

- *Conservative Hypothesis Selection*. Unlike DARWIN, the proposed model chooses new hypotheses that are more closely related to its previous hypothesis, so that the successive hypotheses advanced by the learner form a coherent set of developmental stages. This is necessary in order to satisfy the Developmental Condition.

### 1.5.3 Contributions

The main contributions of the thesis are the following:

1. A methodology for computational simulation of linguistic learning;

2. A plausible learning model that is computationally superior to previous models that have addressed learning in this paradigm;

3. A model with better cognitive fidelity than previous models that have addressed learning in this paradigm.

In short, this thesis takes seriously the cognitive constraints on child language learning that have been suggested by psycholinguists, while also providing a model with better computational behavior that the existing computational models of parameter setting.

This thesis *does not* argue for a theoretically optimal model for language learning. Given the lack of mathematically rigorous proofs of cognitive phenomena, this is not a realistic objective. The thesis is focussed on the simulation model as a methodology for exploring ideas and intuitions about language learning. Although the particular parameter setting algorithm presented here has superior characteristics in some respects when compared to existing parameter-setting models, I will not attempt the task of showing that there are no other learning frameworks that could be applied to parameter setting in order to solve the same problems or to discover inadequacies in a parameter model. In short, I present a model that is superior to all others that have been proposed in this learning paradigm, but allow for future improvement.

### 1.5.4 Evaluation Criteria

The goals of the thesis will be evaluated by the following criteria:

- Analysis of empirical testing on a large volume of test data in two different learning applications (metrical phonology and abstract syntax);
- Empirical testing and analysis of the model's behavior when learning in noisy environments;
- Evaluation of the model's success in solving "tough" learning problems that have been identified in the literature;
- The degree to which the model satisfies the cognitive constraints proposed in Section 1.3.

I will return to a discussion of these criteria and summarize the results of the thesis in Chapter 7.

### 1.5.5 Structure of the Thesis

The remainder of the thesis contains the following chapters:

- **Model Definition**. The definition of the parameter representation, search space, and learning algorithm, including a discussion of how the model improves on the single-hypothesis, deterministic model.

- **Learning Phonological Parameters**. A complete implementation of a parameter-based processor for metrical stress assignment which replicates the metrical parameters developed in (Dresher and Kaye, 1990), including a presentation of the results of learning each hypothesis in the search space determined by the parameter values and their dependencies (432 possible languages).

- **Learning Syntactic Parameters**. An application of the model to the task of learning syntactic parameters, following the abstract syntactic parsing model defined in (Clark, 1990).

- **Learning with Noisy Data**. The robustness of the model is tested in learning with noisy data, and a formal analysis of the robustness of the model is presented.

- **A Discussion of Alternative Models**. Some possible alternatives to the model presented in the thesis are discussed.

- **Conclusions**. The results of the thesis work are evaluated with respect to the evaluation criteria, and some suggestions are made concerning future work.

- **Appendices**. The thesis also contains various appendices which contain supporting examples and test results.

# Chapter 2

# Model Definition

In this section, I present the parameter-setting model developed in this thesis. The model has these basic components:

- A data structure for each hypothesis, containing a bit string representing parameter values and an associated weight indicating the amount of evidence seen for that hypothesis;
- A weighting function that adjusts the weights of active hypotheses according to how well they account for the input data;
- A hypothesis selection algorithm that expands the learner's search when there are no active hypotheses with a weight above a lower bound;
- An upper bound, such that the learning simulation will be terminated when a hypothesis attains a weight greater than the upper bound and is the only active hypothesis.[1]

[1] As discussed previously in Section 1.3.1, self-monitoring learning is used to limit the amount of time used in each learning simulation. It is expected that upper bound be set so that the hypothesis selected in this fashion will be the correct hypothesis (i.e., the bound will be close to the highest possible value of the weighting function). When analyzing the behavior of the algorithm in noisy learning environments, it is necessary to consider what happens beyond this termination point (see Chapter 5).

In the sections that follow, I discuss each aspect of the model in detail, before turning to a simple example and a demonstration of the learner's behavior on problems like those shown in Section 1.4.

## 2.1 Hypotheses and Weighting

Each hypothesis held by the learner has two components: a bit string indicating a particular set of parameter values, and a *weight*, a numerical value which indicates the level of confidence in that particular hypothesis. Intuitively, a higher weight indicates that there is more evidence for a particular hypothesis than one with a lower weight. This notion is made concrete by the following definitions:

1. The set $DATA_t$ is the set of example sentences encountered by the learner up to time $t$;
2. The set $POS_t(h) = s \in L(h), s \in DATA_t$;
3. The set $NEG_t(h) = s \notin L(h), s \in DATA_t$;
4. The value of $NPE_t(h) = |POS_t(h)| - |NEG_t(h)|$.

In other words, the *net positive evidence* ($NPE$) in support of a particular hypothesis can be calculated by counting the number of examples sentences in $L(h)$ and subtracting the number of example sentences not in $L(h)$. For example, if at time $t$, $DATA_t = \{s_1, s_2, s_3\}$, $s_1, s_2 \in L(h_i)$, $s_3 \notin L(h_i)$, then $NPE_t(h_i) = |\{s_1, s_2\}| - |\{s_3\}| = 2 - 1 = 1$.

The weight associated with each active hypothesis at time $t$ is a function of $NPE_t(h)$. The particular family of functions that I have chosen for this model is the set of sigmoid functions, $y = SIG(x) = 2/(1 + e^{-kx}) - 1$, where $k$ is a damping factor that controls the slope of the curve (see Figure 2.1).

This weighting function has two desirable characteristics:

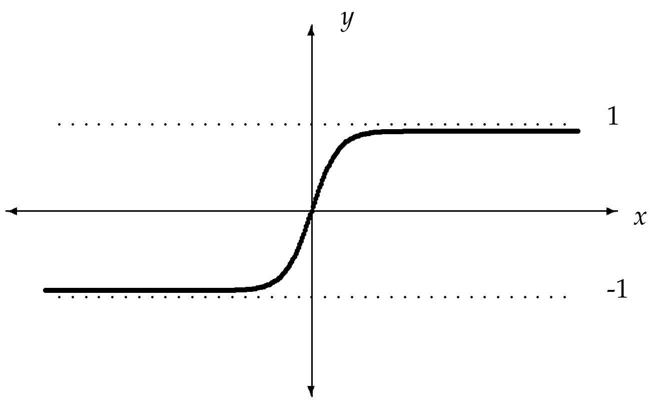

Figure 2.1: **The Sigmoid Function $y = 2/(1 + e^{-kx}) - 1$, $k = 1$.**

- The curve approaches -1 as $NPE$ gets more and more negative, and +1 as $NPE$ gets more and more positive; hence weights of -1 and +1 represent our intuitive notions of complete lack of confidence in a hypothesis vs. complete confidence in a hypothesis;
- Because the slope of the curve becomes more and more shallow as it approaches its asymptotes, fluctuations in $NPE(h)$ near the asymptotes will have little effect on the weight of $h$.

The learner checks each active hypothesis $h$ against each new piece of data $s$; if $s \in L(h)$, then $h$ receives positive weight, otherwise $h$ receives negative weight as determined by the sigmoid function. When many examples are encountered that are not in $L(h)$, the weight of $h$ will get closer and closer to -1. If the weight of a hypothesis gets close enough to -1, it will be removed from the list of active hypotheses. If the weight of a hypothesis gets close enough to +1, the learner has selected to that hypothesis. To formalize the notion of "close enough," I assume the existence of some threshold $\epsilon$, such that hypotheses with weights less than $(\epsilon - 1)$ are removed from active consideration and hypotheses with weights greater than $(1 - \epsilon)$ are selected by the learner as final hypotheses. For example, if $\epsilon = .1$, then hypotheses with weights less than -.9 will be removed from the active list, and hypotheses with weights greater than .9 will be selected as final.

The speed with which the learner approaches its asymptotes can be controlled by manipulating the damping factor, $k$. In addition, the learner may treat the examples inside and outside the language of a given hypothesis uniformly (each contribute a count of 1 or -1 to the net positive evidence) , or asymmetrically (e.g., negative examples subtract more than 1 from net positive evidence). In this model, asymmetrical coefficients are used; the relative effect of $s \notin L(h)$ is adjusted to be 5 times more salient that the presence of $s \in L(h)$ (see Figure 2.5 in Section 2.6).

To summarize, the *weight* of a particular hypothesis $h$ at time $t$ is $SIG(NPE_t(h))$, the value assigned by the sigmoid function to the net positive evidence (an integer) for $h$ at time $t$. Weights are floating point numbers that range from -1 to +1, -1 representing complete lack of confidence in $h$ and +1 representing complete confidence in $h$.

## 2.2 The Single-Value Constraint and Hypothesis Activation

So far, I have described an encoding for hypotheses (bit strings) and a representation of confidence in a hypothesis (weight). I now turn to the task of describing how the learner moves from one hypothesis to another during learning.

In a deterministic (single-hypothesis) learner, changing the current hypothesis can be achieved by "flipping" a value or values in the bit string encoding the current hypothesis. For example, 0101 can be derived from 0000 by flipping the second and fourth parameter values.

In the type of non-deterministic learner described here, more than one hypothesis may be active at a given time. This requires that the learner maintain an *active list* containing all currently activated hypotheses. Adding new hypotheses can be achieved by copying one of the current hypotheses, and then flipping some of its values. For example, $\{0000, 0101\}$ can be derived from $\{0000\}$ by making a copy of 0000, flipping its second and fourth parameter values, and then storing it along with the original hypothesis in the active list (thus indicating that both are active).

It should be clear from the foregoing discussion that there are no *a priori* limitations on how a non-deterministic learner may select new hypotheses or add to the current set of active hypotheses. It is the case, however, that child language learners (though they may experiment with certain ungrammatical ways of building sentences) do not fluctuate wildly in their choice of grammar. In fact, their grammatical development seems to move smoothly through several well-defined stages. For this reason, I adopt the following constraint on hypothesis selection:

> *The Single-Value Constraint*. At any time $t$, if the learner holds some hypothesis $h$, then it may activate only hypotheses that differ from $h$ by the value of a single parameter (Clark, 1990a)[2].

Intuitively, the Single-Value Constraint limits the learner to only those hypotheses that can be derived by flipping a single parameter. This has two desirable effects:

- The learner cannot jump between hypotheses that are completely unrelated;
- To derive a particular parameter setting, the learner must pass through a number of intermediate stages.

I adopt the notion of *1-adjacency* to describe two hypotheses that meet the Single Value Constraint; e.g., $h_i$ is 1-adjacent to $h_j$ iff $h_i$ is identical to $h_j$ except for the value of a single parameter (bit). The Single-Value Constraint limits the search performed by the learner in a way that makes it more plausible as a cognitive model of acquisition than an unconstrained non-deterministic learner.

[2]Although this constraint is discussed by Clark, he explicitly abandons it; as shown in Section 1.4.3, the absence of this constraint can have negative consequences with respect to Pinker's Developmental Condition.

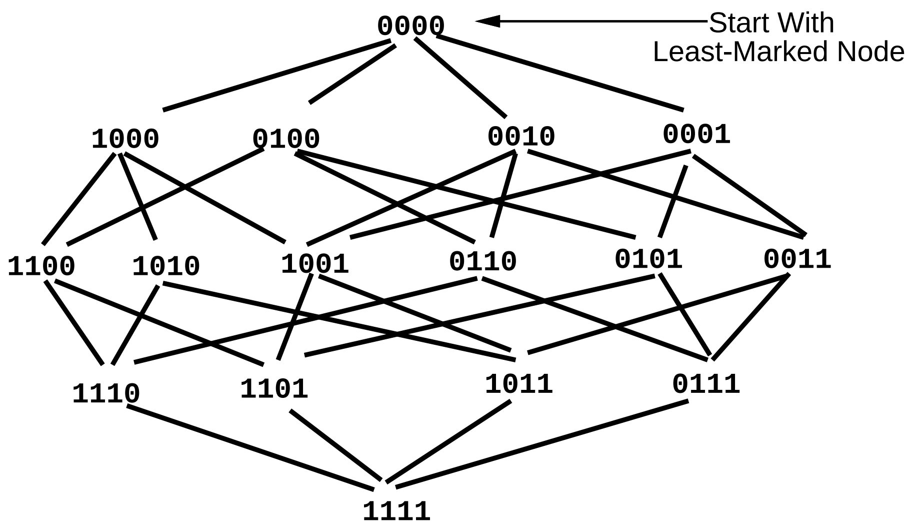

Figure 2.2: **1-Adjacent Hypothesis Graph,** $n = 4$

## 2.3 Searching the Hypothesis Graph

I assume that the learner begins with the least-marked hypothesis, a string of parameter values containing only 0's. For example, in a system with 4 parameters, the initial hypothesis would be 0000. Beginning with the initial node, we can recursively apply the notion of 1-adjacency to derive the entire set of bit strings of length n, arranged in a lattice (the boolean lattice shown in Figure 2.2 illustrates the case for $n = 4$). Each edge in the lattice represents a change in the value of a single parameter, and connects two 1-adjacent hypotheses. I will refer to this kind of 1-adjacent lattice as the *hypothesis graph*. For a learning domain with $n$ parameters, there will be $2^n$ possible hypotheses, so in general the size of the hypothesis graph can be quite large even for relatively small numbers of parameters[3].

Starting with an active list containing just the least-marked hypothesis, the learner

[3]In the implementation of the learning algorithm, the hypothesis graph itself is not explicitly represented. Only that portion of the hypothesis graph that is actively being considered by the learner (i.e., the set of active hypotheses) is represented.

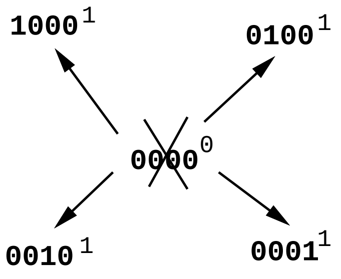

Figure 2.3: **Hypothesis Activation**

begins to process the input data and search systematically and incrementally through the hypothesis graph in order to locate the correct hypothesis. This is accomplished by weighting active hypotheses as described in the previous section, checking each active hypothesis against the current input datum and weighting it accordingly. This process continues until either 1) there is a single hypothesis with a weight $> (1 - \epsilon)$, in which case the learner has selected that hypothesis as its grammar, or 2) there are no more active hypotheses (the weights of all active hypotheses are $< (\epsilon - 1)$). In the former case, the learner stops processing and returns the hypothesis it has selected. In the latter case, the learner activates a set of new hypotheses according to the Single-Value Constraint. When the last active hypothesis is removed from the active list, all hypotheses that are 1-adjacent to it are activated and placed into the active list, and processing continues. For example, if 0000 is not the correct hypothesis, sooner or later the learner will accumulate enough evidence against it, and its weight will dip below $(\epsilon - 1)$, causing it to be removed from the active list[4]. Since it is the only active hypothesis initially, this would in turn cause all 1-adjacent hypotheses to be

[4]The number of examples $s \notin L(h)$ required for the learner to prune node $h$ depends on a) the sigmoid coefficient $k$ used to assign weight when an example $s \notin L(h)$ is encountered, and b) the value of $\epsilon$. For example, if $k = 5$ for negative examples and $\epsilon = .1$, then we can solve for $x$ in $2/(1 + e^{-5x}) - 1 < -.9$; in this case, the value of the sigmoid will be less than $-.9$ for any $x \geq 6$ (since $SIG(-5 * 6) = -0.905148$). Of course, the values of $k$ and $\epsilon$ can be manipulated to cause quick or gradual approach to the two thresholds.

activated, e.g., 1000, 0100, 0010, and 0001 (cf. Figure 2.3). The learner continues to activate and prune hypotheses in this fashion until it selects the correct hypothesis, and may go through several expansions. For example, consider the situation in Figure 2.4; following the initial expansion from the root node, the learner pruned all its active hypotheses and then activated the hypotheses that are 1-adjacent to the last active node, 0001.

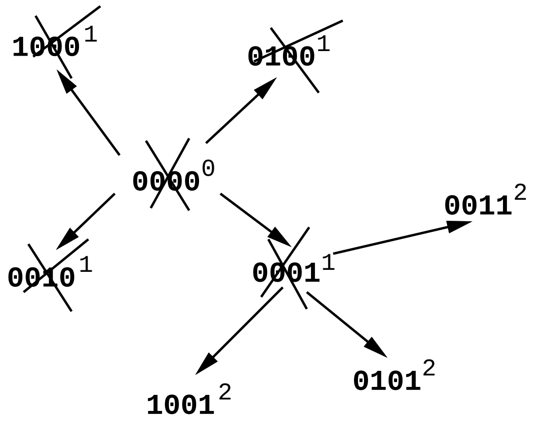

Figure 2.4: **Further Hypothesis Activation**

Although the learning algorithm is non-deterministic, the activation of new hypotheses is tightly constrained. The learner activates new hypotheses only when all of its current hypotheses are pruned. When new hypotheses are activated, the learner obeys the Single-Value Constraint, activating only those hypotheses that are 1-adjacent to the last active hypothesis. This implies that there are at most $n$ hypotheses active at any given time, where $n$ is the number of parameters to be learned. If more than one hypothesis is pruned at the same moment, leaving no active hypotheses, then one of the newly-pruned hypotheses is selected at random as the basis for activating new hypotheses.

## 2.4 The Subset Principle and Avoiding Overgeneralization

The Subset Principle, introduced in Section 1.1.4, states that a parameter should not be set to a superset value before it has been set to a subset value. One formalization of the Subset Principle is that it is a restriction against hypothesizing some language $L(h)$ if there exists some simpler $L(h') \subset L(h)$, and $L(h')$ has not yet been activated by the learner. I will refer to this formalization of the Subset Principle as the Strict Subset Principle.

Note that there is no *a priori* reason why a learner must order the activation of subset languages *before* the superset languages. The necessary condition is that the learner avoid overgeneralization. It has always been assumed that the learner cannot do so on positive evidence only if it activates a superset of the target language, and that the learner must always activate any subset languages first (cf. Berwick, 1985).

It follows that any learner which obeys the Strict Subset Principle must have knowledge of subset/superset relations in the search space of languages. It is interesting to note, however, that using such knowledge to order the hypotheses made by the learner includes the case where the learner recovers from generalization by selecting a subset language after a superset language has been activated.

In this model, I assume what I call the *Relaxed Subset Principle*:

> *The Relaxed Subset Principle*. If there are two languages, $L_1 \subset L_2$ and the learner encounters data that are consistent with both languages, then it it will not select the larger language $L_2$ before activating and discarding the smaller language, $L_1$.

The Relaxed Subset Principle allows the learner to avoid overgeneralization, since it forces the learner to activate any subset languages before selecting a superset language; the crucial difference is that the learner need not activate the subset language before activating the superset language. Knowledge of superset/subset relations can therefore be used later in learning, to "tighten up" an overly general hypothesis that might be selected.[5].

The Relaxed Subset Principle is implemented in the learner as follows. When some hypothesis $h$ receives a weight greater than $(1 - \epsilon)$, the learner activates any subset languages $L(h') \subset L(h)$ that have not yet been activated. Since the learner does not cease operation until it holds a single hypothesis with a weight greater than $(1 - \epsilon)$, this has the effect of delaying selection, since more than one hypothesis will be active.

Furthermore, if a hypothesis receives a weight greater than $(1 - \epsilon)$, then any active languages that are supersets of that language are pruned.

Together, these two mechanisms combine to avoid overgeneralization. The first ensures that all subsets of a successful hypothesis will be tried before the learner selects that hypothesis; the second ensures that superset languages will be pruned if there exists a successful subset language that can account for the data.

## 2.5 Dependencies Between Parameter Values

In some theories, parameters can influence other parameters in what values they may be set to. For example, in the metrical theory proposed by Dresher and Kaye, there are several different parameter value combinations that are explicitly not allowed, because they do not derive valid stress systems (cf. Section 3.2.6 for a full discussion).

In order to support this notion, the learning algorithm supports the use of a *de-*

[5]Consider the case where the Strict Subset Principle must be satisfied. If the learner was ready to select some $L(h)$, but there existed some $L(h') \subset L(h)$, then the learner would have to pick $L(h')$ instead; if in turn $L(h')$ has some subset $L(h'')$, then the learner would be forced to pick $L(h'')$, etc. Since no evidence has accumulated for $L(h)$, the learner has no reason to explore all of its subsets in advance, since $L(h)$ might be ruled out by the data very quickly.

It is also the case that the Strict Subset Principle predicts that children never overgeneralize; however, as noted by various researchers (e.g., (Rumelhart and McClelland, 1986; Ullman, et al. 1989; Hyams, 1986), children do make and recover from overgeneralizations. This sort of behavior is predicted by the Relaxed Subset Principle, since it allows a general language to be activated and then pruned in favor of a less general one.

*pendency table*. Each row in the dependency table is for a particular parameter, and contains two entries, a 0 entry and a 1 entry. Each entry contains a subtable which indicates parameter values which that value of the parameter depends on. For example, if the 1 entry for $P_0$ contains a table with the entry $(2 = 0)$, then $P_0$ cannot be set to 1 unless $P_2$ is set to 0.

The dependency table is used by the learning algorithm to check for well-formedness when new hypothesis are generated.

## 2.6 The Learning Algorithm

The data structures and algorithms for each of the modules of the learner are shown in Figures 2.5 - 2.10.

Figure 2.5 contains the first part of the main procedure, `LEARN`. The variables internal to the procedure are declared, and values are assigned to $\epsilon$ and to the sigmoid coefficients. The list of nodes to be searched (`Active-List`) is initialized by generating a root node for the hypothesis graph, of the same length as the target hypothesis and with weight 0.

Figure 2.6 contains the main loop of the main procedure `LEARN`. Each iteration through the loop corresponds to the processing of a single input example by the learner. First, the `Pruned-List` is initialized; then an input is read in according to the application-specific routine `Generate-Input-Datum`. Then, for each hypothesis node in the `Active-List`, the learner performs the following steps[6]:

1. Call the application-specific procedure `Parse` to determine if the current hypothesis can account for the example;

[6]The semantics of the `FOR` statement implementing the main loop of the program are crucial to successful operation of the algorithm. In this case, `FOR` *does not* copy its list argument, but maintains a pointer directly into `Active-List`. This is necessary in the case where the processing of one hypothesis node causes a node later in the `Active-List` to be pruned; in this case, if `FOR` made a copy of its argument, then the algorithm would still process the node that had already been removed from the `Active-List`.

2. If yes, then add positive weight to the hypothesis according to `Sigmoid-PositiveCoefficient`; otherwise, subtract weight from the hypothesis according to `SigmoidNegativeCoefficient`;

3. If the weight of the hypothesis dips below $(\epsilon - 1)$, remove it from the `Active-List` and add it to the `Pruned-List`;

4. If the weight of the hypothesis rises above $(1 - \epsilon)$, perform the following steps:

   - Determine any hypotheses that generate subset languages of the current hypothesis, and unless they are already active, add them to the `Active-List`;
   - Determine any hypotheses that generate superset languages of the current hypothesis, and if they are active, remove them from the `Active-List` and add them to the `Pruned-List`;
   - If the current hypothesis is the only node remaining in the `Active-List` following these steps, then select it as the final hypothesis.

5. If the `Active-List` is empty after these steps are performed, then pick a node from the `Pruned-List` and activate the hypotheses that are 1-adjacent to it (adding them to the `Active-List`).

The learner continues to process input examples until it selects a hypotheses and terminates when that hypothesis is the only hypothesis in the active list.

Figure 2.7 contains the definition for the `HNode` data structure and the definitions of `Weight-Positive` and `Weight-Negative`, the two routines that add and subtract weight from hypotheses.

Figure 2.8 illustrates the implementation of the table of parameter dependencies, and gives the definition of `Well-Formed?`, a routine used by the learner to check if new hypotheses are well-formed with respect to these dependencies (the dependencies active in a particular simulation are application specific).

Figure 2.9 shows the definitions of `Adjacent-Nodes`, `Subset-Nodes`, and `Superset-Nodes`, three routines used by the learner to create (well-formed) new hypotheses.

Figure 2.10 contains the definition for `Pick-Pruned-Node`, which is called when the `Active-List` is empty and the learner must expand its search; it also contains a description of the application-specific procedures `Generate-Input-Datum` and `Parse`, which must be defined for each application of the learner, and a description of the generic procedures referenced by the pseudocode descriptions in the figures.

```
PROCEDURE Learn (Target, Subset-Parameters, Dependencies);

 /* Target is the hypothesis representing the target grammar;
    Subset-Parameters is a list of integers indicating which of
     the parameters in the hypothesis are subset parameters;
    Dependencies is a table capturing dependencies between
     parameter values. */

 DECLARE Node, Active-List, Pruned-List, Inactive-List,
         Input, Parse-Result, Epsilon;

/* Node:          A hypothesis node being tested against the input example;
   Active-List:   The currently active hypothesis nodes;
   Pruned-List:   Hypothesis nodes pruned on the current input example;
   Inactive-List: All hypothesis nodes that have been pruned;
   Input:         The current input example;
   Parse-Result:  1 if the current hypothesis successfully parses, 0 if not;
   Epsilon:       (1 - Epsilon) is the selection threshold,
                  (Epsilon - 1) is the pruning threshold.  */

 DECLARE SigmoidNegativeCoefficient, SigmoidPositiveCoefficient;

/* SigmoidNegativeCoefficient: Supplied as the coefficient to the sigmoid
                               function when ParseResult = 0;
   SigmoidPositiveCoefficient: Supplied as the coefficient to the sigmoid
                               function when ParseResult = 1.
   K: Sigmoid damping factor.  */

 Epsilon <- .01;

 SigmoidNegativeCoefficient <- 5; /* Negative Examples Have 5x the Effect */
 SigmoidPositiveCoefficient <- 1; /*   of Positive Examples               */
 K <- .1;

 Active-List <- Make-HNode (Generate-Root-Node(Target) , 0.0);
```

Figure 2.5: **Learning Algorithm (Main Program: Part I)**

```
 LOOP;
  Pruned-List <- ();
  Inactive-List <- ();
  Input <- Generate-Input-Datum (Target);
  FOR Node IN Active-List DO:

     Parse-Result <- Parse (Input, HNode.Hypothesis(Node));

     IF Parse-Result = 0
        THEN Weight-Positive (Node, SigmoidPositiveCoefficient, K);
        ELSE Weight-Negative (Node, SigmoidNegativeCoefficient, K);
     ENDIF;
     IF HNode.Weight (Node) < Epsilon - 1 /* Prune Unsuccessful Node */
        THEN Remove (Node, Active-List);
             Push (Node, Pruned-List);
     ENDIF;
     IF HNode.Weight (Node) > 1 - Epsilon /* Try to Select a Hypothesis */
        THEN
             /* Activate Inactive Subsets. */

             FOR Subset IN
                  Subset-Nodes (Node, Subset-Parameters, Dependencies) DO:
                 UNLESS OR ( Member (Subset, Active-List),
                             Member (Subset, Inactive-List ) DO:
                  Push (Subset, Active-List);
             ENDFOR;

             /* Prune Active Supersets. */

             FOR Superset IN
                  Superset-Nodes (Node, Subset-Parameters, Dependencies) DO:
                 IF Member (Superset, Active-List)
                    THEN Remove (Superset, Active-List);
                         Push (Superset, Pruned-List);
                 ENDIF;
             ENDFOR;

             /* If Only One Node Remains Active, Select It. */

             IF Equal (Length (Active-List), 1) THEN Return (Node);
     ENDIF;
  ENDFOR;
  FOR Node in Pruned-List DO:
       Push (Node, Inactive-List);
  IF Empty (Active-List) /* Activate 1-Adjacent Hypotheses. */
     THEN Active-List <-
           Adjacent-Nodes (Pick-Pruned-Node (Pruned-List),
                           Subset-Parameters, Dependencies);
  ENDIF;
 END;
END;
```

Figure 2.6: **Learning Algorithm (Main Program: Part II)**

```
STRUCTURE HNode (Hypothesis, Weight); /* a two-cell data structure */

PROCEDURE Generate-Root-Node (target); /* same length as target, all 0's */
 DECLARE Result;
 Result <- Copy (target);
 FOR i from 1 to Length (Result) DO:
  Bit (Result, i) <- 0;
 ENDFOR;
 RETURN (Result);
END;

PROCEDURE Weight-Positive (node, c, k); /* add weight to a hypothesis */
 DECLARE NPE;
 NPE <- Sigmoid-Inverse (HNode.Weight (node));
 NPE <- NPE + c;
 HNode.Weight (node) <- Sigmoid (NPE, k);
END;

PROCEDURE Weight-Negative (node, c, k); /* remove weight from a hypothesis */
 DECLARE NPE;
 NPE <- Sigmoid-Inverse (HNode.Weight (node));
 NPE <- NPE - c;
 HNode.Weight (node) <- Sigmoid (NPE, k);
END;

/* Assume routines Sigmoid and Sigmoid-Inverse which compute
   the appropriate sigmoid weighting function and its inverse. */
```

Figure 2.7: **Learning Algorithm (Subroutines: Part I)**

```
/* The Dependency-Table keeps track of which parameter values
   depend on other parameter values. The entry for each parameter
   has two parts, dependencies on its zero value and dependencies
   on its one value, which are represented as subtables.

    Example:

      Parameter      0 value         1 value
      --------------------------------------
          1           2 = 0           2 = 1
                      3 = 1           3 = 0

   In the example above, if P1 = 0, then P2 must equal 0 and P3
   must equal 1; if P1 = 1, then P2 must equal 1 and P2 must
   equal 0. Checking to see if a hypothesis is "well-formed" means
   checked to see if it satisfies all of the dependencies between
   parameter values.  */

PROCEDURE Well-Formed? (Hypothesis, Dependency-Table)
 DECLARE Dependencies
 FOR i from 1 to Length (Hypothesis) DO:
  Dependencies <- Table-Lookup (i, Dependency-Table);
  IF Bit (Hypothesis, i) = 0
    THEN Dependencies <- Table-Lookup (0, Dependencies);
         FOR d in Dependencies DO:
           UNLESS Bit (Hypothesis, First (d)) = Second (d) DO:
             RETURN (FALSE);
         ENDFOR;
  ENDIF;
  IF Bit (Hypothesis, i) = 1
    THEN Dependencies <- Table-Lookup (1, Dependencies);
         FOR d in Dependencies DO:
           UNLESS Bit (Hypothesis, First (d)) = Second (d) DO:
             RETURN (FALSE);
         ENDFOR;
  ENDIF;
  RETURN (TRUE);
END;
```

Figure 2.8: **Learning Algorithm (Subroutines: Part II)**

```
PROCEDURE Adjacent-Nodes (Hypothesis, Dependencies);
 DECLARE Temp, Result;
 FOR i from 1 to Length (Hypothesis) DO:
  Temp <- Copy (Hypothesis);
  IF Bit (Hypothesis, i) = 0
    THEN  Bit (Copy, i) <- 1;
    ELSE  Bit (Copy, i) <- 0;
  ENDIF;
  IF Well-Formed? (Copy, Dependencies)
    THEN Push (Make-HNode (Copy, 0.0), Result);
  ENDIF;
 ENDFOR;
 RETURN (Result);
END;

PROCEDURE Subset-Nodes (Hypothesis, Subset-Parameters, Dependencies);
 DECLARE Temp, Result;
 FOR p in Subset-Parameters DO:
  IF Bit (Hypothesis, p) = 1
    THEN Temp <- Copy (Hypothesis);
         Bit (Temp, p) = 0;
         IF Well-Formed? (Temp, Dependencies)
           THEN Push (Make-HNode (Temp, 0.0), Result);
         ENDIF;
  ENDIF;
 ENDFOR;
END;

PROCEDURE Superset-Nodes (Hypothesis, Subset-Parameters, Dependencies);
 DECLARE Temp, Result;
 FOR p in Subset-Parameters DO:
  IF Bit (Hypothesis, p) = 0
    THEN Temp <- Copy (Hypothesis);
         Bit (Temp, p) = 1;
         IF Well-Formed? (Temp, Dependencies)
           THEN Push (Make-HNode (Temp, 0.0), Result);
         ENDIF;
  ENDIF;
 ENDFOR;
END;
```

Figure 2.9: **Learning Algorithm (Subroutines: Part III)**

```
PROCEDURE Pick-Pruned-Node (Pruned-List);
 DECLARE Result;
 IF Length (Pruned-List) = 1 THEN
   Result <- First (Pruned-List);
   ELSE Result <-
         List-Lookup (Pruned-List,
                      Random (1, Length (Pruned-List)));
 ENDIF;
 RETURN Result;
END;


APPLICATION-SPECIFIC PROCEDURES

 Generate-Input-Datum (hypothesis):
  Generates a single input example that is in L(hypothesis)

 Parse (datum, hypothesis):
  Sets parameters according to hypothesis, parses datum, and returns
  1 if the parse produced a legal structure and 0 otherwise.


GENERIC PROCEDURES

 Make-HNode ():             Creates an empty HNode data structure
 Length (datum):            Returns the length of a list or vector
 Remove (datum, list):      Removes a datum from a list
 Push (datum, list):        Destructively adds a datum to the front of a list
 Pop (datum, list):         Destructively removes a datum from the front of
                             a list
 Member (datum, list):      Returns T if datum is in a list, NIL otherwise
 Empty (list):              Returns T if list has length 0, NIL otherwise
 Copy (datum):              Returns a copy of a list or vector
 Bit (vector, i):           Returns the vector bit at offset i
 Table-Lookup (key, table): Returns the table entry at key in table
 List-Lookup (n, list):     Returns the nth member of list
```

Figure 2.10: **Learning Algorithm (Subroutines: Part IV)**

## 2.7 A Simple Example

A simple example which illustrates the operation of the learning algorithm is shown in Appendix A[7]. In the traced example, the target hypothesis is 0101, and the learner begins with the least-marked hypothesis, 0000. To simulate learning, each "input example" considered by the learner is just the target hypothesis itself; each active hypothesis is therefore compared with the target hypothesis to simulate parsing. As a result, no hypotheses but the target hypothesis will receive positive weight.[8]

After 6 cycles, hypothesis 0000 has accumulated a weight of $-0.905146$, and is pruned. The nodes which are 1-adjacent to 0000 are activated. By cycle 12, each of these nodes has received a negative weight below the threshold for pruning, so they are all pruned. One hypothesis, 0100, is picked at random and its 1-adjacent nodes are activated. One of the nodes activated, 0101, is the target hypothesis; therefore, it begins to accumulate positive weight. After cycle 18, the other active nodes have accumulated enough negative weight to be pruned. Finally, after several more cycles, node 0101 accumulates a weight of 0.905153, and the learner selects 0101 in cycle 42.

## 2.8 Learnability Problems Revisited

As discussed in Section 1.4.2, a strictly deterministic error-driven model fails to select the correct hypothesis in certain situations. In this section, I return to a discussion of those problems and show how the proposed model can overcome them. I also present a learning scenario in which the learner uses its implementation of the Relaxed Subset Principle to avoid overgeneralizing when it has selected a superset of the target language.

[7]In this example, only positive examples $s \in L(h_{target})$ are presented to the learner.

[8]This learning scenario is simplified for the purposes of illustration; In Chapter 3, I present more empirically relevant examples.

### 2.8.1 Parameter Flipping and Noisy Examples

| **Input Example** | **Hypothesis** |
|---|---|
| $\vdots$ | $\vdots$ |
| $s_m \in L(h)$ | $W(h) = .001$ |
| $s_{m+1} \in L(h)$ | $W(h) = .025$ |
| $\vdots$ | $\vdots$ |
| $s_n \in L(h)$ | $W(h) = .898$ |
| $s_{n+1} \notin L(h)$ | $W(h) = .887$ |
| $s_{n+2} \in L(h)$ | $W(h) = .898$ |
| $\vdots$ | $\vdots$ |

Figure 2.11: **Learning in the Presence of Ungrammatical Examples**

Consider the learning scenario illustrated in Figure 2.11. Let $h$ be the hypothesis containing the correct parameter values, and let $L(h)$ be the set of inputs consistent with that hypothesis. Assume that $L(h)$ is the language to be learned. Presumably, at some time $m$ the learner would encounter an example $s_m \in L_h$, and begin to accumulate positive weight for $h$. After many more examples are processed, the weight of $h$ will approach +1, as observed here at time $n$. Suppose that at time $n + 1$ the learner encounters $s_{n+1} \notin L_h$. Since the slope of the sigmoid weighting function is very shallow near its asymptotes (cf. Section 2.1), the learner will subtract only a small amount of weight from $h$. The learner does not therefore flip a parameter (change its hypothesis) on the basis of a single example; rather, it is the cumulative effect of larger amounts of data that tend to confirm or disconfirm particular hypotheses over time. This makes the learner much more resilient to the presence of infrequent ungrammatical examples, and it is able to overcome the problems that strictly error-driven deterministic learners must face when learning in the presence of ungrammatical examples.

The weighting strategy presented here was tested on a small learning problem containing two word order parameters and four hypotheses corresponding to the basic word orders (SOV, OVS, SVO, VOS). An experiment was conducted in which 200 grammatical examples of SVO sentences were presented to the learner. The weight associated with the correct hypothesis is shown in the graph in Figure 2.12. As predicted, the learner's confidence in that hypothesis grows steadily and approaches 1 (complete belief)[9].

Another experiment was conducted, in which another 200 examples were presented to the learner, but this time with a 10% error rate (roughly 1 out of every 10 examples was ungrammatical, i.e., it could not be parsed successfully as an SVO sentence). The weight associated with the SVO hypothesis is shown in Figure 2.13.

The learner was still able to select the correct hypothesis, but the level of weight assigned to the SVO hypothesis after 200 examples is somewhat less than when learning without errors, and the learner's belief in the SVO hypothesis can be seen to fluctuate near the origin more in Figure 2.13 than in Figure 2.12. Hence the model predicts that the learner should still be able to set parameters correctly when ungrammatical examples are present, but that it may require more data before selecting the correct hypothesis. I will return to the matter of learning with noisy data in Section 5, which presents the results of testing the learner's behavior on noisy samples along with a more formal analysis of the learner's robustness.

### 2.8.2 Non-Determinism and Hypothesis Ordering

Returning to the learning problem presented in Figure 1.7 (shown again in Figure 2.14), we can see how limited non-determinism can solve the hypothesis ordering problem. In this situation, the learner must make the correct choice when $H_1$ is pruned; recall that $H_2$ is the target hypothesis. When the learner decides that $H_1$ cannot account for the input data, it need not limit itself to picking either $H_2$ or $H_3$; since both are local to

[9]This learning problem is based on two parameters formulated by Gibson (1987) and discussed in (Nyberg, 1987); for more details concerning the experiment mentioned here, see (Nyberg, 1989).

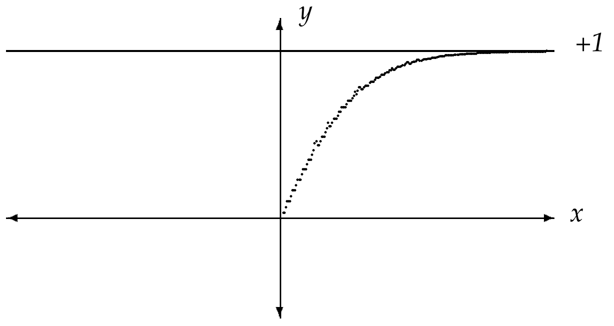

Figure 2.12: **Weight Assigned to Word Order SVO Over 200 Examples**

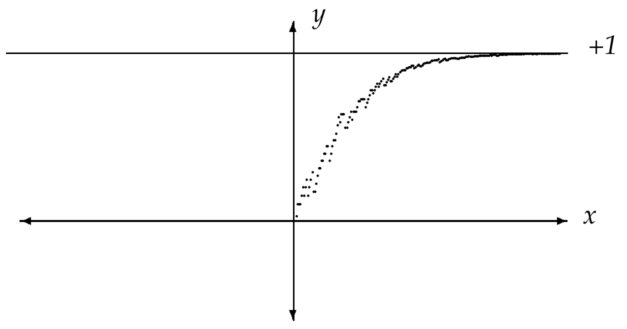

Figure 2.13: **Weight Assigned to Word Order SVO Over 200 Examples, 10% Error Rate**

$H_1$, it can activate both and assume that the evidence provided by subsequent input examples will serve to discriminate between them. In this case, the learner would activate both $H_2$ and $H_3$; after processing further input examples, the learner would prune $H_3$ and select $H_2$.

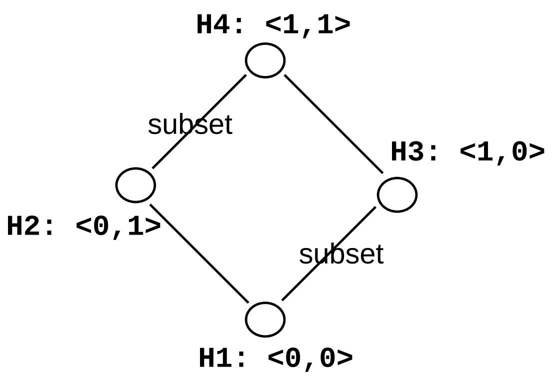

Figure 2.14: **A Simple Hypothesis Ordering Problem**

### 2.8.3 Non-Determinism and Subset Shift

Limited non-determinism can also help the learner to overcome the Subset Shift problem (cf. Section 1.4.2). Consider again the scenario illustrated in Figure 1.8 (shown again in Figure 2.15). Recall that the languages determined by $H_2$ and $H_3$ (English and Irish, respectively) overlap in that both can assign case to the subject of a non-finite clause. As a result, there will be a set of sentences for which the learner cannot decide (on the basis of a single example) whether the language is an exceptional case-marking language (English, $H_2$) or a structural case-marking language (Irish, $H_3$). When faced with this learning scenario, a strictly deterministic learner must choose between $H_2$ and $H_3$ when $H_1$ is pruned.

The limited non-deterministic learner does not fall prey to the Subset Shift problem, for the reasons mentioned in the previous section. When $H_1$ is pruned, the learner

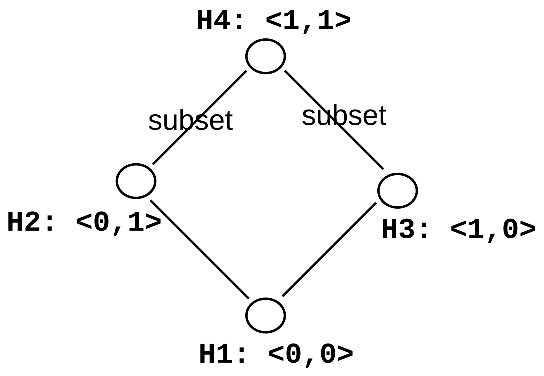

Figure 2.15: **The Subset Shift Problem**

can activate both $H_2$ and $H_3$. On the basis of further data (which allow the learner to distinguish between English and Irish based on the values of other parameters; (Clark, 1990a)), the learner will prune whichever language is not the target language.

## 2.8.4 Avoiding Overgeneralization

As discussed in Sections 2.4 and 2.6, the learner uses the Relaxed Subset Principle to recover from overgeneralization. Before selecting some language $L(h)$, the learner first activates any $L(h') \subset L(h)$. If $L(h')$ receives enough positive weight greater than $(1 - \epsilon)$, then $L(h)$ will be pruned.

This mechanism is illustrated in Appendix B, which shows a trace of the learner as it solves the learning problem shown in Figure 2.16. Figure 2.16 shows a set of embedded languages from the perspective of hierarchical subset relations (at left) and set inclusion (at right). The target language is $L(H7)$, which is deeply embedded inside the initial hypothesis language $L(H1)$.

Turning to the trace in Appendix B, we can follow the steps of the learner as it reduces its overly general hypothesis to the correct hypothesis[10]. At cycle 3, hypothesis

[10]In this example, the weighting coefficient $k$ has been set to $-5$ for $s \notin L(h)$ and to 10 for $s \in L(h)$.

$H1$ achieves a weight higher than $(1-\epsilon)$; as a result, its two subset languages, $H2$ and $H3$, are activated. During cycle 5, $H2$ achieves a weight above the threshold, and its inactive subset, $H4$, is activated; $H1$, a superset of $H2$, is deactivated. $H3$ also achieves a weight above the threshold, and has identical subset/superset relations with $H4$ and $H1$; but no further action occurs since $H4$ has already been activated and $H1$ has already been pruned.

During cycle 7, $H4$ achieves a weight above the threshold; its inactive subsets, $H5$ and $H8$, are activated, and its active supersets, $H3$ and $H2$, are pruned. During cycle 9, $H5$ achieves a weight above the threshold; its inactive subsets, $H6$ and $H7$, are activated, and its active superset, $H4$, is pruned. $H8$, which is not a superset of the target language, receives enough negative weight and is pruned during this cycle.

Finally, during cycle 11, $H5$, $H6$ and $H7$ are active. Since $H6$ receives enough negative weight, it is pruned. $H7$, on the other hand, reaches the $(1-\epsilon)$ threshold. Its active superset, $H5$, is pruned. Since $H7$ has no subset languages that it must activate, the learner has selected a single hypothesis, $H7$.

## 2.9 Summary

In this chapter, I have presented a constrained non-deterministic model of parameter setting that overcomes problems inherent in strictly-deterministic error-driven models. Because this model weighs evidence both for and against hypotheses using a sigmoidal weighting function, it is resilient to the presence of ungrammatical examples. Non-determinism allows the learner to hold more than one hypothesis at each time $t$, thus avoiding the hypothesis ordering and Subset Shift problems. However, non-determinism is strictly limited, placing an upper bound on the number of active hypotheses. At any time $t$, the learner may have only $n$ active hypotheses, where $n$ is

This allows the learner to select the target hypothesis more quickly than usual, for the purposes of illustration. With the coefficients set to their usual values ($-5$ and 1, respectively) the learner requires 146 cycles to select the correct language.

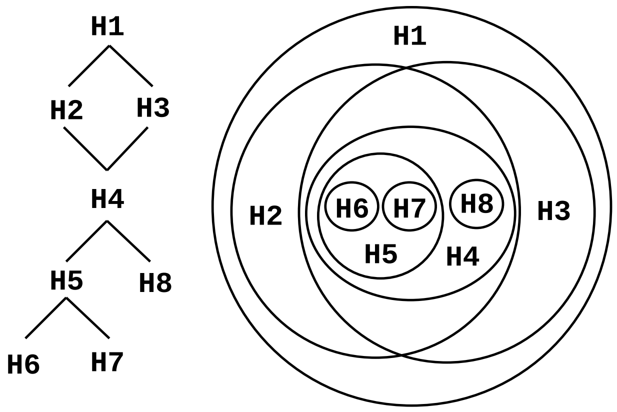

Figure 2.16: **A Subset Learning Problem**

the number of parameters to be learned.

# Chapter 3

# Learning Phonological Parameters

To test the learning algorithm presented in the previous chapter, I have replicated the phonological parameters learned by the YOUPIE system (Dresher and Kaye, 1990) in a computer program that assigns stress to sequences of syllables according to the values of a set of 11 parameters. The program builds syllables from a set of phonological segments and groups them together into metrical constituents (also referred to as *feet*), which are then arranged in a *word tree*, to which primary stress is assigned.

The goal of this empirical study is to show that the limited non-deterministic model can successfully learn the parameters of metrical phonology from varied sets of data. In general, the parameters of metrical phonology and the effect that they have on the stress patterns of words are drawn directly from the Dresher and Kaye parameter model, with a few refinements. Of course, the learning algorithm employed here is completely different.

In the following sections, I describe the metrical theory adopted in this system, and show how it is implemented and integrated with the limited non-deterministic learning algorithm. I then turn to a description of the data used to test the learner, and present the results of testing the learner on both natural languages and the full set of 432 languages derived from the 11 parameters and their dependencies[1].

[1]The dependencies between the 11 parameters and their values are discussed in Section 3.2.6.

## 3.1 Metrical Theory

Metrical theories of phonological stress have focussed on the construction of syllables and syllable constituents within individual words, in order to predict how stress is assigned to those constituents and hence to words themselves[2]. Different structures (trees, grids, combinations of trees and grids) have been proposed in theories of metrical constituents (Hayes, 1981; Prince 1983; Liberman and Prince, 1977). I adopt a model that is almost identical to the model discussed in (Dresher and Kaye, 1990), which uses only metrical trees to assign stress to syllables in the word. In the remainder of Section 3.1, I summarize the metrical system they propose, and illustrate the main characteristics of their treatment of stress (for full details, the reader is referred to (Dresher and Kaye, 1990, pp. 142-147)).

### 3.1.1 Syllable Structure

Syllables are built from a stream of phonetic segments containing vowels and consonants. Each vowel has an associated sonorant quality, which is encoded as an integer. A value of 2 indicates primary stress; a value of 1 indicates secondary stress; and a value of 0 indicates no stress. The raw data presented to the learner are strings of letters and integers; the letters indicate the segments, and the integers follow each vowel and indicate its stress value.

Examples are shown in Figure 3.1. The representation of *Vancouver* indicates that the second syllable receives primary stress, while the first syllable receives secondary stress and the final syllable is completely unstressed. The representation of *algebra*, on the other hand, indicates that the first syllable recieves primary stress, and that the other syllables are unstressed[3].

[2]As in (Dresher and Kaye, 1990), I limit my focus to stress assignment within the word, and do not consider the effects of phrasal syntax or lexical exception on stress assignment.

[3]These examples and the corresponding trees shown in Figure 3.3 are taken from (Dresher and Kaye, 1990, p. 141).

```
v a 1 n c o 2 u v e 0 r        (Vancouver)

a 2 l g e 0 b r a 0            (algebra)
```

Figure 3.1: **Sample Data: Segments and Stress Values**

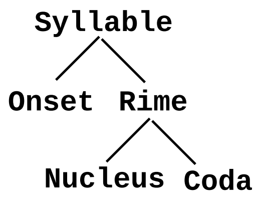

Figure 3.2: **Internal Structure of the Syllable**

Syllables themselves have internal structure, and must be constructed from the raw stream of segments. The theory assumed here holds that segments are arranged into tree structures for each syllable. A syllable may contain an onset, nucleus, and coda, arranged as shown in Figure 3.2. Syllables that begin with a vowel, such as the first syllable of *algebra*, lack an onset; open syllables that end in a vowel, such as the final syllable of *algebra*, lack a coda. A syllable must have a nucleus, but need not have an onset or coda. The nucleus always contains a vowel, which is in general the most sonorant segment in the syllable and is the segment upon which stress devolves. The syllable structures of *Vancouver* and *algebra* are shown in Figure 3.3.

### 3.1.2 Metrical Constituents and Secondary Stress

Once the segments in a word have been arranged into syllable structures, the syllables are grouped into metrical constituents, also referred to as *feet*. Languages that have

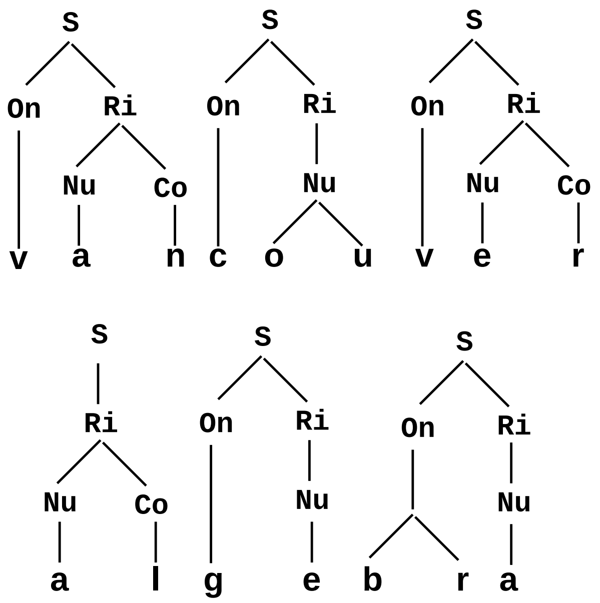

Figure 3.3: **Example Syllable Structures**

rhythmic repeating stress patterns (like English) have words constructed from *binary feet*, or metrical constituents containing just two syllables per foot. Some languages have fixed stress, assigning stress to just a single syllable in the word. Such words are constructed from *unbounded feet*, single metrical constituents containing all the segments in the word[4]. Examples of binary and unbounded feet are shown in Figure 3.4. Each foot has one segment which is marked as the *head*, the segment in the foot which receives the most stress. An asterisk denotes secondary stress assigned to the head of each foot[5].

```
(a* pa) (la* chi) (co* la)   Binary Feet (English)

(e ga li te*)                Unbounded Foot (French)
```

Figure 3.4: **Examples of Binary and Unbounded Feet**

The construction of binary or unbounded feet, as well as the assignment of stress to the head of each foot, is controlled by the values of the following constituent construction parameters:

- *Direction of Constituent Construction ($P_1$).* If $P_1$ is 0, then metrical feet are constructed from left to right; if $P_1$ is 1, feet are constructed from right to left.
- *Headedness of Constituents ($P_2$).* If $P_2$ is 0, then constituents (feet) are left-headed; if $P_2$ is 1, they are right-headed.
- *Boundedness of Constituents ($P_3$).* If $P_3$ is 0, then constituents (feet) are bounded (binary); if $P_3$ is 1, then constituents are unbounded in size.

[4]For the moment, I do not consider the effects of extrametricality, which will be discussed shortly.
[5]Primary stress is assigned through construction of the word tree, to be discussed below.

These parameters combine to derive $2^3 = 8$ different basic foot types. The different basic types of metrical feet constructed for words containing 5 syllables are illustrated in Figure 3.5[6].

| $P_1$ | $P_2$ | $P_3$ | Metrical Feet |
|---|---|---|---|
| 0 | 0 | 0 | $((\sigma^* \ \sigma)\ (\sigma^* \ \sigma)\ (\sigma^*))$ |
| 0 | 0 | 1 | $((\sigma^* \ \sigma \ \sigma \ \sigma \ \sigma))$ |
| 0 | 1 | 0 | $((\sigma \ \sigma^*)\ (\sigma \ \sigma^*)\ (\sigma^*))$ |
| 0 | 1 | 1 | $((\sigma \ \sigma \ \sigma \ \sigma \ \sigma^*))$ |
| 1 | 0 | 0 | $((\sigma^*)\ (\sigma^* \ \sigma)\ (\sigma^* \ \sigma))$ |
| 1 | 0 | 1 | $((\sigma^* \ \sigma \ \sigma \ \sigma \ \sigma))$ |
| 1 | 1 | 0 | $((\sigma^*)\ (\sigma \ \sigma^*)\ (\sigma \ \sigma^*))$ |
| 1 | 1 | 1 | $((\sigma \ \sigma \ \sigma \ \sigma \ \sigma^*))$ |

Figure 3.5: **Basic Constituent Types, Encoded by 3 Parameters**

### 3.1.3 The Word Tree and Primary Stress

The *word tree* is simply a bundle of metrical feet; in the examples shown in Figure 3.5, the outer set of parentheses delimiting the set of metrical feet could be said to represent the word tree. Primary stress is assigned according to the word tree, and is controlled by the parameter $P_0$:

- *Primary Stress Assignment ($P_0$).* The word tree is either strong on the left or right; if $P_0$ is 0, then the word tree is strong on the left; if $P_0$ is 1, then the word tree is strong on the right. Primary (word) stress is assigned to the head of either the

[6]In general, not all combinations of parameters derive coherent stress systems. For example, the Directionality parameter becomes irrelevant when unbounded feet are constructed; in such cases, the Headedness parameter alone is enough to locate word stress. I return to a discussion of such parameter dependencies in Section 3.2.6.

leftmost or rightmost foot, accordingly[7].

As an example, consider the word *Apalachicola* and the two possible assignments of word stress illustrated in Figure 3.6. In both cases, binary feet are constructed. In the first example, word stress devolves on the head of the leftmost constituent (primary stress is indicated by a double asterisk). In the second example, word stress devolves on the head of the final constituent.

```
((a** pa) (la* chi) (co* la))    Initial Word Stress

((a* pa) (la* chi) (co** la))    Final Word Stress
```

Figure 3.6: **Examples of Initial and Final Word Stress**

The basic rules of constituent construction and stress assignment are captured by $P_0$, $P_1$, $P_2$, and $P_3$. The segments in a word are arranged into syllables, which are then arranged into metrical constituents; the heads of these constituents are marked with secondary stress. Then primary stress is assigned according to the word stress parameter.

There are other phenomena which affect stress assignment and considerably complicate the process of constituent construction. The other parameters in the system proposed by Dresher and Kaye deal with exceptions to the basic rules of stress assignment described by the first four parameters. These phenomena and their associated parameters are discussed below.

[7]Note that there is no need to construct a word tree in the case of unbounded feet; primary stress devolves on the head of the single unbounded foot, which spans the entire word.

### 3.1.4 Quantity Sensitivity

Syllables can be classed as “light”, “heavy”, or “super-heavy”, depending on the types and numbers of segments they contain (Hayes, 1981; Halle and Vergnaud, 1987; Levin, 1985)[8]. In the examples I have presented so far, the process of stress assignment is indifferent to the presence of heavy syllables in the word; all syllables are treated equally in a system that contains just parameters $P_0$ through $P_3$. However, in some languages stress assignment proceeds differently when heavy syllables are present. Such languages are called *quantity-sensitive*. In a quantity-sensitive language, certain heavy syllables must receive stress, regardless of where they appear in the segment stream. In other words, the presence of heavy syllables may disrupt the regular pattern of stress assignment determined by the values of $P_0$ through $P_3$.

Quantity-sensitivity is described by the following parameters:

- *Quantity Sensitivity ($P_4$).* If $P_4$ is 0, then metrical feet are not quantity sensitive; if $P_4$ is 1, then feet are quantity sensitive.

- *Sensitivity to Rime or Nucleus Branching ($P_5$).* If $P_5$ is 0, then branching (heavy) rimes must be stressed; if $P_5$ is 1, then branching nuclei must be stressed.

When feet are sensitive to branching rime structures, then the presence of a coda in the rime (implying a branching rime) results in a syllable that is always stressed. When feet are sensitive to branching nuclei, the presence of a long vowel in the nucleus results in a syllable that is always stressed.

As stated above, the usual process of stress assignment can be disrupted in languages that are quantity sensitive. Consider the Maranungku examples in Figure C.3[9]. In the first example, the word *merepet* receives the stress pattern (201) as a result of the construction of two left-headed binary feet and primary stress assignment to the

[8]For example, long vowels, consonant clusters, or final consonants in a syllable can all be considered “heavy” in various languages.

[9]The examples of metrical structure referred to in this section can be found in Appendix C.

left-most foot[10]. In the second example, the word *yangarmata* receives the stress pattern $(2010)$ through a similar process.

For the moment, assume that Maranungku is quantity sensitive to branching rime structures. If that were the case, the word *merepet* would still receive a stress pattern of $(201)$, since the syllable with a branching rime (the final syllable) can be stressed without disrupting the usual process of constituent construction (it receives stress as the head of the second binary foot, and satisfies the constraint that branching rimes must be stressed). However, the word *yangarmata* has two heavy syllables (the first two syllables), and the normal process of constituent construction violates the quantity sensitivity constraint, since it results in a constituent structure that does not stress the second syllable of the word, which contains a branching rime (cf. Figure C.3). Since both branching syllables must be stressed, the stress pattern $(2101)$ is produced. Since the second syllable cannot be placed in a binary foot with the first syllable, the first syllable forms a unary foot and binary foot construction continues from the second syllable (see Figure C.5)[11].

There are also languages where secondary stress may be assigned *only* to heavy syllables (note that parameters $P_4$ and $P_5$ do not preclude assignment of stress to light syllables during foot construction). This implies that only heavy syllables can be the head of a foot, which occurs in languages such as Aguatec Mayan. To account for this distinction, Dresher and Kaye propose an additional parameter:

- *Branching (Heavy) Constituent Heads ($P_6$).* If $P_6$ is 0, then the head of a foot (a syllable receiving secondary stress) need not branch (may be light); if $P_6$ is 1, then the head of a foot must branch (must be heavy).

As noted by Dresher and Kaye, the presence of quantity-sensitivity has interesting ramifications for language learning. Consider the Maranungku example given above.

[10]When only a single syllable remains at a word boundary during foot construction, that syllable is placed in a foot by itself.

[11]I adopt the notion that a foot may have only a single head (Halle and Vergnaud, 1987).

For 4-syllable words that don't contain heavy syllables, a quantity-sensitive variant of Maranungku (call it "Maranungku-QS") would always assign a stress pattern of (2010). However, for 4-syllable words that contain heavy syllables, the stress pattern would be different. It is therefore the case that for every quantity-insensitive language, there exists a quantity-sensitive language that assigns exactly the same patterns to words in some cases, but different stress patterns in other cases. This means that in general, languages with $P_4$ set to 1 will be supersets of the same language with $P_4$ set to 0 and no heavy syllables. This implies that the problem of subset languages (cf. Section 1.1.4) must be addressed by a phonological parameter-setting learner, since choosing the wrong value of $P_4$ might cause the learner to incorrectly assume that the language being learned is quantity-sensitive when it is not.

For example, assume that the language to be learned contains only light syllables. Assume also that the language is not quantity-sensitive. Suppose that the learner picks a hypothesis with $P_4$ set to 1 (Quantity-Sensitive). If the learner were to encounter a word with heavy syllables, it would assign a quantity-sensitive stress pattern that would most likely differ from the pattern assigned if $P_4$ were 0. Unfortunately, since the language to be learned contains only light syllables, such data would not be forthcoming. As a result, the learner would have no clue that it had selected the wrong value for $P_4$, and it would most likely converge to an incorrect parameter settting.

### 3.1.5 Extrametricality

In some of the world's languages, the basic stress pattern cannot be captured in the system described by parameters $P_0$ through $P_6$. Consider the case of Lakota, an American Indian language (see Figure C.2). Lakota has a fixed stress pattern of the type associated with the construction of unbounded constituents; however, stress always devolves on the *second* syllable rather than the first or last syllable. With $P_1 = 1$ (indicating unbounded feet) and $P_2 = 0$ (indicating left-headed feet), we can account for languages with fixed initial stress, but not fixed second-syllable stress. The ability to "skip" a syllable has been referred to as *extrametricality*; a syllable that appears to fall outside the normal process of foot construction is said to be *extrametrical*[12].

The notion of extrametricality is captured by the following parameters:

- *Extrametricality ($P_7$).* If extrametrical syllables are allowed, then $P_7 = 1$; otherwise, $P_7 = 0$.
- *Direction of Extrametricality ($P_8$).* When $P_8 = 0$, the extrametrical syllable is at the left word boundary; when $P_8 = 1$, the extrametrical syllable is at the right word boundary.

### 3.1.6 Stress Clash

Some languages do not tolerate the presence of two adjacent syllables that both receive secondary stress. For example, consider the stress pattern assigned to the Warao word *yiwaranae* (see Figure C.4). Following construction of metrical feet, the word tree contains a unary foot followed by two binary feet. Since in Warao feet are left-headed, the head of the second foot is adjacent to the head of the initial foot. Apparently, Warao does not tolerate this sort of *stress clash*, and invokes a stress-avoidance rule that destresses a weak (unary, non-branching) foot in a stress clash. The resulting stress pattern, (01020), is derived from the word tree following stress clash avoidance, which destresses the initial (non-branching) foot[13].

[12]Extrametricality was introduced by Hayes (1981), and is constrained such that it is active only at the periphery of the word; at most a single syllable may be counted as extrametrical, either at the beginning or at the end of the word.

[13]The minus sign in Figure C.4 marks the foot that has been destressed. Stress clash avoidance has been accounted for by restructuring of the metrical tree (Hayes, 1981; Hammond, 1986) or by a process of movement within the metrical grid (Prince, 1983); I assume a destressing rule that simply removes stress from the weak foot in a stress clash, without restructuring the metrical tree.

| Parameter | Principle |
|---|---|
| P0 | The word-tree is strong on the [Left/Right] |
| P1 | Feet are [Binary/Unbounded] |
| P2 | Feet are built from the [Left/Right] |
| P3 | Feet are strong on the [Left/Right] |
| P4 | Feet are quantity sensitive (QS) [No/Yes] |
| P5 | Feet are QS to the [Rime/Nucleus] |
| P6 | A strong branch of a foot must itself branch [No/Yes] |
| P7 | There is an extrametrical syllable [No/Yes] |
| P8 | It is extrametrical on the [Left/Right] |
| P9 | A weak foot is defooted in clash [No/Yes] |
| P10 | Feet are noniterative [No/Yes] |

Figure 3.7: **Metrical Parameters**

### 3.1.7 Branching and Iterativity

As noted by Dresher and Kaye (and recently discussed by Halle (1990)), there are languages which assign stress to only a single foot in a word (like unbounded constituent construction), but where the number of syllables in the word affects the assignment of stress (like binary constituent construction)[14]. This distinction is captured by $P_{10}$:

- *Non-Iterativity ($P_{10}$).* If $P_{10} = 1$, then only the strongest foot in a word tree will receive stress, while the heads of other feet will not receive stress. If $P_{10} = 0$, then all heads receive secondary stress.

### 3.1.8 Summary

The parameters I have described in this section are summarized in Figure 3.7. Although this system represents only one of a number of ways of accounting for stress assignment

[14]In the examples discussed by Halle, it is clear that some examples of stress assignment seem to require the presence of secondary stresses to account for the placement of primary stress; however, the secondary stresses do not appear in the surface word.

rules, it provides a sufficiently rich test environment for the purposes of this thesis. In particular, it contains enough parameters that interact with each other in non-trivial ways to present a reasonably "tough" problem for a language learning algorithm[15].

The parameters proposed by Dresher and Kaye are intended to capture the core principles of metrical phonology. Each parameter setting captures one way of configuring the core principles of stress assignment. A complete characterization of a single phonological system must also include a description of peripheral phenomena such as lexical or exceptional stress patterns, language-specific destressing rules, etc. The model utilized here by no means captures all of the cross-linguistic variability in stress assignment; however, it does capture a large enough set of stress-related phenomena to be of interest from a linguistic perspective as well as a learnability perspective.

## 3.2 Implementation

I now turn to a discussion of how I have implemented the Dresher and Kaye parameter model in a computer program that assigns stress to words. The input to the stress processor is a set of unstressed segments; these are parsed successively into syllable structures, feet, and a word tree. Secondary and primary stress are assigned according to the rules of constituent construction and head location, as determined by the settings of parameters $P_0$ through $P_{10}$. The stress processor performs the following steps in assigning stress to a word (see Figure 3.8):

1. The segments (phones) in each syllable are parsed into explicit syllable trees, with the segments (if any) before the vowel forming the onset and the segments (if any) following the vowel (or vowels) forming the coda;

2. The syllables are parsed into feet. Syllables are assigned a default stress of 0 (unstressed);

[15]I return to a discussion of parameter interactions in Section 3.2.6.

3. The head of each foot is located and secondary stress (stress value 1) is assigned to each head;

4. The feet are arranged in a word tree and primary stress (stress value 2) is assigned to the head of the foot bearing primary stress;

5. If feet are non-iterative, then all secondary stress is removed, leaving only primary stress;

6. If feet are iterative and stress clash is not tolerated, non-branching (unary) feet adjacent to stressed segments are destressed[16].

Next, I discuss the details of the data structures and algorithms associated with each module in the stress processor.

### 3.2.1 Syllable Construction

Syllables are represented using CommonLisp structures, shown in Figure 3.9. The onset, nucleus and coda each contain a field called `phones` which stores the phonetic segments (phones) associated with that part of the syllable. The nucleus also has a `stress` field, which is filled by an integer indicating the level of stress: 0 (unstressed), 1 (secondary stress), or 2 (primary stress). The rime structure contains a nucleus and a coda (since a coda is optional, this field can be empty), and the syllable itself contains an onset (which is optional and may be empty) and a rime. The syllable structure also contains three flag fields, `word-stress?`, `defooted?`, and `heavy?`. The first two flags are marked when a syllable receives primary stress or is defooted, respectively, and are used by the program which prints out syllables to mark the printed representation of the syllable accordingly. The `heavy?` flag is set if the nucleus (or rime) is branching and the appropriate parameters are set, to facilitate

[16]Since non-iterative feet never bear secondary stress, it is not necessary to check stress clash when feet are non-iterative (cf. Section 3.1.7).

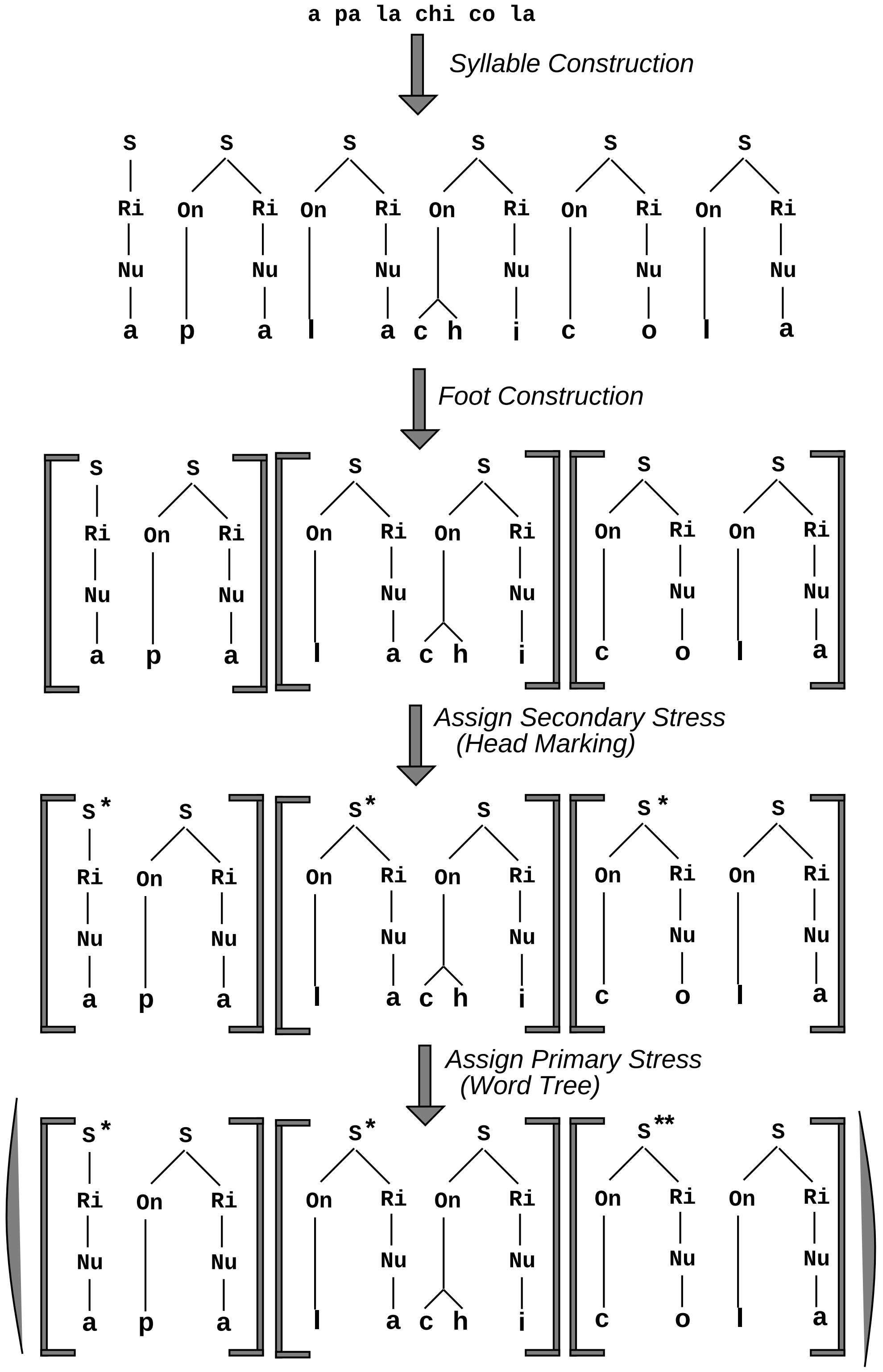

Figure 3.8: **Constructing the Stressed Word Tree from Input Segments**

subsequent quantity-sensitive constituent construction when necessary (these flags are not crucial to the operation of the stress processer, but add to the perspicuity of the implementation).

```
(defstruct syllable
  onset rime word-stress? defooted? heavy?)

(defstruct onset phones)

(defstruct rime nucleus coda)

(defstruct nucleus phones stress)

(defstruct coda phones)
```

Figure 3.9: **Syllable Structure Definitions**

The input data processed by the stress processor are raw word representations, which consist of lists of phones and an associated stress value. The job of the syllable constructor is to parse that list of phones into the appropriate syllable structure, and associate the stress value with the nucleus of the syllable. The actual representation of the input is slightly different from that used by Dresher and Kaye, and is illustrated in Figure 3.10[17].

Then, for each list of segments in the word, the syllable parser allocates a syllable structure and places all phones (if any) preceding the first vowel into the `phones` field of the onset. Then any vowels are placed into the `phones` field of the nucleus. Finally, any phones following the vowel are placed into the `phones` field of the coda. Then

[17]I assume that the speech stream has already been segmented into words, and that the signal denoting each word has been reduced to a set of timing slots, each roughly represented by the actual glyphs used in the English orthography for the word. The work presented here does not turn on the issue of how this segmentation is performed.

```
((v a n 0)(c o u 0)(v e r 2))

((e 0)(g a 0)(l i 0)(t e 2))
```

Figure 3.10: **Input to the Syllable Parser**

the final element of the list of segments, the stress value, is stored in the `stress` field of the nucleus.

During creation of the syllable, the syllable parser checks the values of $P_4$ and $P_5$. If $P_4$ is 1, then feet will be quantity-sensitive; in this case, if $P_5$ is 0, the syllable's `heavy?` flag is set to `TRUE` if the `phones` field of the coda is non-empty (which indicates a branching rime), and if $P_5$ is 1, then the syllable's `heavy?` flag is set to `TRUE` if the `phones` field of the nucleus has a length greater than 1 (branching nucleus). The `heavy?` flag is set to `FALSE` otherwise.

When the syllable parser concludes its operations, the input has been transformed into a set of syllable data structures, their contents appropriately filled and flagged.

### 3.2.2 Metrical Constituent Construction

There are two algorithms used for metrical constituent construction: one for bounded (binary) feet ($P_1 = 0$), and one for unbounded feet ($P_1 = 1$). In the case of binary feet, these are constructed by iterating over the set of syllables, either from left to right ($P_2 = 0$) or from right to left ($P_2 = 1$). In either case, the value of Extrametricality ($P_7$) is checked, and if $P_7 = 1$, then either the leftmost syllable is ignored ($P_8 = 0$) or the rightmost syllable is ignored ($P_8 = 1$). Extrametrical syllables do not participate in foot construction, and are attached to the word tree as bare syllables. The remaining syllables are grouped into feet containing two syllables. If there is an odd number of syllables, then a single unary foot is built at the right word boundary (if construction is left-to-right) or at the left word boundary (if construction is right-to-left)[18].

The construction of unbounded feet is a more straightforward task. All the syllables in the word (with the exception of an extrametrical syllable, if present) are placed into a single foot that is as long as it needs to be to contain the metrical syllables in the word.

### 3.2.3 Secondary and Primary Stress Assignment

Once the syllables in the word have been parsed into feet, stress is assigned. First, secondary stress is assigned to the head of each foot, according to the value of $P_3$. In the case of bounded (binary) feet, either the leftmost syllable in the foot ($P_3 = 0$) or the rightmost syllable in the foot ($P_3 = 1$) receives a secondary stress of 1[19]. In the case of unbounded feet, the same process is performed on the single foot constructed for the word, with secondary stress assigned to either the leftmost syllable ($P_3 = 0$) or the rightmost syllable ($P_3 = 1$).

The feet and any extrametrical syllables are then attached to a *word tree*. Since the computer program that assigns stress must store syllables and feet in a list anyhow, there is no need for a separate data structure. However, the word tree is important for stress assignment, since it determines the location of primary (word) stress. Primary stress is assigned to either the leftmost foot in the word ($P_0 = 0$) or the rightmost foot in the word ($P_0 = 1$). In the case of unbounded feet, which contain only a single foot, this determination is trivial and always indicates the single foot in the word. Primary stress is assigned by locating the strongest syllable in the appropriate foot (the head), and incrementing the `stress` field of its nucleus. Since all heads have already received secondary stress, this second increment results in a stress value of 2.

[18]Note that in words with an even number of syllables, such a unary foot can also occur in penultimate or in second position, if the final or initial syllable is extrametrical.

[19]The assignment of stress is performed by incrementing the integer stored in the `stress` field of the syllable's nucleus. Since this integer is initialized to 0, the first increment results in a stress value of 1.

### 3.2.4 Stress Clash Avoidance

After building the word tree, the stress processor must check the value of $P_9$. If $P_9 = 1$, then the language in question does not tolerate stress clash, and the word tree must be checked for possible stress clashes. This is achieved by scanning across the word tree and checking for any adjacent syllables that are stressed. For example, consider the process of assigning stress to the Warao word *yiwaranae*, illustrated in Figure C.4. Following construction of the word tree, we note that two adjacent syllables (the first two) receive secondary stress, and violate Warao's stress clash avoidance rule (in Warao, $P_9 = 1$). In such a stress clash, the weak (non-branching) of the two feet is *defooted*. This is accomplished by resetting the value of the `stress` field of the syllable nucleus to 0 (removing all stress that was previously assigned), and setting the syllable's `defooted?` flag to `TRUE`, indicating that this syllable was destressed (this step is performed purely for the sake of tracing the steps in processing, and does not influence the learning algorithm). In the final output shown in Figure C.4, this is indicated by the minus sign marking the first syllable, which has lost its secondary stress.

The value of $P_9$ is checked and destressing performed only if the value of $P_{10}$ (Iterativity) is 0 (feet are iterative). When feet are non-iterative, only primary stress is marked, so stress clash cannot occur.

### 3.2.5 Parameter Definitions

The parameters used by the stress processor are defined as follows. Each parameter has a set of two legal values. These are mapped onto either 0 or 1 depending on their ordinal position in the definition (see Figure 3.11). For example, parameter $P_5$ has legal values `:rime` and `:nucleus`, which are mapped onto 0 and 1, respectively. The parameters are defined as keywords and then mapped to integers so that the program code can reference the keywords, which are more meaningful to the programmer, while the learner can manipulate just the 0 and 1 encodings of the parameters, which are easily stored and changed by manipulating 0's and 1's in a vector.

```
(defparm $p0 :left :right)

(defparm $p1 :binary :unbounded)

(defparm $p2 :left :right)

(defparm $p3 :left :right)

(defparm $p4 :no :yes)

(defparm $p5 :rime :nucleus)

(defparm $p6 :no :yes)

(defparm $p7 :no :yes)

(defparm $p8 :left :right)

(defparm $p9 :no :yes)

(defparm $p10 :no :yes)
```

Figure 3.11: **Parameter Definitions**

In (Dresher and Kaye, 1990), these parameters are given labels 1, 2, 3, 4, 5, 6, 7, 8a, 8, 9 and 10. To make the learner easier to program, I mapped these original labels onto a strictly-integer, zero-origin set of labels, 0 through 10. The latter set of labels is used throughout this thesis.

### 3.2.6 Parameter Dependencies

As discussed by Dresher and Kaye (1990, p. 9), the set of 11 parameters described here does not indicate $2^{11} = 2048$ possible stress systems. This is due to the fact that not all combinations of parameter values are meaningful; in fact, some parameters have possible values which are only meaningful if other parameters are already set to some particular value. For example, consider parameters $P_4$ and $P_5$. Since $P_5$ determines the type of quantity sensitivity (`:rime` or `:nucleus`) when $P_4 = 1$, it would be pointless for the learner to change the value of $P_5$ when $P_4 = 0$. In this case, we arbitrarily assume that any parameter that depends on another parameter (as $P_5$ depends on $P_4$) shall always have a value of 0 when the parameter it depends on is set to 0. This drastically reduces the set of possible parameter settings by removing meaningless variation.

```
; "P5 is suspended (must be 0) unless P4 is QS (0)."
  (dependency $p5 :implies ($p4 0))

; "P8 is suspended (must be 0) unless P7 is Yes (1)."
  (dependency $p8 :implies $p7)

; "If P4 is QI (1), the P1 must be Binary (0)."
  (dependency $p4 :implies ($p1 0))

; "P6 requires that P4 is QS (0)."
  (dependency $p6 :implies ($p4 0))

; "P1 = Binary (0) is not available when P6 is Yes (1)."
  (dependency $p6 :implies $p1)

; "P2 is suspended (must be 0) when P1 is Unbounded (1)."
  (dependency $p2 :implies ($p1 0))

; "P3 is suspended (must be 0) when P6 is Yes (1)."
  (dependency $p3 :implies ($p6 0))
```

Figure 3.12: **Parameter Dependencies**

The set of parameter dependencies recognized by the stress processor is shown in Figure 3.12. Since these dependencies are fully described by Dresher and Kaye (1990, p. 9), I will not discuss them in any further detail. Once defined, the dependencies are referenced by the learner when new hypotheses are created, so that hypotheses which contain meaningless parameter values are not activated.

There is one point of departure from the dependencies listed by Dresher and Kaye. As shown above in the section on Stress Clash Avoidance, the stress processor performs defooting when necessary to avoid stress clash. The Dresher and Kaye model does not handle destressing in stress assignment directly; rather, the learner considers resetting the stress-clash avoidance parameter if it detects what seems to be a weak foot adjacent to a stressed syllable (Dresher and Kaye, 1990, p. 144-145). In my model, both values of $P_9$ are recognized by the learner, resulting in a solution space of 432 different stress systems, instead of the 216 stress systems learned by the Dresher and Kaye model[20].

### 3.2.7 Integration with the Learning Algorithm

The stress processor is integrated with the learning algorithm in a learning system for metrical parameters, as shown in Figure 3.14. The stress processor passes two pieces of information to the learning algorithm: the observed stress pattern associated with the input word, and the stress pattern constructed by the metrical processor for the same syllables (like YOUPIE, the metrical processor strips off the stress markings from the input word and stores them for comparison with the pattern assigned by the current parameter settings). In addition, the learning algorithm also has access to the current set of parameter values, which it can modify during learning.

It is important to note that the amount of processing performed on the two stress patterns (input and output) is minimal: they are simply checked to see if they match. If not, the learner will subtract weight from the current hypothesis. If the patterns do match, then the current hypothesis successfully accounts for the current piece of data, and receives additional weight.

[20]However, there are further dependencies that derive from explicit treatment of stress clash; see Section 3.6.5 for further discussion.

```
PROCEDURE Parse (Input, Hypothesis);
 DECLARE InputSegments, InputStress, OutputStress;
 InputSegments <- ExtractSegments (Input);
 InputStress <- ExtractStress (Input);
 OutputStress <- ExtractStress (AssignStress (InputSegments));
 IF InputStress = OutputStress
   THEN RETURN 1;
   ELSE RETRUN 0;
 ENDIF;
END;
```

Figure 3.13: **Algorithm for PARSE**

## 3.3 Metrical Test Data

To test the learner on a comprehensive range of data, I have constructed a data generation program that can produce input data of any length for any of the legal stress systems delimited by the 11 parameters and their dependencies. This program is called to produce a new input word each time the learner reads a new piece of data during a learning trial. In this section I discuss the process of data generation as it relates to the space of possible parameter settings, the set of possible syllable types, and the input words considered by the learner.

### 3.3.1 Generating the Set of Legal Hypotheses

The first step is to define the set of legal hypotheses determined by the parameters, their values and the dependencies between their values (cf. Figures 3.11 and 3.12). This is accomplished by generating all possible hypotheses of length 11, permuting a vector of length 11 and copying any resulting configurations that satisfy all the dependencies shown in Figure 3.12. Once the parameter settings for all the legal hypotheses (or stress systems) have been generated, we can proceed to generate adequate test data

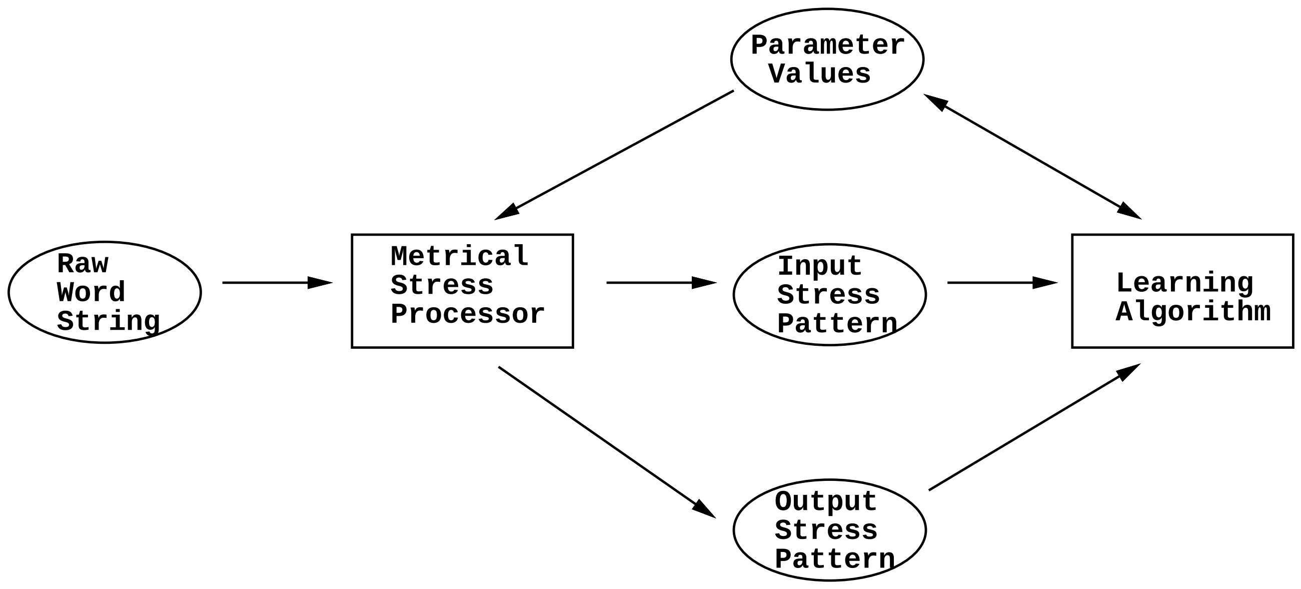

Figure 3.14: **Parameter-Setting Architecture for Metrical Learner**

and test the learner on each possible stress system. The set of legal hypotheses can be seen in Appendix D.

### 3.3.2 Possible Syllable Types

The next step is to list all the possible syllable types that may occur. For the purposes of this model, it is necessary to consider both heavy and light syllables, since quantity-sensitive systems distinguish between them. Heavy syllable types include those with branching rime (a closed syllable that has a coda), those with branching nucleus (a syllable with a long vowel), and those with both branching rime and branching nucleus[21]. Light syllable types include just the open syllable with a short vowel (neither branching rime nor branching nucleus).

[21]Since the Dresher and Kaye model makes no distinctions based on weight of the onset or coda, syllables that are considered heavy by virtue of consonant clusters in the onset or coda are not considered here.

These distinctions derive the following set of syllable types:

V

VV

CV

CVV

CVC

CVVC

The parameters proposed by Dresher and Kaye make no distinctions based on the presence or absence of onsets[22]. In replicating their model, it is therefore unnecessary to represent both V and CV (for example), since both would be treated equally under any and all parameter settings. We can therefore further reduce the set of syllable types by considering only the variant containing an onset, yielding the following set of syllable types:

CV

CVV

CVC

CVVC

### 3.3.3 Generating Input Words

To generate data for each language in the set of possible stress systems, we must produce all possible permutations of the set of syllable types for each possible word length $n$. Since there are 4 possible syllable types, there will be $4^n$ distinct word types of length $n$. For example, suppose $n = 3$; then there will be $4^3 = 64$ distinct word types of length 3.

[22]Dan Everett (personal communication) notes that some languages do in fact distinguish between CV and V syllables, and between VCC and VC syllables (Everett, 1988).

In the language learning literature there has been considerable attention paid to the "boundedness of degree of error" of the input data (Wexler and Culicover, 1980; Lightfoot, 1989; Hammond, 1990b). It is intuitively obvious that shorter words (and shorter sentences) are more frequently spoken than longer ones. Presumably, then, children are exposed to a much higher frequency of shorter words than longer words. The assumption is that the longer words or sentences become, the less likely they are to be encountered by the language learner. It is therefore of interest to determine, for a given learning algorithm, whether it can be shown to learn the languages in its solution space given a finite bound on the length of each datum. Boundedness of degree was first associated with error by Wexler and Culicover (1980); since it is assumed that the learner changes its hypothesis when it makes an error on some input datum, it is important to ascertain whether all such "triggering" errors occur in data of a sufficiently short length, which can be guaranteed to occur in a finite input sample of realistic size.

Recent work by Hammond suggests that the boundedness of degree of error for metrical phonology might be no more than a length of 7 syllables (Hammond, 1990b); this seems to coincide with the general consensus among field linguists that the vast majority of words in most languages are 7 syllables or less in length (Everett, personal communication). For these reasons, I have chosen to test the learner on words of length 1, 2, 3, 4, 5, 6 and 7 syllables.

**The Learning Paradigm**

In summary, the learning paradigm for the phonological learner is characterized as follows:

- *Learning Algorithm*: The learning algorithm specified in Figures 2.5 - 2.10;
- *Set of Things to be Learned*: The 432 hypotheses from the Dresher and Kaye parameter set, as implemented in the stress processer;

- *Set of Learning Environments*: A set of texts, each consisting of a possibly infinite number of example stress patterns drawn from the target language, which obey a random distribution across word length and syllable type (see below);

- *Hypotheses that Occur to the Learner*: Sets of active hypothesis nodes, as defined by the hypothesis graph (cf. Chapter 2) and the operation of the learning algorithm;

- *Termination Condition*: Since the learner is presented only with examples $s \in L_{target}$, and the nature of the learning algorithm is such that it will never abandon the target hypothesis when presented only with examples $s \in L_{target}$, the learner is self-monitoring, and stops when it has activated the target hypothesis $h$, $W(h) > (1 - \epsilon)$, and $h$ is the only active hypothesis.

The algorithm for generating one example input word for a given hypothesis is shown in Figure 3.15. The learner first picks some length $1 < n < 7$ at random[23], and then generates a list of syllables of length $n$ by randomly picking from the list of possible syllable types $n$ times. The stress processor is called on the resulting string of syllables, assigning the correct stress pattern for the target language. The result is an example word from the target language to be learned, which is then input to the learning algorithm.

## 3.4 Preliminary Testing

Before we turn to a presentation of learning results from the entire set of 432 languages in the learner's hypothesis space, it is instructive to consider the results from preliminary testing, in which the learner was presented with data from five languages: French, Maranungku, Warao, Latvian, and Lakota. This set of languages contains a mix of

[23]Given the lack of concrete information concerning relative frequency of word length across all languages, the present implementation abstracts away from the differing frequencies of words of different lengths.

```
PROCEDURE Generate-Input-Datum (Hypothesis);
 DECLARE WordLengths, WordLength, Syllable, Result;
 GLOBAL SyllableList;
 WordLengths <- (1 2 3 4 5 6 7);
 WordLength <- RandomlyPickOne (WordLengths);
 FOR i from 1 to WordLength DO:
    Syllable <- RandomlyPickOne (SyllableList);
    Push (Syllable, Result);
 END;
 Result <- AssignStress (Result, Hypothesis);
 RETURN Result;
```

Figure 3.15: **Algorithm for GENERATE-INPUT-DATUM**

different stress patterns (see Figure 3.16). To test the learner, 100 trials were conducted with each language. The number of input words processed before the learner selected the correct hypothesis was counted. The results of testing are shown in Figure 3.17.

The easiest language to learn was Maranungku; because all the parameters are set to their default values in Maranungku, the initial hypothesis held by the learner (00000000000) describes the Maranungku stress pattern. The learner therefore does not need to explore any other hypotheses, and quickly selects the initial hypothesis after 31 input examples.

French is somewhat more difficult for the learner to process, since three parameters must be set to reach the correct hypothesis. The learner required 105 input examples to learn the fixed final stress pattern of French.

Latvian is similar to French, having fixed initial stress. Therefore it is not surprising that the learner required on average about the same number of examples (128) to learn Latvian.

Like Maranungku, Warao has a binary stress pattern; however, its otherwise smooth stress pattern is disrupted by the stress-clash avoidance rule, which changes the binary stress pattern in some cases. Since presumably stress clash does not occur in every

| Language | P0 | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| French | :left<br>0 | :unbd<br>1 | :left<br>0 | :right<br>1 | :yes<br>1 | :rime<br>0 | :no<br>0 | :no<br>0 | :left<br>0 | :no<br>0 | :no<br>0 |
| Maranungku | :left<br>0 | :bin<br>0 | :left<br>0 | :left<br>0 | :no<br>0 | :rime<br>0 | :no<br>0 | :no<br>0 | :left<br>0 | :no<br>0 | :no<br>0 |
| Warao | :right<br>1 | :bin<br>0 | :right<br>1 | :left<br>0 | :no<br>0 | :rime<br>0 | :no<br>0 | :no<br>0 | :left<br>0 | :yes<br>1 | :no<br>0 |
| Latvian | :left<br>0 | :unbd<br>1 | :left<br>0 | :left<br>0 | :yes<br>1 | :rime<br>0 | :no<br>0 | :no<br>0 | :left<br>0 | :no<br>0 | :no<br>0 |
| Lakota | :left<br>0 | :unbd<br>1 | :left<br>0 | :left<br>0 | :yes<br>1 | :rime<br>0 | :no<br>0 | :yes<br>1 | :left<br>0 | :no<br>0 | :no<br>0 |

Figure 3.16: **Parameter Values for Preliminary Test Languages**

word, it takes more examples before the learner gathers enough evidence that the stress-clash avoidance parameter ($P_9$) has the value 1. As a result, the learner required on average 211 examples to learn Warao. The prediction made by the model is that the presence of a stress-clash avoidance rule makes the stress pattern of a language more difficult to learn.

Of these 5 language, the language that required the most examples to learn was Lakota. Lakota is like Latvian, having unbounded, left-headed feet, but with one crucial difference — in Lakota, the Extrametricality parameter ($P_7$) has the value 1. The left-most syllable in Lakota words is extrametrical, resulting in fixed second-syllable stress rather than fixed initial stress. The learner required 284 examples on average to learn this stress pattern. The model therefore seems to predict that extrametricality is hard to learn. Discussion of empirical results continues in Section 3.6.

| Language | Hypothesis | Average No. Examples |
|---|---|---|
| French | 01011000000 | 105 |
| Maranungku | 00000000000 | 31 |
| Warao | 10100000010 | 211 |
| Latvian | 01001000000 | 128 |
| Lakota | 01001001000 | 284 |

Figure 3.17: **Preliminary Test Results.** Language, parameter values, and average number of examples required for the learner to select the correct language.

## 3.5 Comprehensive Testing

The stress learner has been tested on all 432 languages in its hypothesis space; the results of these learning trials are shown in Appendix D. Each entry in the appendix lists the following:

- *Hypothesis*. The hypothesis representing the parameter values of the target language to be learned (the hypotheses in the appendix are listed in order of increasing markedness (number of 1 bits));
- *Target Activation*. The number of the cycle in which the learner first activated the target hypothesis;
- *Selection*. The number of the cycle in which the learner assigned a weight greater than $(1 - \epsilon)$ to the target hypothesis, and the target hypothesis was the only hypothesis remaining in the active list;
- *Expansion*. The number of times that the learner performed hypothesis activation because there were no nodes remaining in the active list.

## 3.6 Discussion

In the remainder of this section, I discuss the predictions made by this model for which these data provide evidence.

### 3.6.1 Markedness and Learning Time

The learner predicts that languages which are more marked (contain more parameters which are not set to the default (0) value) take longer to learn. Since the learner begins with the least-marked hypothesis (all 0's) and limits each successive round of hypothesis activation according to the Single Value constraint, it looks next at hypotheses with 1 bit set, then 2 bits set, etc. As a result, the learner in general takes longer to learn languages that are more marked.

The space of 432 possible stress systems tested can be divided into equivalence classes according to markedness, starting with hypotheses with 0 bits set, then 1 bits set, etc. The average number of total cycles and expansions for each markedness class are shown in Figure 3.18. These results are summarized in Figure 3.19.

As expected, the learner takes longer on average to learn hypotheses with more bits set. This mirrors the intuition implicit in most parameter-setting research, namely, that setting a parameter means waiting for the appropriate triggering data to appear in the learning environment. This has the effect of placing importance on the definitions of the parameters themselves; the model makes predictions about the ease or difficulty of learning particular languages based on their position in the hypothesis graph (markedness). As a result, the choice of default vs. marked values for parameters will directly influence learnability.

This characteristic, which follows from the Single Value Constraint and its use in hypothesis activation, results in a model that is more easily evaluated with respect to empirical data. In both YOUPIE and DARWIN, the learning time is not affected by the choice of default and marked values for parameters; as a result, these models

| Class | No. Hypotheses | Avg. Total | Avg. Expansion |
|---|---|---|---|
| 0 | 1 | 31 | 0 |
| 1 | 7 | 40 | 1 |
| 2 | 24 | 93 | 4 |
| 3 | 54 | 146 | 7 |
| 4 | 87 | 180 | 10 |
| 5 | 102 | 198 | 11 |
| 6 | 86 | 215 | 12 |
| 7 | 50 | 228 | 13 |
| 8 | 18 | 251 | 14 |
| 9 | 3 | 256 | 14 |
| 10 | 0 | — | — |
| 11 | 0 | — | — |

Figure 3.18: **Average Total Cycles and Expansion, Markedness Classes**

make no prediction about parameter definitions, and cannot be used to prefer one parameterization over another on these grounds.

This raises an issue with respect to markedness that was previously mentioned in Section 1.1.5. The notion of "markedness" cannot play two roles in a learning theory without creating a conflict; namely, the use of markedness to make empirical predictions in a parameterization (as discussed above) conflicts with the use of markedness to order hypotheses to ensure learning (for example, the use of default/marked values as defined by the Strict Subset Principle (cf. Sections 1.1.4, 2.4)).

Systems which must enforce notions of markedness that are derived from constraints on learning algorithms themselves rather than empirical predictions about child language can fail to make correct the correct decision about default/marked values. For example, in his discussion of null subject languages, Berwick concluded that the values of the null subject parameter must be ordered so that the non-null subject language is ordered before the null subject language, since (with all other parameters

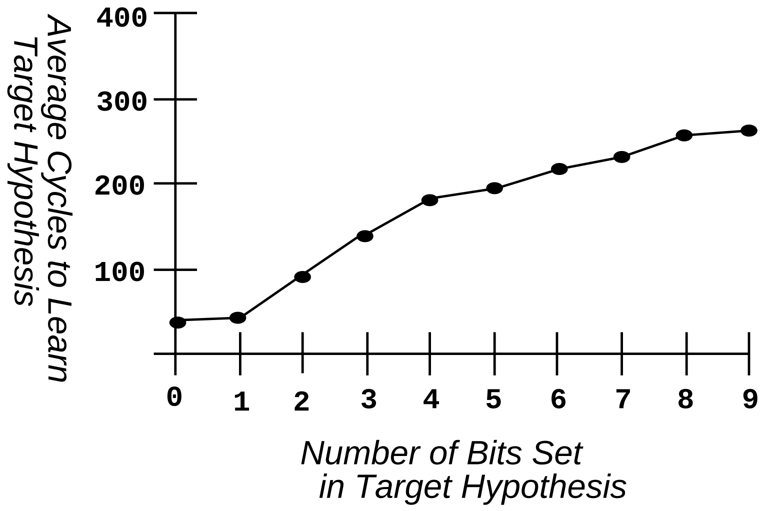

Figure 3.19: **Average Cycles vs. Number of Bits Set in Target Hypothesis**

held constant) the null subject language is a superset of the non-null subject language (Berwick, 1985). What is clear from the empirical data concerning syntactic acquisition, however, is that children order the null subject language *before* the non-null subject language (Hyams, 1986). In the case of the null subject parameter, the empirical notion of "marked" differs from the notion of "marked" which is based on a contraint on the learning algorithm.

I would claim that a learning model that does not make distinctions between default and marked values based on machine constraints is to be preferred, precisely because problems like these can be avoided. In the extreme case, the use of learnability constraints to impose a hypothesis ordering can result in learning behavior that is opposite from the behavior noted in children (as in the case of the null subject parameter). In YOUPIE, determinism, the Subset Principle, and the need to formulate coherent, unambiguous cues implies a strict ordering on cue application (and parameter setting) which is based solely on the needs of the learning algorithm rather than an empirically meaningful parameter definition. This results in a prediction about learning stages which is an artifact of the learning algorithm. There would be no way to change the order of parameter setting in YOUPIE without constructing a completely new set of cues.

The model presented in this thesis does not suffer from these problems. The behavior of the learner directly reflects the nature of the parameter values as defined by the theorist, thus making it possible to change the default/marked values of the parameters as desired. This results in a model that is more flexible and which does not place constraints on markedness based on the particular learning method chosen. The learner is successful because it uses the Relaxed Subset Principle, which allows the learner to recover from overgeneralization when it occurs, rather than strictly ordering hypotheses as required by the Strict Subset Principle. As noted above, the Strict Subset Principle can force a system to make predictions which are "unnatural." A system which can make and then recover from overgeneralization can not only avoid the problems caused by a strict ordering on hypotheses; it can also exhibit behavior observed in children (overgeneralization + recovery (Ullman, et al., 1989)) which a more strictly ordered system cannot.

### 3.6.2 Identically Stressed Languages

Testing the model on data from all of the 432 languages also reveals certain characteristics of the parameters themselves that are of interest from both a linguistic perspective and a learnability perspective.

Given the parameterization proposed by Dresher and Kaye, the learning algorithm was not always able to select the target language it was given to learn, because it selected some other language instead. In this section and the sections that follow, I will describe the two reasons for this: identically stressed languages and languages with a high degree of overlap.

The first point of interest is that the parameters described in (Dresher and Kaye, 1990) do not derive a set of 432 unique languages. That is to say, there are some languages that generate identical surface stress patterns on every word in the sample set, despite the fact that they have completely different parameter settings.

```
 A: #*00010000010

Metrical constituents (RIGHT-headed, BINARY feet built from the LEFT):
Final output (weak feet are defooted in clash):

     [  0        2 +    ][  0        1      ][  0 -    ]
       / \      / \        / \      / \        / \
      K   *    K   *      K   *    K   *      K   *
         / \      / \        / \      / \        / \
        O        OO         OO  K    O   K      OO  K

B: #*00000001000

Metrical constituents (LEFT-headed, BINARY feet built from the LEFT)
  (the LEFT-most segment is extrametrical):
Final output:

        0       [  2 +      0      ][  1        0      ]
       / \        / \      / \        / \      / \
      K   *      K   *    K   *      K   *    K   *
         / \        / \      / \        / \      / \
        O          OO       OO  K      O   K    OO  K
```

Figure 3.20: **Languages with Identical Stress Patterns**

For example, consider the languages shown in Figure 3.20. Both A and B have binary feet built from the left; however, A has right-headed feet and stress clash avoidance, and B has left-headed feet and a left-extrametrical syllable. Interestingly, both of these combinations yield the same stress pattern, not only for the example word, but for all words[24].

Since the languages are identical, the learner will select either one when presented

[24]This is true of all words of any length, so extending the learning environment to words of length greater than 7 syllables would not derive different results.

data from one of the languages. Its choice depends on which hypothesis is activated first, since the data will support both hypotheses. This phenomena raises an issue of plausibility for parameterizations of grammar, namely, if the parameterization admits two distinct parameter settings that nevertheless derive identical languages, then it is difficult to see how the learner could select just one of them, unless there is some strict ordering on languages.

### 3.6.3 Languages with a High Degree of Overlap

The second point of interest is that some pairs of languages share almost all of the same stress patterns for the sample word set. The parameter settings which encode these languages derive almost identical stress patterns, with very few words that are assigned a different stress pattern in the two languages.

| Word Length | Identical Stress | Different Stress |
|---|---|---|
| 1 | 4 | 0 |
| 2 | 16 | 0 |
| 3 | 48 | 16 |
| 4 | 224 | 32 |
| 5 | 960 | 64 |
| 6 | 3968 | 128 |
| 7 | 16128 | 256 |

Figure 3.21: **Overlap Between Languages 01011101010 and 01001101010**

For example, Figure 3.21 shows the overlap between two unbounded, QS languages which differ only by the value of the head assignment parameter. The reason for the overlap can be explained as follows. In unbounded, quantity sensitive languages, the heads of the unbounded feet are marked according to which syllables are heavy. The head assignment parameter (which assigns secondary stress to either the leftmost or rightmost syllable in an unbounded foot) is only meaningful in a quantity sensitive language when there are no heavy feet in the word. Since I assume a uniform random distribution of syllable types within the word, with heavy syllables being equally likely as light syllables, there will be very few words with only light syllables in an average text. Figure 3.21 shows the actual overlap between the two languages; the **Identical Stress** column shows how many of the total number of words for each word length are assigned the same stress pattern; the **Different Stress** column shows how many of the total for each word length are assigned a different stress pattern. There are $4^1 + 4^2 + ... + 4^7 = 21844$ different words in the test sample; in this case, only 496 words out of 21844 are assigned a different stress pattern by the two languages (this represents about 2% of the sample, so the languages overlap by 98%).

Given the infrequent nature of the discriminating examples in this case, the learner selects either of the two languages when it is presented with data from either language, making the prediction that these languages are indistinguishable. This suggests that adult speakers of these languages might not be able to make the correct distinction on the infrequent words that would require the default assignment of a head (cf. Section 3.6.6). It is also true that the presence of infrequent triggering data interacts with the model's ability to learn with noisy data (cf. Chapter 5)[25].

### 3.6.4 Parameter Dependence and Stress Clash Avoidance

The third point of interest raised by the empirical testing concerns the stress clash avoidance parameter, $P_9$. Recall that $P_9 = Yes$ causes non-branching feet that are adjacent to a stressed syllable to be destressed. In the space of 432 languages derived by the Dresher and Kaye parameters, there are languages that are identical even when

[25]Jaime Carbonell (personal communication) points out that learner that performs "local optimization" by activating all hypotheses 1-adjacent to a successful hypothesis (i.e., not just the subset hypotheses) might be able to find the correct language after selecting a highly-overlapping one; however, as seen with the example shown in Figure 3.21, highly-overlapping languages are not necessarily derived by hypotheses that differ only by a single bit value.

they vary in the value of $P_9$, because non-branching feet never occur in a position where they might be destressed.

```
#*10010001010

Metrical constituents (RIGHT-headed, BINARY feet built from the LEFT)
  (the LEFT-most segment is extrametrical):
Final output (weak feet are defooted in clash):

        0       [  0        1      ][  2 +    ]
      / \        / \      / \        / \
     K   *      K   *    K   *      K   *
       / \        / \      / \        / \
      O          O        O          O
```

Figure 3.22: **Stress Clash Example**

Consider the stress pattern illustrated in Figure 3.22. The language has binary feet, built from the left, with a left-extrametrical syllable. This implies that every word with an even number of syllables will have a stress pattern similar to the one shown, with a non-branching foot at the right word boundary[26]. Because the word tree is strong on the right, this foot will always receive primary stress; so although there is a stress clash with the neighboring foot, there is no weak (non-branching) foot with a secondary stress that can be defooted. As a result, the destressing rule has no effect.

This effect is due in part to the fact that the destressing rule is ordered after primary stress assignment. If the order of the destressing rule were parameterized, so that it could take place either before or after primary stress assignment, then setting it to the "before" value in the case noted above would allow destressing to take place, since the rightmost non-branching foot would have a stress value of 1, and could therefore be destressed by the destressing rule. It is left to future research to explore

[26]Words with an odd number of syllables will end in a binary foot, and there will be no stress clash.

this possibility[27].

There are other language pairs where a difference in the stress clash avoidance parameter does not render an identical language, but one that has a high degree of overlap. For example, the difference between `10111101000` and `10111101010` is slight, since there are very few examples where destressing changes the stress pattern of the word. Language `10111101010` has a right-headed, binary stress pattern built from the right, with a left-extrametrical syllable; in addition, the language is quantity-sensitive to branching nuclei. As a result, the only words where destressing will occur are those like the example shown in Figure 3.23, where the penultimate syllable is heavy. Since the rightmost syllable will be marked as the head of a unary foot in this case, and the word tree is strong on the right, the rightmost syllable will always receive primary stress and be adjacent to a syllable with secondary stress. Since the language is right-headed (and feet can only have one head), the penultimate, heavy syllable must also be in a foot by itself. Hence it can fill the role of the non-branching foot to be destressed by the destressing rule.

For languages `10111101000` and `10111101010`, there are only 672 words in the sample set of words of length 7 that are assigned different stress patterns by the destressing rule. This represents only 3% of the sample, so the languages overlap by 97%. This makes the value of the stress clash avoidance parameter difficult to learn for these two languages, for the same reasons mentioned for the head assignment parameter above.

[27]In general, the destressing rule for stress clash avoidance is not as simple as the rule presented here, which represents only one way that languages can choose to destress in avoiding stress clash. A more thorough treatment would consider more realistic variation in the destressing rule. It is worthwhile to point out, however, that the YOUPIE system abstracted away from destressing altogether, and that the insight gained from the presence of even the simple destressing rule included in this system indicates the wealth of interactions between destressing and other phonological processes that must be accounted for in future metrical stress learning models.

```
* (set-parameters-from-vector #*10111101010)
* (str '((k o 0)(k o o 0)(k o 0)))

Input segments: #([KO]<0> [KOO]<0> [KO]<0>)

Metrical constituents (RIGHT-headed, BINARY feet built from the RIGHT)
  (the LEFT-most segment is extrametrical)
  (branching NUCLEUS must be strong):

      0       [  1      ][  1      ]
     / \        / \        / \
    K   *      K   *      K   *
       / \        / \        / \
      O          OO         O

Word tree (strong on the RIGHT)

      0       [  1      ][  2 +    ]
     / \        / \        / \
    K   *      K   *      K   *
       / \        / \        / \
      O          OO         O

Final output (weak feet are defooted in clash):

        0       [  0 -    ][  2 +    ]
       / \        / \        / \
      K   *      K   *      K   *
         / \        / \        / \
        O          OO         O

* (set-parameters-from-vector #*10111101000)
* (str '((k o 0)(k o o 0)(k o 0)))

Input segments: #([KO]<0> [KOO]<0> [KO]<0>)

Metrical constituents (RIGHT-headed, BINARY feet built from the RIGHT)
  (the LEFT-most segment is extrametrical)
  (branching NUCLEUS must be strong):

      0       [  1      ][  1      ]
     / \        / \        / \
    K   *      K   *      K   *
       / \        / \        / \
      O          OO         O

Word tree (strong on the RIGHT)

      0       [  1      ][  2 +    ]
     / \        / \        / \
    K   *      K   *      K   *
       / \        / \        / \
      O          OO         O
```

Figure 3.23: **Infrequent Stress Clash**

### 3.6.5 Parameter Dependence and Non-Iterative Feet

The presence of parameter $P_{10}$, which introduces non-iterativity, presents a difficulty for a parameter-setting learning. As described in Section 3.1.7, a language with non-iterative feet assigns stress as though secondary stresses were assigned through the usual process of foot construction, in order to locate the head which receives primary stress; however, the secondary stresses do not appear in the surface stress pattern. When $P_{10}$ is set to 1, all secondary stress is erased, and the only information accessible to the learner is the location of the primary stressed head. Since there are many language variations in the 11 parameter system that assign the same word stress and vary only in their assignment of secondary stress, the presence of non-iterativity has the effect of erasing the distinctions between languages.

For example, the stress-clash avoidance rule (cf. Section 3.1.6) affects only the secondary stresses in the word; when active, it destresses a non-branching foot with secondary stress. Since secondary stress is erased anyway when non-iterativity is activated, a variation in the value of $P_9$ will have no effect whatsoever on the resulting language. For every language that is non-iterative ($P_{10} = 1$), there will be two identical "variations" with $P_9 = 0$ and $P_9 = 1$. This conflation is illustrated in Figure 3.24.

In another example, extrametricality has no effect on the surface stress pattern of the word when it appears at the opposite edge of the word from primary stress. In this case, the presence of an extrametrical syllable affects only secondary stress, and since secondary stress is erased in non-iterative languages, varying the values of $P_7$ and $P_8$ will result in indistinguishable "variations." (cf. Figure 3.25).

When $P_6 = 1$, then non-heavy syllables can head a foot. When binary feet are constructed so that the head is located at the same edge of the word as primary stress, and feet are non-iterative, then varying the value of $P_4$ (quantity-sensitivity) will have no effect. For example, Figure 3.26 illustrates the situation where the presence of a heavy syllable (branching rime) in the second position would otherwise produce a distinction between two languages, if it were not for the effect of non-iterativity.

```
HYPOTHESIS: #*00010000011

Input segments: #([KO]<0> [KO]<0> [KO]<0>)

Metrical constituents (RIGHT-headed, BINARY feet built from the LEFT):

   [  0        1      ][  1      ]
     / \      / \        / \
    K   *    K   *      K   *
       / \      / \        / \
      O        O          O

Word tree (strong on the LEFT)

   [  0        2 +    ][  1      ]
     / \      / \        / \
    K   *    K   *      K   *
       / \      / \        / \
      O        O          O

Final output (non-iterative, stress only strongest head):

     [  0        2 +    ][  0      ]
       / \      / \        / \
      K   *    K   *      K   *
         / \      / \        / \
        O        O          O

HYPOTHESIS: #*00010000010

Input segments: #([KO]<0> [KO]<0> [KO]<0>)

Metrical constituents (RIGHT-headed, BINARY feet built from the LEFT):

   [  0        1      ][  1      ]
     / \      / \        / \
    K   *    K   *      K   *
       / \      / \        / \
      O        O          O

Word tree (strong on the LEFT)

   [  0        2 +    ][  1      ]
     / \      / \        / \
    K   *    K   *      K   *
       / \      / \        / \
      O        O          O

Final output (weak feet are defooted in clash):

     [  0        2 +    ][  0 -    ]
       / \      / \        / \
      K   *    K   *      K   *
         / \      / \        / \
        O        O          O
```

Figure 3.24: **Non-Iterativity and Stress Clash**

```
HYPOTHESIS: #*00000001101

Input segments: #([KO]<0> [KO]<0> [KO]<0>)

Metrical constituents (LEFT-headed, BINARY feet built from the LEFT)
  (the RIGHT-most segment is extrametrical):

   [  1        0      ]   0
     / \      / \        / \
    K   *    K   *      K   *
       / \      / \        / \
      O        O          O

Word tree (strong on the LEFT)

   [  2 +      0      ]   0
     / \      / \        / \
    K   *    K   *      K   *
       / \      / \        / \
      O        O          O

Final output (non-iterative, stress only strongest head):

     [  2 +      0      ]   0
       / \      / \        / \
      K   *    K   *      K   *
         / \      / \        / \
        O        O          O

HYPOTHESIS: #*00000000001

Input segments: #([KO]<0> [KO]<0> [KO]<0>)

Metrical constituents (LEFT-headed, BINARY feet built from the LEFT):

   [  1        0      ][  1      ]
     / \      / \        / \
    K   *    K   *      K   *
       / \      / \        / \
      O        O          O

Word tree (strong on the LEFT)

   [  2 +      0      ][  1      ]
     / \      / \        / \
    K   *    K   *      K   *
       / \      / \        / \
      O        O          O

Final output (non-iterative, stress only strongest head):

     [  2 +      0      ][  0      ]
       / \      / \        / \
      K   *    K   *      K   *
         / \      / \        / \
        O        O          O
```

Figure 3.25: **Non-Iterativity and Stress Clash (Continued)**

```
HYPOTHESIS: #*00001000001

Input segments: #([KO]<0> [KOK]<0> [KO]<0>)

Metrical constituents (LEFT-headed, BINARY feet built from the LEFT)
  (branching RIME must be strong):

   [  1      ][  1        0      ]
     / \        / \      / \
    K   *      K   *    K   *
       / \        / \      / \
      O          O   K    O

Word tree (strong on the LEFT)

   [  2 +    ][  1        0      ]
     / \        / \      / \
    K   *      K   *    K   *
       / \        / \      / \
      O          O   K    O

Final output (non-iterative, stress only strongest head):

     [  2 +    ][  0        0      ]
       / \        / \      / \
      K   *      K   *    K   *
         / \        / \      / \
        O          O   K    O

HYPOTHESIS: #*00000000001

Input segments: #([KO]<0> [KOK]<0> [KO]<0>)

Metrical constituents (LEFT-headed, BINARY feet built from the LEFT):

   [  1        0      ][  1      ]
     / \      / \        / \
    K   *    K   *      K   *
       / \      / \        / \
      O        O   K      O

Word tree (strong on the LEFT)

   [  2 +      0      ][  1      ]
     / \      / \        / \
    K   *    K   *      K   *
       / \      / \        / \
      O        O   K      O

Final output (non-iterative, stress only strongest head):

     [  2 +      0      ][  0      ]
       / \      / \        / \
      K   *    K   *      K   *
         / \      / \        / \
        O        O   K      O
```

Figure 3.26: **Non-Iterativity and Stress Clash (Continued)**

Other examples of languages that are identical under the effects of non-iterative destressing include the following:

- Binary foot construction always assigns either first, second, penultimate or final word stress, depending on the values of $P_0$, $P_7$ and $P_8$. Varying the value of $P_2$ (Directionality) does not alter the placement of primary stress, so it does not result in variation in the language in non-iterative systems;
- Binary foot construction and unbounded foot construction will result in identical languages under non-iterativity, in all cases where the two languages assign the same primary stress.

There are other examples, too numerous to mention, of particular language pairs that are quite distinct when $P_{10} = 0$, and indistinguishable when $P_{10} = 1$. I will forego the inclusion of further examples here.

### 3.6.6 Empirical Predictions and Evidence

As discussed in Chapter 2, honoring the Single-Value Constraint gives the learning algorithm more cognitive plausibility, since it proceeds through an orderly progression of hypotheses that would tend to reflect the gradual learning observed in children. It would be appropriate to evaluate the behavior of the phonological learning application by comparing the progression of hypotheses activated by the learner to any empirical evidence available concerning the time course of the child's phonological development.

Unfortunately, such data are not available at present. There have been no longitudinal studies investigating the time course of phonological development. Existing work includes the investigation of early infant responses to different stress patterns (e.g., the work of Mehler and his colleagues (1988)), which indicates that infants show differences in attentional focus when presented with samples from two languages simultaneously; and some investigation of the relationship between phonological stress and the development of grammatical morphemes (e.g., the work of Gerken (1987; 1990)).

The type of data that would be pivotal in evaluating proposals concerning phonological parameters and stress learning would include some indication of the chronological ordering of parameter setting in child learning (i.e., which parameters are set first), for all the natural languages (or all the languages in the parameter space, as the case may be). Such data could be obtained through transcription of recorded data acquired longitudinally for several children for each language. Such a data gathering effort would be monumental, and is certainly beyond the scope of the present work; however, future work must address this issue before parameterized theories of metrical phonology can be completely evaluated with respect to empirical data.

One promising source of empirical data concerning phonological parameter setting is the second-language acquisition of English by adults (Archibald 1989, 1990, 1991). Through the analysis of production task test results, Archibald has shown that native speakers of Hungarian and Polish move through an "interlanguage" state, where they have correctly learned some but not all of the phonological parameters of English and make predictable errors in stress assignment. All of the interlanguage errors analyzed by Archibald are analyzable in terms of metrical parameters; for example, the Hungarian subjects consistently made errors that indicated they had transferred the L1 settings of the Directionality and Quantity Sensitivity parameters (incorrectly) from Hungarian to English. We would expect the metrical stress learner to make similar kinds of errors; in fact, the "interlanguage" observed in L2 learners has a direct correlate in the intermediate hypotheses that would be activated by the learner as it moves from L1 to L2 on the basis of new data from L2. I will not investigate this further here, but future work should address this issue. It would be particularly relevant to simulate this kind of learning environment and analyze the learning model's behavior.

## 3.7 Summary

In this chapter, I have presented an implementation of a metrical processor that assigns stress to input words according to the values of the parameters defined in (Dresher and Kaye, 1990). This processor was integrated with the learning algorithm presented in Chapter 2 and tested on the entire space of 432 languages determined by the parameters and their dependencies. The results of testing indicate that the learner can successfully learn the parameters of metrical phonology, although there are some characteristics of the Dresher and Kaye parameter space (e.g., identically-stressed languages, overlapping languages) that pose a problem for any parameter learner. I will return to a discussion of these problems in Chapter 6.

# Chapter 4

# Learning Syntactic Parameters

In this section, I discuss the application of the limited non-deterministic learner to the task of learning syntactic parameters. I adopt the framework presented in (Clark, 1990b) for the definition of parameters to be learned, replicating the learning environment used to test Clark's DARWIN learner.

This task is undertaken as a means of demonstrating that the learning algorithm presented here is not tied in any way to the model of phonological processing presented in the previous chapter. In this chapter I show how the same framework can be applied successfully to a different parameter learning problem.

Unlike the theory of stress assignment, which is amenable to more or less complete formalization at a certain level of abstraction, the theory of syntax is much broader and more complex. There has not been any adequate formalization of all the possible parameters even within a single domain of syntax, let alone the entire field of inquiry. For this reason it is not possible to construct a syntactic model analogous to the one presented in Section 3 for metrical stress assignment rules. It might be possible, for example, to build a parameterized model of, say, the Binding Theory, which has been studied in some detail (Wexler and Manzini, 1987; Lust, 1988), but to do so while abstracting away from the effects of other syntactic modules on learning would produce results of limited usefulness.

I instead adopt the approach followed by Clark (1990b). In order to test his DARWIN learning model, Clark created an abstract characterization of syntactic parameters and syntactic parsing which serves to provide a framework within which abstract syntactic parameters may be learned. In the remainder of this section, I describe the replication of Clark's learning environment for the purposes of testing the limited non-deterministic learning algorithm, and the results of testing the learner on some example learning problems. In contrast to the work presented in Section 3, the application of the learner to this type of learning problem does not involve a realistic implementation of the parsing module itself. The goal of this investigation is to show that the learning algorithm presented in this thesis can be used to learn parameters in any linguistic environment; it is not limited to learning phonological parameters. Although I present several results from testing the learner on large search spaces, the learner has not been tested on syntactic parameters in as much depth as it has on metrical parameters; further investigation is left to future research.

## 4.1 Implementation

As described earlier in Section 1.4.3, Clark's DARWIN model uses an abstraction of syntactic parsing for the purposes of testing the learner. As I was fortunate enough to have access to Clark's CommonLisp source code, I was able to adapt the parse simulator for use with my learning algorithm.

In the remainder of this section, I describe only a few particulars of Clark's model; the thorough reader is referred to (Clark, 1990) for complete details.

### 4.1.1 Expressibility

In his description of DARWIN, Clark introduces the notion of *expressibility*. He notes that any given input example may be parsed correctly even if some of the learner's parameters are not yet set correctly. A given input example *expresses* a parameter if that parameter must be set to its correct value for the input example to be parsed correctly. Note that an input example which expresses few parameters will be parsed correctly by several hypotheses; and an input example which expresses many parameters will only be parsed correctly by fewer hypotheses, those with the correct values of all of those parameters.

To abstract away from the problem of building an entire parsing system, Clark allows the input examples to directly represent the values of the parameters they express, with the parameter values that are not expressed filled in with "don't care" values. For example, assume that the target hypothesis is 0101. An input example that can be parsed correctly if only the first two parameters are set correctly is represented as $(01 - -)$; dashes are used to indicate "don't care" values.

To simulate parsing, each active hypothesis is checked against the parameter values expressed by a particular input datum; if the hypothesis matches the expressed values, then it receives positive weight, otherwise it receives negative weight[1].

### 4.1.2 Masking

The relative expressiveness of a datum is described by the fraction of the total number of parameters it expresses. For example, if there are 10 parameters and some datum $s$ expresses 5 of them, then the expressiveness of $s$ is .5.

Expressiveness is controlled in Clark's model by a variable called the *mask*, which is an integer $m, 0 < m < n$, where $n$ is the total number of parameters. Each time a piece of data is generated by the simulation, the expressiveness of the new datum is randomly selected as an integer between 1 and $m$. This implies that at most $n - m$ parameters must be *masked*, or hidden from the learner in a given input example, depending on the random choice. For example, if $n = 10, m = 5$, then a random

[1]Clark uses a more complicated fitness metric to calculate fitness of a given hypothesis. For the purposes of testing the limited non-deterministic model, it is sufficient to test each input example against the hypothesis itself, assigning a successful parse if they match and an unsuccessful parse if they do not.

selection of some $r, 1 \leq r \leq 5$ means that $10 - r$ bits must be masked out in the datum presented to the learner. Assume that $r = 3$, and that the data generator produces the datum (0101101010). Then the data generator must mask $10 - 3 = 7$ bits at random. One possible outcome of masking this datum is (01—0—-). The implication of this piece of data is that the learner need only have set the correct values for the first, second and sixth parameter to correctly parse the input.

### 4.1.3 Testing Hypotheses Against the Data

The same learning algorithm was used to learn these parameters using Clark's data generation scheme. To test each active hypothesis, the learner checked each non-masked bit position in the datum to determine whether the corresponding bit position in the hypothesis has the same value. If an expressed bit position in the datum has a different value from the corresponding position in the hypothesis, then this is interpreted by the learner as a failed parse, and the example is treated as an $s \notin L(h)$ and $h$ receives less weight; otherwise $h$ receives more weight, as determined by the sigmoid weighting function.

### 4.1.4 Learning Paradigm

In summary, the learning paradigm for the syntactic learner is characterized as follows:

- *Learning Algorithm*: The learning algorithm specified in Figures 2.5 - 2.10;
- *Set of Things to be Learned*: Hypotheses of length 4, 10, 20 and 30 parameters, with expressiveness values of .2, .4, .6, .8 and 1.0 (see following section for details);
- *Set of Learning Environments*: A set of texts, each consisting of a possibly infinite number of data examples. For a given expressiveness $e$ (i.e., up to $e$ out of $n$ parameters may be expressed by a single datum (see above)), for each datum $r$ was randomly chosen, $0 \leq r \leq e$. Then $m = (n - r)$ bits were masked with

"don't care" values in the target hypothesis, which was presented as the abstract datum. As described above, this replicates Clark's data presentation scheme (for more discussion, see (Clark, 1990b));

- *Hypotheses that Occur to the Learner*: Sets of active hypothesis nodes, as defined by the hypothesis graph (cf. Chapter 2) and the operation of the learning algorithm;
- *Termination Condition*: Since the learner is presented only with examples $s \in L_{target}$, and the nature of the learning algorithm is such that it will never abandon the target hypothesis when presented only with examples $s \in L_{target}$, the learner is self-monitoring, and stops when it has activated the target hypothesis $h$, $W(h) > (1 - \epsilon)$, and $h$ is the only active hypothesis.

## 4.2 Test Data and Results

In this section, I present the result of some learning experiments using Clark's parsing abstraction on different learning problems. The following variables were tested:

- *Number of Parameters*. Experiments were conducted on learning problems with 4, 10, 20 and 30 parameters;
- *Number of Parameters Set in Target Hypothesis*. Experiments were conducted with all the parameters set in the target hypothesis (e.g., 1111, 1111111111, etc.) and with half the parameters randomly set in the target hypothesis (e.g., 0011, 0110001001, etc.). As discussed in the previous section, learning time is proportional to the number of bits set in the target hypothesis. Accordingly, an experiment with half the parameters set in the target hypothesis produces results for learning hypotheses in the "middle" of the hypothesis graph, while an experiment with all the parameters set in the target hypothesis produces results for learning the bottom node in the hypothesis graph, which could be considered the "worst case" scenario;

- *Expressiveness of the Input Data*. Each experiment was attempted with expressiveness of .2, .4, .6, .8, and 1.0.

| Set | N | Expr. | Con. | Cyc. | Exp. | Set | N | Expr. | Con. | Cyc. | Exp. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 4 | 1 | 10 | 141 | 2 | 10 | 20 | 4 | 10 | 438 | 9 |
| 2 | 4 | 2 | 10 | 77 | 2 | 10 | 20 | 8 | 10 | 326 | 10 |
| 2 | 4 | 3 | 10 | 63 | 2 | 10 | 20 | 12 | 10 | 180 | 11 |
| 2 | 4 | 4 | 10 | 52 | 2 | 10 | 20 | 16 | 10 | 172 | 13 |
| 4 | 4 | 1 | 10 | 179 | 4 | 10 | 20 | 20 | 10 | 186 | 18 |
| 4 | 4 | 2 | 10 | 98 | 4 | 20 | 20 | 4 | 10 | 501 | 19 |
| 4 | 4 | 3 | 10 | 76 | 4 | 20 | 20 | 8 | 10 | 383 | 22 |
| 4 | 4 | 4 | 10 | 70 | 4 | 20 | 20 | 12 | 10 | 270 | 23 |
| 5 | 10 | 2 | 10 | 353 | 4 | 20 | 20 | 16 | 10 | 253 | 26 |
| 5 | 10 | 4 | 10 | 191 | 5 | 20 | 20 | 20 | 10 | 254 | 28 |
| 5 | 10 | 6 | 10 | 117 | 5 | 15 | 30 | 6 | 10 | 500 | 14 |
| 5 | 10 | 8 | 10 | 98 | 5 | 15 | 30 | 12 | 10 | 367 | 16 |
| 5 | 10 | 10 | 10 | 87 | 6 | 15 | 30 | 18 | 10 | 265 | 21 |
| 10 | 10 | 2 | 10 | 489 | 9 | 15 | 30 | 24 | 10 | 278 | 27 |
| 10 | 10 | 4 | 10 | 217 | 10 | 15 | 30 | 30 | 10 | 325 | 36 |
| 10 | 10 | 6 | 10 | 153 | 10 | 30 | 30 | 6 | 10 | 588 | 30 |
| 10 | 10 | 8 | 10 | 133 | 10 | 30 | 30 | 12 | 10 | 493 | 36 |
| 10 | 10 | 10 | 10 | 125 | 11 | 30 | 30 | 18 | 10 | 406 | 41 |
|  |  |  |  |  |  | 30 | 30 | 24 | 10 | 412 | 47 |
|  |  |  |  |  |  | 30 | 30 | 30 | 10 | 430 | 53 |

Figure 4.1: **Learning 4, 10, 20 and 30 Parameters**

Figures 4.1, 4.2, 4.3, 4.4 and 4.5 illustrate the test results from these experiments.

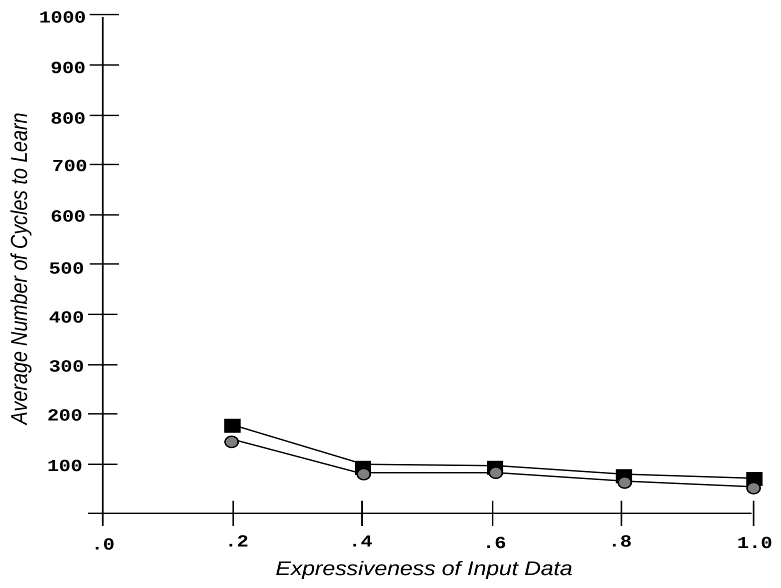

Figure 4.2: **Average Learning Cycles,** $n = 4$

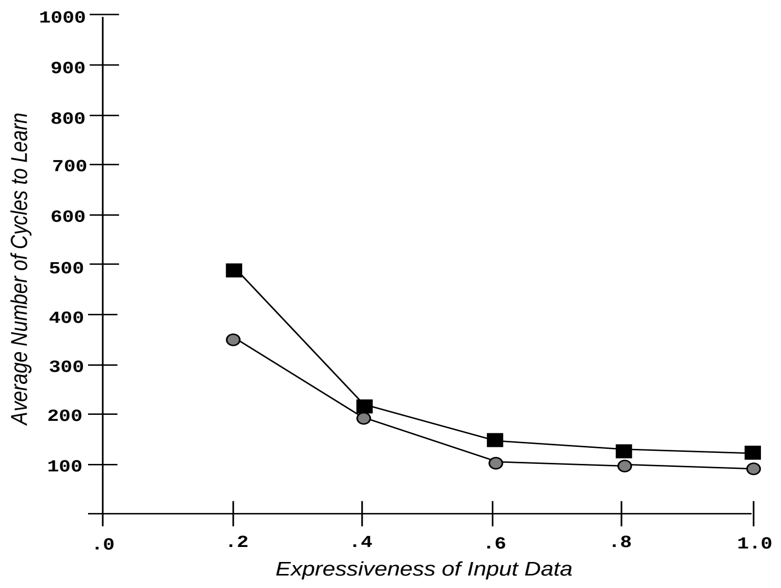

Figure 4.3: **Average Learning Cycles,** $n = 10$

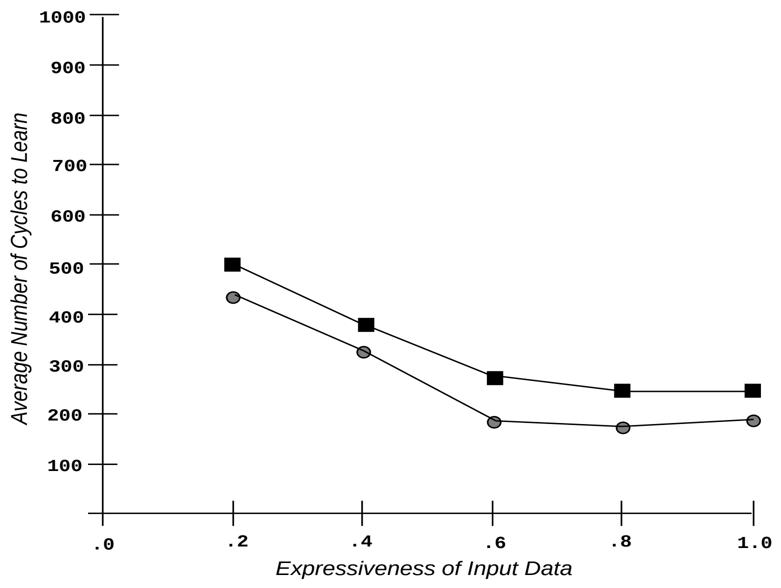

Figure 4.4: **Average Learning Cycles,** $n = 20$

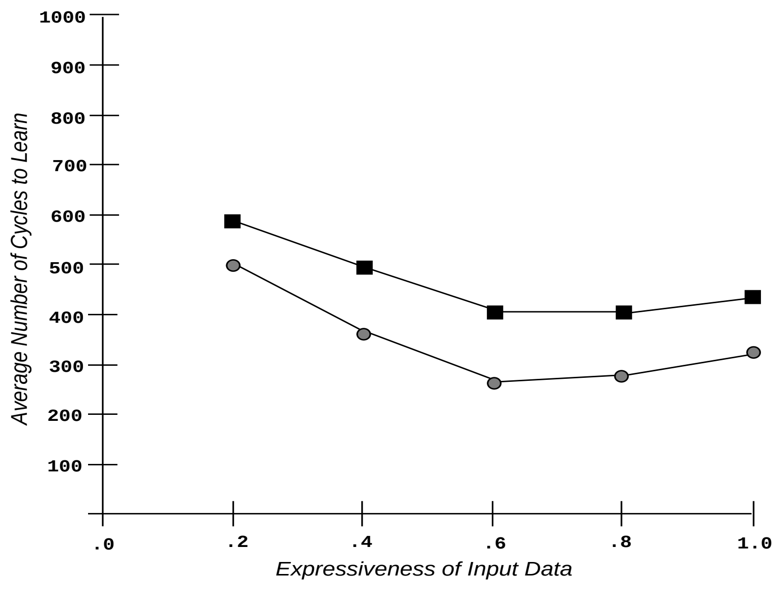

Figure 4.5: **Average Learning Cycles,** $n = 30$

## 4.3 Discussion

The results presented in this section indicate that the learning algorithm is computationally tractable in abstract parameter spaces of sizes at least up to 30 parameters[2]. Experimental results indicate that the learner requires on average an amount of input examples that increases linearly as the size of the parameter space increases.

In addition, the application of the learning algorithm to abstract parameter spaces like Clark's syntactic parsing model shows that the model is not tied in any way to the parameters of metrical phonology. This is a concrete demonstration of the model's extensibility.

One interesting result is that the average learning time is sensitive to the expressiveness of the input. In all the parameter spaces tested, the learning algorithm required more time to learn when the data were less expressive. This is intuitive, given that less expressive data presents less evidence to the learner with each input example, since fewer bits are set and the data are less discriminating with respect to the set of possible hypotheses. A less intuitive result is the slight increase seen in learning time as expressiveness reaches 1.0 in larger learning spaces (cf. Figures 4.4 and 4.5). I will not investigate this phenomenon further here, but it is necessary to point out that it is unlikely that any realistically complex natural language would have such high expressiveness on average (there are very few sentences in any language which express every single parameter value in the language).

## 4.4 Summary

In this chapter, I have demonstrated the application of the learning algorithm to the problem of learning abstract syntactic parameters of the type proposed in (Clark, 1990a). The learner was tested on spaces of up to 30 parameters and was able to

[2]This was given as an example of a "hard" problem in (Clark, 1990a).

correctly select the target language in each case. The number of input examples required to select the target language is linearly related to the number of parameters in the space.

# Chapter 5

# Learning With Noisy Data

## 5.1 The Noise Model

One of the characteristics of the learning algorithm presented here is that it can learn even in the presence of some number of ungrammatical examples $s \notin L(h_{target})$. In this chapter, I discuss the noise model used to test the algorithm and discuss limits on the ratio of noise to positive examples. I present some empirical results from testing the algorithm with this noise model, using the abstract parameter space discussed in Chapter 4. I will then discuss some outstanding problems when learning in noisy environments, and suggest some future extensions to the present model that could be made to improve its performance.

### 5.1.1 Definition

The noise model used here is what I call a "random injector" model. Before each datum is presented to the learner, a weighted coin toss is used to choose whether a) a grammatical example $s \in L(h_{target})$ is to be presented, or b) an ungrammatical (noisy) example $s \notin L(h_{target})$ is to be presented. In the former case, an input example is constructed normally as described in Chapter 4; in the latter case, an input example constructed normally, but then one of the expressed bits in the datum is “flipped” (if 0, then changed to 1; if 1, then changed to 0) to create a datum that will not match the target hypothesis and will therefore be considered outside the target language in the abstract syntactic parameter model (see Chapter 4).

To simulate a certain percentage of noise in a data presentation, the weighted coin toss was set up according to the desired proportion of noise to signal. For example, if the proportion 1/6 (noise/signal) was selected, the random number generator would be made to choose an integer between 1 and 6; if 1 were selected, then a noisy example would be injected into the presentation, otherwise a grammatical example would be presented.

Intuitively, this model will almost always produce data presentations where the noisy examples are “spread out” over the length of the presentation. Texts which include large “bursts” of noisy examples occur only with very low probability. Since the noise model is intended to test the resilience of the model to sporadic ungrammatical examples in the learning environment, this is to be desired. Nevertheless, there are texts that occur with low probability that pose problems for the learning algorithm; I defer discussion of these texts to Section 5.3.1.

### 5.1.2 Limits on the Ratio of Noise to Positive Examples

Assuming that the learner has activated the correct target language $L(h)$, the presence of noisy examples in the input might cause the learner to prune $L(h)$ and activate another hypothesis. Given the ratio of noise to signal that was used to generate the data presentation, we can predict how well the learning algorithm will perform by relating the ratio of noisy examples to the number of examples $s \notin L(h)$ required for the algorithm to prune $L(h)$. This number of examples is related to the slope of the sigmoid function, which depends on both the damping factor $k$ and the two sigmoid weighting coefficients (cf. Chapter 2). Recall that the weight of a hypothesis $h$ depends on the net positive evidence seen for $h$ so far in the input text: $NPE_t(h) = |POS_t(h)| - |NEG_t(h)|$.

The introduction of the weighting coefficients allows the learner to adjust the relative impact of positive and negative examples; in the experiments presented throughout the thesis, the positive coefficient $K_+ = 1$, and the negative coefficient $K_- = 5$. Thus the weight of a hypothesis at time $t$ is given by the equation

$$y = SIG(x) = 2/(1 + e^{-k\times(K_+\times|POS_t(h)|-K_-\times|NEG_t(h)|)}) - 1.$$

Simply speaking, we can multiply the number of positive examples in a text by $K_+$, and subtract from that number the number of negative examples in the text multiplied by $K_-$ to calculate the net positive evidence.

Assume that $K_+ = 1$ and $K_- = 5$, as stated above, and that the damping factor $k = .1$, and $\epsilon = .1$. The sigmoid will cross the lower threshold $(\epsilon - 1)$ when $x = -30$ ($SIG(-30) = .905149$). Thus a hypothesis will be pruned when $NPE_t(h) \leq -30$. If $POS_t(h) = n$, then a hypothesis will be pruned if $NEG_t(h) = n/5 + 6$. For example, if there are 5 positive examples and 7 negative examples in a text of length 12, the hypothesis will be pruned[1].

The intuitive notion of a "killer text" is that the learning algorithm will fail to assign a weight greater than $(1-\epsilon)$ to the target hypothesis on any finite text of length $t$ where $K_+ \times |POS_t(h)| - K_- \times |NEG_t(h)| = x$, $SIG(x) \leq (\epsilon - 1)$. Given the default settings of the constants used here, this occurs when $POS_t(h) - 5 \times NEG_t(h) = -30$. If there are $n$ positive examples, then there must be $.2n + 6$ negative examples. This implies a ratio of negative examples to total examples which approaches 16.6% as the size of the text gets larger.Therefore, when given a finite text of length $t$, the learner will assign a weight greater than $(1 - \epsilon)$ to the target hypothesis and terminate learning with the correct answer if the ratio of negative examples falls below this percentage.

When we consider infinite texts, however, the problem is not quite so straightforward. If we associate the random coin-toss method of noise injection with some

[1]It should be clear that the behavior of the sigmoid weighting function can vary widely according to how the values of $k$, $K_+$ and $K_-$ are set.

probability distribution, then the Weak Law of Large Numbers tells us that the actual observed distribution of noise to signal in a given data presentation will approach the coin-toss probability in the limit. Intuitively, this means that the we can guarantee that the actual ratio of noise to signal is the same as the ratio used to generate the noise *only* in an infinite text. More precisely, the ratio of noise to signal used to generate the data presentation corresponds to the *limiting relative frequency* of noisy examples.

As stated earlier, texts with large "bursts" of noisy examples occur only with very low probability. Nevertheless, for any $k$, we can imagine an infinite text that contains a burst of $k$ noisy examples which will still obey the limiting relative frequency, since the overall effect of the burst of noise on the ratio will disappear as the text grows infinitely bigger. In fact, we can imagine a text with any finite number of such bursts, followed by an infinite "tail", that will obey the limiting relative frequency.

In Section 5.2, I will present a discussion of empirical results gathered by testing the algorithm, using the noise model here, with the same termination condition used in Chapters 3 and 4. These results confirm the intuitions presented above concerning the acceptable ratio of noise to signal in the model. As discussed in Chapter 1, these results are not equivalent to limiting results, due to horizon effects that occur when learning stops after a finite time in noisy environments. I will return to a discussion of limiting results in noisy environments and show how the model could be modified to learn with infinite noisy texts in Section 5.3.1.

In the remainder of this section, I focus on some formal results concerning noisy data, "killer" texts, and probability of convergence.

### 5.1.3 Probability of Finite Killer Text of Size $n$

A *killer text* is a sequence of examples that will cause the learner to abandon a single hypothesis $h$ with weight greater than $(1 - \epsilon)$. This corresponds to the case where the learner has successfully chosen the target language and then encounters a text which causes it to abandon the correct hypothesis.

A text is constructed by selecting a sentence $s$ at each time $t$, such that

$s_t \notin L(h)$ with probability $P_e$

$s_t \in L(h)$ with probability $(1 - P_e)$.

There are two types of sentences to choose from at each time $t$, so there will be $2^n$ different ways to construct a text of length $n$.[2]

Since the noise model assumes independence from sentence to sentence (no conditional probability for $s$ based on the previous $s$, for example), the probability of some text $T$ can be calculated by the product of the probabilities of each of its sentences. For example, if a particular text contains 5 positive examples and 3 negative examples, its probability is $(P_e)^3 \times (1 - P_e)^5$. Assume that $P_e = .1$. Then the probability of this text would be $(.1)^3 \times (.9)^5 = .000591$.

The probability that any text of size $n$ containing $m$ negative examples will occur can be calculated by multiplying the product of the sentence probabilities, as derived above, by the number of ways that such a text can be constructed by choosing a sentence at each time $t$. The number of ways to place $m$ negative examples in a text of size $n$ is $C(n, m) = \frac{n!}{m!(n-m)!}$. The probability of some text of length $n$ containing $m$ negative examples is therefore $C(n, m) \times (P_e)^m \times (1 - P_e)^{n-m}$. Since the example text has 8 sentences total and 3 negative examples, the number of ways to construct such a text is $C(8, 3) = \frac{8!}{3! \times 5!} = 56$. Therefore, the probability of any text of length 8 containing 3 negative examples is $56 \times .00591 = .033$.

For any $n$, we are interested in the case where $m$ is chosen so that the ratio of negative examples is such that the learner abandons a chosen hypothesis. Since negative examples are weighted to be 5 times as salient as positive examples, a text that contains anything more than 1/5 as many negative examples as positive examples

[2]Note that in the general case there are many more ways to construct texts taking into account the identity of the sentences themselves, but since the learner distinguishes sentences only by their membership in $L(h)$, we can make this simplification without loss of generality.

will cause the learner to abandon the chosen hypothesis. Thus,

$$C(n,(n/5+1)) \times (.1)^{(n/5+1)} \times (.9)^{(n-n/5+1)} \tag{5.1}$$

calculates the probability of a "minimal" killer text of size $n$. Figure 5.1 shows how this probability gets smaller for $n$ up to 1000. In the limit, the probability of a minimal killer text is 0. It can be shown that $lim_{n\to\infty}C(n,m) \times (P_e)^m \times (1-P_e)^{n-m} = 0$ as follows. The maximal value for the product of $(P_e)^m \times (1-P_e)^{n-m}$ is at $P_e = .5$. Thus we have $\lim_{n\to\infty} \frac{n!}{m!(n-m)!} \times 2^{-m} \times 2^{(n-m)}$, which is equivalent to $\lim_{n\to\infty} \frac{n!}{m!(n-m)!2^n}$. The maximal value of $C(n,m)$ for any n is at $m = \lfloor \frac{n}{2} \rfloor + 1$ (when $n$ is odd), or $m = \frac{n}{2}$ ($n$ is even). For $n$ even, the denominator becomes $2^n \times \frac{n}{2}! \times \frac{n}{2}!$. There are $n$ terms in $\frac{n}{2}! \times \frac{n}{2}!$; if we group a 2 with each we get the following sequence of terms in the denominator:

$$n \times n \times (n-2) \times (n-2) \times (n-4) \times (n-4) \ldots \tag{5.2}$$

If we compare this sequence with the terms in the numerator (from $n!$), we can cancel all the even terms, and note that the odd terms in the denominator are all strictly greater than the remaining terms in the numerator:

$$\frac{n \quad (n-1) \quad (n-2) \quad (n-3) \quad (n-4) \quad (n-5) \quad \ldots}{n \quad\quad n \quad\quad (n-2) \quad (n-2) \quad (n-4) \quad (n-4) \quad \ldots}$$

A similar fraction is derived when $n$ is odd, with all the odd terms cancelling:

$$\frac{n \quad\quad (n-1) \quad (n-2) \quad (n-3) \quad (n-4) \quad (n-5) \quad \ldots}{(n+1) \quad (n-1) \quad (n-1) \quad (n-3) \quad (n-3) \quad (n-5) \quad \ldots}$$

In both cases, each term in the denominator is strictly greater than the corresponding term in the numerator, so the value of the fraction goes to 0 as $n \to \infty$.

| **N** | Prob. Minimal Killer Text |
|---:|:---|
| 10 | 0.057 |
| 50 | 0.006 |
| 100 | $4.956 \times 10^{-4}$ |
| 150 | $4.472 \times 10^{-5}$ |
| 200 | $4.243 \times 10^{-6}$ |
| 250 | $4.143 \times 10^{-7}$ |
| 300 | $4.121 \times 10^{-8}$ |
| 350 | $4.154 \times 10^{-9}$ |
| 400 | $4.228 \times 10^{-10}$ |
| 450 | $4.335 \times 10^{-11}$ |
| 500 | $4.471 \times 10^{-12}$ |
| 550 | $4.634 \times 10^{-13}$ |
| 600 | $4.822 \times 10^{-14}$ |
| 650 | $5.035 \times 10^{-15}$ |
| 700 | $5.271 \times 10^{-16}$ |
| 750 | $5.533 \times 10^{-17}$ |
| 800 | $5.820 \times 10^{-18}$ |
| 850 | $6.134 \times 10^{-19}$ |
| 900 | $6.476 \times 10^{-20}$ |
| 950 | $6.847 \times 10^{-21}$ |
| 1000 | $7.249 \times 10^{-22}$ |

Figure 5.1: **Probability of Minimal Killer Text of Size** $n$

### 5.1.4 Other Killer Texts

Note that there are certainly other ways to construct killer texts of size $n$. Beyond minimal killer texts, one can consider any text where $(n/5 + 1) < m \leq n$. Note, however, that increasing the size of $m$ relative to $n$ has two effects: the value of $C(n, m)$ decreases, since there are fewer ways to fit in $m$ negative examples; and the value of $(P_e)^m \times (1 - P_e)^{n-m}$ will also decrease, since $P_e$ is multiplied more times into the product[3]. In these cases, the probability will always be less than the probability of a minimal killer text, so the limit of each of these possibilities will also be 0 as $n \to \infty$.

The probability of any killer text of size $n$ can be calculated by the following sum:

$$\sum_{m=(n/5+1)}^{n} C(n, m) \times (P_e)^m \times (1 - P_e)^{n-m}. \tag{5.3}$$

Since $lim_{n\to\infty} \sum_i f(x_i) = \sum_i lim_{n\to\infty} f(x_i)$, the limit of the sum will be sum of the limits, which are all zero; so the limit of the sum will also be zero.

### 5.1.5 Probability of an Infinite Killer Text

If we consider all possible data presentations in the limit, then there are an infinite number of infinite texts that can be constructed by selecting each $s$ as described in the previous section. Trivially, one can construct an infinite killer text by selecting any finite killer text and appending an arbitrary infinite sequence of sentences to it. The probability that the learner will encounter some text selected from the infinite set of texts is 1; therefore each text itself must have non-zero probability. By the Strong Law, an event with non-0 probability will occur in the limit; so the probability of an infinite killer text is 1.[4]

[3] I assume that $P_e$ is always less than $(1 - P_e)$.

[4] The model, as implemented, can eventually recover from killer texts if allowed to keep learning. As discussed elsewhere in this chapter, the model may be augmented by the addition of a memory mechanism that requires more and more noise to be present for a successful hypothesis to be pruned on successive killer texts.

### 5.1.6 Probability of Convergence in the Model

It is useful to consider whether the learning algorithm can be guaranteed to converge to the correct hypothesis in finite time. In this section I will consider the properties of the model, and show that convergence can be guaranteed only in the limit.

It is difficult to characterize the exact behavior of the model without a complete definition of the particular language space to be learned (sentences in each language, their relative frequency, overlap with other languages, etc.). I will therefore assume what for this model is the worst case: For each $h$, $L(h)$ contains no $s$ which is in any other language $L(h')$. This implies that the learner will not assign any positive weight to partially correct hypotheses[5]. In this case, the learner will activate the default hypothesis; assuming that it is not the target language, it will soon prune the default hypothesis and activate its neighbors. Since none of these will contain sentences in the target language, they will all be pruned soon, and at the same time. When the learner has no more active hypotheses, and more than one hypothesis was pruned on the last input example, the learner randomly selects one of the pruned hypotheses and activates its neighbors. If the target hypothesis is not one of them, then they will all be pruned simultaneously and one will be chosen at random and its neighbors will be activated, etc.

Without loss of generality, assume that the learner performs this after each input example if the target hypothesis is not activated[6]. Then at each input example, a new node would be added to the search path, starting with the default node, then the neighbor that was chosen to have its neighbors activated, etc. The probability of a path of length $n$ will be $(1/m)^n$, where $m$ is the number of parameters.

For any two hypotheses, the path between them will be no longer than $m$; if the

[5]Note that assigning positive weight to partially correct hypotheses assumes that some language which are "close" in terms of bit settings have a significant intersection in terms of sentences; as shown in Chapter 3, this is not necessarily the case, so it is reasonable to assume the worst case.

[6]This could be achieved by setting the coefficient for negative examples in the current model to be -30.

two hypotheses were maximally different (one having a 0 everywhere the other has a 1), then all $m$ parameters would have to be changed to go from one to the other, resulting in a path of length $m$. At any time $t$, therefore, there exists some path from the current hypothesis to the target hypothesis that is no longer than $m$ and therefore has a probability of $(1/m)^m$. At each time $t$, there is therefore a non-zero probability that the learner will follow the right path to the target hypothesis. Nevertheless, we cannot guarantee that this will occur for any predetermined finite time, since for any $m$ there also exists a path from the default node that does not contain the target hypothesis and which also has a non-zero probability. Even so, the Strong Law tells us that the learner will eventually choose the correct path, since any event with a non-zero probability will occur in the limit.

Although there are a finite number of hypotheses to learn, the worst case behavior of the model is worse than exhaustively visiting $2^n$ hypotheses. This is due to the fact that the algorithm must reconsider previously abandoned hypotheses, in case the correct hypothesis was visited but pruned because of noise in the data presentation. If we idealize to no-noise learning, then the algorithm can be modified such that it does not reactivate previously visited hypotheses (e.g., by keeping a list of previously tried nodes). In this case, the worst case behavior of the model becomes the $2^n$ case where all nodes must be visited once (the target language is the last hypothesis node visited in the hypothesis graph).

It is also interesting to consider the best-case behavior of the algorithm. In an ideal domain, there exists a monotonic distance relation $D$, such that:

- $D(h_i, h_j)$ is defined to be the number of non-matching bit values in $h_i$ and $h_j$; and
- For any text $T$, $D(h_i, h_{target}) < D(h_j, h_{target})$ implies that $NPE_T(h_i) > NPE_T(h_j)$.

Given a finite text, the learner will always add more weight to a hypothesis closer to the target language, since its $NPE$ would be greater. In any given set of active hypotheses, the learner will find more evidence for the subset of that set that was closest to the target language. When all active hypotheses are pruned, the learner will activate the neighbors of a previous hypothesis that had minimal distance to the target language . Of all the new hypotheses activated, those that decrease the distance to the target hypothesis (by setting another bit correctly) would survive the longest, and so on. In such an ideal domain, the convergence time would be linear, and related to the number of bits that must be set in the target language.

## 5.2 Empirical Testing

### 5.2.1 Method

The learner's resistance to noisy data was tested using the syntactic parameters discussed in Section 4. The data generation routine was the same as the one used to learn syntactic parameters without noise, with the following exception. When an ungrammatical example was to be presented, one of the expressed parameters in the encoded (grammatical) example was flipped, producing the encoded equivalent of an ungrammatical example. For example, if the grammatical datum selected was (01–01), then one possible noisy example would be (01–11).

The learner was tested with noisy samples generated with random error rates of 1/20 (5%), 1/10 (10%), 1/8 (12.5%) and 1/6 (16.6%). The results are tabulated in Figure 5.2.

### 5.2.2 Results

The table of results shown in Figure 5.2 can be interpreted as follows. The **Set** column indicates the number of bits set (1's) in the target hypothesis. The **N** column indicates the total number of bits in the target hypothesis. The **Expr** column indicates the expressiveness of the data, the number of bits expressed by each input datum (cf. Section 4.1.1). For each error rate tested, the *Cyc* column indicates the average number of cycles (input examples) required for the learner to select the target hypothesis, and the *Exp* column indicates the average number of times the learner had to activate new hypotheses. The experiment was conducted 10 times for each configuration in the table.

With an error rate of 1/6, I was unable to obtain experimental results for the final 8 rows of the table (these missing results are marked **NA** in the table). This is because computational tractability degrades sharply, in larger parameter spaces with high expressibility and a high error rate. Although the theoretical limit on the noise rate is 1/6 (as discussed in the foregoing sections), in practical testing on available computers I was forced to terminate these particular experiments before collecting an adequate amount of data.

Taken as a whole, these results confirm that the learning algorithm can successfully select the target language even when there is a relatively high uniform error rate (up to 1/8 (12.5%) for all parameter spaces tested, and up to 1/6 (16.6%) for parameter spaces of up to size 10).

## 5.3 Problematic Issues

In this section, I discuss three areas where the noise model has an impact on the performance of the learner, before turning to a more comprehensive discussion of noisy data in the child's environment in the following section.

### 5.3.1 Bursts of Noise and Infinite Data Presentations

As mentioned above, it is possible to construct an infinite data presentation that obeys the limiting relative frequency of noisy examples, yet has any number of noise bursts that are large enough to cause the learner to abandon the correct hypothesis. Although these sorts of texts occur with low probability and were not encountered during empirical testing, it is worthwhile to consider the theoretical impact of such texts on the model presented here[7].

The behavior of the learner given such presentations can be described by considering the weight it assigns to the target hypothesis. Assume that the learner accumulates a weight greater than $(1 - \epsilon)$ for the target text, and then encounters a burst of noisy examples that causes the weight of the target hypothesis to dip below $(\epsilon - 1)$. This will cause the target hypothesis to be pruned; assume also that after some time, the target hypothesis is reactivated and the learner begins accumulating weight for it again.

In a data presentation with recurring bursts of noisy examples, a graph of the weight of the target hypothesis might look something like the curve shown in Figure 5.3.

When re-activating a hypothesis, the algorithm sets the hypothesis weight to 0. This has the effect of "forgetting" all the positive evidence seen for a hypothesis. This is undesirable when learning from infinite texts, since in the limit the positive evidence must outweigh the negative evidence, but there may be recurring bursts of noise that are just large enough and frequent enough to cause the learner to "forget" the data seen for the target hypothesis. Since the learner always starts at 0 with a reactivated hypothesis, bursts of constant size and frequency would cause the model to oscillate in this fashion.

The implemented model is not designed to handle bursts of noisy examples, but there is at least one refinement that could be made to the learning algorithm to address this shortcoming. Assume for the moment that the algorithm does not always assign a weight of 0 to hypotheses when they are activated. For each hypothesis that attains a weight of $(1 - \epsilon)$, the learner remembers the highest weight attained by that hypothesis. If that hypothesis is pruned and subsequently reactivated, the learner assigns an initial weight equal to the remembered highest weight, rather than 0. In contrast to the oscillating behavior seen in Figure 5.3, this would produce something more akin to "stair-climbing" behavior; the learner would continue where it left off if it chose to

[7]Thanks to Kevin Kelly for pointing out this problem.

activate the pruned hypothesis again. This would have the effect of "remembering" the net positive evidence for a hypothesis even if it were pruned, thus lessening the effects of noise bursts. The overall effect is that it would be harder and harder for a noise burst to prune a promising hypothesis, since all the positive evidence is remembered. To cause oscillation, it would be necessary for noise bursts to become more frequent and/or larger in size. It will become impossible for the hypothesis to be pruned, since the limiting relative frequency of noise in the presentation prevents the size and frequency of noise bursts to grow in the limit[8].

### 5.3.2 Bursts of Noise and Avoiding Overgeneralization

It has been pointed out that the learning algorithm as presented is also susceptible to small bursts of noise at one crucial time during learning: when the learner has decided to activate a subset language in order to observe the Relaxed Subset Principle[9]. For example, assume that there is some $h$ with a weight $> (1-e)$, and its subset hypothesis $h'$, the target language, is activated. At the time of activation, $h'$ will have a weight of 0. If there is a subsequent burst of consecutive noisy examples, $h'$ will quickly reach a weight below $(\epsilon - 1)$ and be pruned, even if the subset language is the correct hypothesis. Note that the superset language, $h$, has a weight near 1, so it will be much less affected by the noise burst, and it will still be active when $h'$ is pruned. In a case where it was then the only active hypothesis, the learner would incorrectly select $h$.

There are many possible refinements of the learning algorithm that could be implemented to address this shortcoming; I will mention two of them here:

- *Weighting Subset Hypotheses Equally.* At the time that a subset language is acti-

[8]A rigorous proof is beyond the scope of this thesis, and is left to future work. The argument is based on the idea that if a noise burst of $K_{noise}$ examples $s \notin L(h)$ is needed to prune $h$ after $K_{normal} > K_{noise}$ examples were seen in total, then in the limit as $K_{normal} \to \infty$, $K_{noise}$ would need to approach infinity to continue to throw off a model with memory. If we always assume that $P(s \notin L(h)) < P(s \in L(h))$, this becomes a zero probability event in the limit.

[9]Kevin Kelly (personal communication).

vated, it is assigned an initial weight equal to that of the superset language that activated it. This would make the subset hypothesis as resilient to noise as the already-activated superset.

- *Periodic Subset Activation.* Instead of activating a subset hypothesis only once, when the superset hypothesis reaches the $(1-\epsilon)$ threshold, the learning algorithm could periodically or randomly re-introduce the subset hypothesis for an active superset. This would greatly decrease the probability that a random burst of noise would cause the learner to select an incorrect hypothesis, since it would have an infinite number of "tries" at selecting the subset.[10]

Testing either of these proposals would require substantial alterations to the other components of the learning algorithm. In the case of weighting subsets equally, the selection criteria would have to be altered, since assigning a weight greater than $(1-\epsilon)$ to the subset hypothesis would have the immediate effect of pruning the superset language and selecting the subset language, as long as the next example encountered by the learner was in the subset language. It might be possible to assign to the subset some weight greater than 0, but less than $(1-\epsilon)$, avoiding this problematic case. The choice of this initial subset weight and its distance from 0 would make the learner more or less able to avoid selecting the superset language after incorrectly pruning the subset language on a burst of noise.

The periodic reintroduction of the subset language is more robust, since it is guaranteed to survive any single burst of noise (whereas the existing model as well as the variation just discussed will not). In order to simulate this behavior, the termination criteria of the simulation would have to be altered; in particular, it would no longer be appropriate to select a language and terminate learning given a single active hypothesis with weight greater than $(1-\epsilon)$. The learning algorithm would have to process data examples indefinitely, while periodically re-activating any subset languages related to

[10]This idea was suggested by Jaime Carbonell.

the current hypothesis. This implies converging to the correct language in the limit, rather than selecting a language after a finite interval.

I have chosen not to address this particular problem by revising the learning algorithm, since this would require significant effort beyond the scope of this thesis. The results presented here do not turn on the issue, since the claims made here are based on a uniform distribution of noisy examples rather than bursts of noisy examples, which occur only with very low probability in the uniform noise distribution. It would not be enough simply to modify the algorithm and its corresponding coded program without performing all the empirical tests once again, a task that would take several months. Future work should address this problem, as well as the general problems with bursts of noise in the child's learning environment, which are discussed in Section 5.4.

### 5.3.3 Overlapping Languages

As illustrated in Section 3.6, there are cases where different hypotheses generate languages with a significant number of stress patterns in common (in some cases, a 98% overlap). In this kind of situation, a learning algorithm that tries to learn conservatively in order to ignore noise in the input runs the risk of ignoring the distinguishing examples (those assigned different stress patterns by the two hypotheses) because they "look like" noise in their frequency distribution.

It is important to note that the learning algorithm could potentially distinguish between each pair of overlapping languages in a noise-free environment. If the algorithm were modified so that each negative example caused the current hypothesis to be pruned (essentially like traditional error-driven learning), it could successfully prune an incorrect but highly overlapping language once one of the infrequent distinguishing examples were encountered. This could be achieved by setting the sigmoid negative coefficient to a very high value (in the default model discussed here, any value over 200), which would have the effect of weighting a hypothesis below $(\epsilon - 1)$ on a single negative example, regardless of how close the previous weight was to 1.

Many of the problems discussed in Section 3.6 are problems inherent in exhaustive enumeration of the parameterization chosen by Dresher and Kaye, which admits some language variations that are quite possibly not natural. Many of the parameter variations which generate identical or highly overlapping languages could be ruled out by placing further dependencies on parameter values to reflect the insights gained from this investigation. Nevertheless, the problem of distinguishing noise from triggering data will always be an issue in language learning, since many linguistic phenomena are rare and do not occur any more frequently than noisy data in the child's learning environment[11].

I will now turn to a discussion of noisy data in the learning environment, and discuss the implications both for this model and the general problem of language learning.

## 5.4 Noisy Data in the Child's Learning Environment

In this section, I present a discussion of the psycholinguistic implications of different data presentations containing noise, the behavior of the learning algorithm, and the definition of triggering data.

In a model where the child processes a single sentence at a time, each sentence will either be assigned a grammatical structure, indicating a successful parse; or it will not be assigned a grammatical structure, indicating that no successful parse was possible with the current grammar[12]. There are two possible explanations in each case:

A. **Parsed Correctly**.

[11]As mentioned before, Ochs has done work in Samoan concerning the delayed acquisition of less frequent constructions (Ochs, 1985); also, Clark and Roberts have advanced the notion that diachronic language change can be triggered by ambiguous or unexpressed parameter settings (Clark, 1991).

[12]In the discussion which follows, I will abstract away from the effects of cognitive limitations, e.g., sentences which do not parse because they violate memory constraints rather than grammatical constraints.

A1. The current grammar is the target grammar, and the sentence is parsed correctly by the target grammar; or,

A2. The current grammar is not the target grammar, and the sentence is parsed correctly by the current grammar.

B. **Not Parsed Correctly**.

B1. The current grammar is the target grammar, and the sentence is not parsed correctly by the target grammar; or,

B2. The grammar is not the target grammar, and the sentence is parsed correctly by the target grammar.

Examples like B1 are usually referred to as "noisy examples", or ungrammatical sentences that are encountered once the correct grammar has already been selected. Examples like B2 are usually referred to as "errors" in error-driven learning; i.e., examples that cause the learner to abandon the current (incorrect) hypothesis and resume the search for the correct hypothesis.

| | Grammar Correct | Grammar Incorrect |
|---|---|---|
| $s \in L(h)$ | A1 | B2 |
| $s \notin L(h)$ | B1 | A2 |

### 5.4.1 Bursts of Examples $s \notin L(h)$

- **In the Initial Segment of the Input**.
  Presumably, there is some point beyond which the initial segment of the input will have an indelible effect on the child's learning; if the initial segment of text with $s \notin L(target)$ is smaller than this size, then presumably a child will show no traces of the experience in his final adult grammar. For example, if a child were to spend his first two weeks of life in a Russian maternity ward before

returning home to the U.S. with his English-speaking parents, he would most likely acquire normal adult English with no delay at all, despite the fact that he would have heard hundreds or even thousands of sentences $s \notin L(target)$[13]. On the other hand, a child that spends the first two or three years of life with his Russian grandparents will probably acquire English as a second language.

These facts would seem to suggest that children learning their first language are during some period of time quite malleable with respect to the input data, and that later they become "fixed" and cannot undo the learning they have done.

A question arises, namely, how can we propose an algorithm that predicts this effect? We can easily achieve the first effect, since the current algorithm (as well as a Bayesian or other statistically-oriented learner) can undo the effect of previous learning if enough "new" examples are encountered. However, the current algorithm can *always* do this when learning on infinite texts without a termination condition, and does not exhibit the loss of malleability that children seem to exhibit once they reach a certain age[14].

- **At Some Intermediate Stage Before Convergence**.
  Here I will associate the term "convergence" with the loss of malleability in the child. Presumably, if the child during some critical language learning stage is influenced by the methodic introduction of some sentences $s \notin L(h)$, then learning of the correct grammar might be delayed or even changed, depending on the number of examples encountered. For example, if at the stage when the child is acquiring the value of the Bounding Domain parameter a number of sentences with long-distance anaphors are systematically introduced, one might

[13]Thanks to Kevin Kelly for this example.

[14]No existing learning programs address this phenomena. Possible approaches include a theory of maturation, which could be modelled by an independent variable which constrains the set of possible hypotheses, or some theory of fixity based on length of exposure, which could be modelled by a dynamic weighting function. I will not explore these possibilities here; giving a detailed account of either possibility would require significant additional work.

imagine that this would either delay the correct setting of the parameter or even cause the child to converge to an incorrect grammar (assuming that the base language does not allow LDA). The current algorithm can model this[15].

- **After Convergence**.

  For small numbers of examples $s \notin L(h)$, the parameter settings should be unchanged. However, given a large enough text, the current algorithm will change its hypothesis.

  For sufficiently large numbers of examples $s \notin L(h)$, we must consider the possibilities of filtering (conscious) and perhaps second language learning. The first involves "ignoring" some input examples, assuming that they are somehow tagged as non-salient. The second assumes a partitioning of the hypothesis space into a second set of parameter values, so that a different distinct grammar may be learned. The second grammar would be consciously accessed by the hearer/speaker in appropriate attentional contexts.

  Neither of these two phenomena are predicted or supported by the present algorithm, although the present learning system could be extended by the addition of a filtering mechanism and multiple hypothesis spaces in order to support them.

### 5.4.2 Sporadic or "Sparse" Examples $s \notin L(h)$

A language learner should be able to filter the effects of occasional noisy examples in the input, but if there is a high proportion of noisy examples, then learning may be delayed or even unsuccessful. A problem arises, since for certain infrequently occurring examples, the learning algorithm cannot distinguish between these examples and noisy examples[16]. Since the learner treats all examples $s \notin L(h)$ equally (it has no theory of noisy examples), it cannot distinguish between examples with equally low

[15]Some question here; interaction with fixity; subset/non-subset parameters; etc.

[16]For example, several of the stress patterns discussed in Section 3.6.

frequencies of occurrence.

This seems unsatisfactory, since language learners seem to acquire even infrequently appearing constructions. Since learners are also resilient to noise in the input (performance errors, etc.), it would seem that human language learner can distinguish between "salient" errors, which should be attended to, and "noise", which should be filtered. This would require (minimally) some theory of salience, for example, a theory of "how far off" an example is from the current grammar. Examples that are close to the current grammar might trigger refinement, while examples that could only be parsed with radically different parameter values might be filtered.

Assume that the language to be learned has parameter $P$ set to value $v$, but that $P = v$ is expressed by very few examples in the language; in fact, assume that $Pr(s)$, $s$ expresses $P = v$, $< Pr(s')$, $s' \notin L_{target}$ (noisy example). How can the learner "pay attention" to $s$ while filtering $s'$?

Suppose that $s$ is not in $L(h)$ because of the value of a single parameter. That is, the learner can determine that $s$ would be parsed correctly if only a single parameter were reset. In this case, $s$ would fit the notion of a "triggering example" from the literature[17]. Presumably, $s'$ would not be a triggering example, and the learner could therefore weight the presence of $s$ and filter $s'$.

| | Grammar Correct | Grammar Incorrect |
|---|---|---|
| 1-adjacent | salient trigger | salient noise |
| not 1-adjacent | filtered trigger | filtered noise |

This strategy raises the issue of "salient noise" and "filtered triggers." Salient noise includes examples that are parsable by a grammar that is 1-adjacent to the current grammar, but are not in $L_{target}$. Filtered triggers include examples in $L_{target}$ that are not in a grammar that is 1-adjacent to the current grammar. Both of these categories of example will be encountered during learning.

[17]Assuming independence, etc.

If the "salient" strategy is to be used, we must consider when it is appropirate. When the learner hasn't converged on the target or a language 1-adjacent to the target, the filtered triggers shouldn't be filtered. On the other hand, if salient noise is weighted heavily during intermediate learning, the learner will be led down wrong paths and will take longer to converge.

One possibility is to limit the use of the salience measure until the learner is tending to convergence. There are two cases to consider: one where the language held is the target language; the other where it is not. Suppose $L(h) = L_{target}$. Then any examples $s$ that are 1-adjacent to $L$ will be noise; if the salient heuristic is used, the learner might be tricked into giving up the correct hypothesis. Suppose $L \neq L_{target}$; then an $s$ that is 1-adjacent to $L$ might be just the triggering example that is necessary to "push" the learner into the correct hypothesis. Despite this, the 1-adjacent heuristic can fall prey to malicious noise, and cannot by itself suffice as a successful theory of noisy examples.

The notion underlying the description of triggering data in the literature is that of a set of examples that all express some common property (e.g., a long-distance anaphor) that is not captured by the current grammar. The question is, how are these "salient" examples separated from noise? And how does the learner "count" their occurrence, such that it won't be fooled by one malicious noise presentation of a property that isn't in the language?

### 5.4.3 Cue-Based Learning

The ability to detect even a single input example with a particular linguistic property is beyond the scope of the present learning algorithm, which abstracts away from the particular properties of individual sentences and considers only whether they are in or not in the current language. The ability to detect properties of individual sentences is one characteristic of cue-based learning, which is specifically designed to match particular linguistic patterns (Dresher and Kay, 1990). Assume that for highly overlapping languages $L$ and $L'$, there exist $S$, a set containing sentences in $L'$ but not in $L$, and $S'$, a set containing sentences in $L$ but not in $L'$. One can imagine that for each class of examples in $S$, we could formulate some cue that recognizes that class of example and which indicates that examples from that class are salient examples that should be taken as strong evidence that the learner should switch languages.

There are at least two problems with such an approach. The first concerns the presence of noisy examples along with the examples $S$ and $S'$. In a noisy learning environment, a cue-based learner might be tricked by the presence of a noisy example that just happens to fall into a class of examples covered by one of its cues for salient triggering data. In this case it would switch its hypothesis and perhaps select an incorrect language. Dresher and Kaye have proposed the mechanism of "etching" as one means of patching this shortcoming (Dresher and Kaye, 1990, p. 185). The etching mechanism involves a certain activation threshold for each cue, i.e., it must be matched $n$ times by examples in the data before it is actually triggered. This can help to avoid the problem of triggering cues in the face of noise that "looks salient," but is still prey to horizon effects when presented with more than just a few noisy examples in a consecutive burst of noise.

The second problem with cue-based learning as an alternative lies in the complexity inherent in implementing cue-based models. One way of partitioning the set of possible cues is into two classes: *language-independent* cues and *language-dependent* cues. Language-independent cues are pattern/action pairs that are based on a particular parameter value and do not depend on the settings of other parameter values. For example, the statement "If P0 = 1 and you find a syllable with no stress at the left edge of the word, then set P0 to 0" is one example of a language-independent cue; it applies to any language with P0 set to 1. On the other hand, there is nothing in the basic assumptions made by Dresher and Kaye to rule out the formulation of language-dependent cues. For example, statements like "If P0 = 1 and P2 = 0 and P9 = 0 and P4 = 1 and you detect a certain pattern at the right edge of the word, then set P10 to 1" are relevant only to specifically those languages with those parameter settings. Driving this to its logical conclusion, there is nothing that prevents a cue-based system from formulating cues that are for specific languages and mention all of the parameters. This would derive a huge set of language-specific cues for each language in the learning space. It is not clear that such a set of language-specific cues could be shown to be cognitively plausible, although I shall not investigate the matter further here.

## 5.5 Summary

- The learner can successfully learn each target hypothesis with a uniform random distribution of negative examples (errors) that form up to 12.5% of the sample text;
- The learner cannot learn in the presence of bursts of consecutive noisy examples, although I have shown how it could be extended to do so in future work;
- The learner can learn highly overlapping languages in a noise-free environment by setting the sigmoid negative coefficient to a sufficiently high value so that a single negative example causes a hypothesis to be pruned;
- The learner cannot learn highly overlapping languages in a noisy environment, because of the inherent lack of distinction in the noise model between noisy examples and highly salient (but infrequent) triggering examples; this is left to future work.

| Set | N | Expr. | 1/20 Error | | 1/10 Error | | 1/8 Error | | 1/6 Error | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | *Cyc.* | *Exp.* | *Cyc.* | *Exp.* | *Cyc.* | *Exp.* | *Cyc.* | *Exp.* |
| 2 | 4 | 1 | 119 | 2 | 160 | 2 | 204 | 2 | 273 | 2 |
| 2 | 4 | 2 | 89 | 2 | 85 | 2 | 151 | 2 | 142 | 2 |
| 2 | 4 | 3 | 73 | 2 | 82 | 2 | 171 | 2 | 281 | 2 |
| 2 | 4 | 4 | 66 | 2 | 122 | 2 | 158 | 2 | 199 | 2 |
| 4 | 4 | 1 | 157 | 4 | 171 | 4 | 174 | 4 | 452 | 4 |
| 4 | 4 | 2 | 107 | 4 | 142 | 4 | 182 | 4 | 192 | 4 |
| 4 | 4 | 3 | 88 | 4 | 120 | 4 | 164 | 4 | 319 | 4 |
| 4 | 4 | 4 | 79 | 4 | 121 | 4 | 142 | 4 | 243 | 4 |
| 5 | 10 | 2 | 429 | 4 | 289 | 5 | 288 | 5 | 382 | 8 |
| 5 | 10 | 4 | 164 | 5 | 193 | 5 | 190 | 5 | 226 | 6 |
| 5 | 10 | 6 | 122 | 5 | 125 | 5 | 213 | 6 | 442 | 5 |
| 5 | 10 | 8 | 114 | 6 | 121 | 6 | 148 | 5 | 229 | 5 |
| 5 | 10 | 10 | 98 | 5 | 120 | 6 | 219 | 6 | 357 | 6 |
| 10 | 10 | 2 | 451 | 10 | 302 | 10 | 341 | 10 | 317 | 10 |
| 10 | 10 | 4 | 197 | 10 | 216 | 10 | 240 | 10 | 291 | 10 |
| 10 | 10 | 6 | 167 | 10 | 185 | 10 | 223 | 11 | 251 | 10 |
| 10 | 10 | 8 | 163 | 12 | 196 | 11 | 181 | 11 | 442 | 10 |
| 10 | 10 | 10 | 135 | 10 | 194 | 11 | 207 | 11 | 226 | 11 |
| 10 | 20 | 4 | 546 | 9 | 397 | 10 | 409 | 11 | 400 | 10 |
| 10 | 20 | 8 | 264 | 10 | 228 | 11 | 338 | 12 | 415 | 11 |
| 10 | 20 | 12 | 185 | 12 | 239 | 13 | 244 | 14 | NA | NA |
| 10 | 20 | 16 | 170 | 13 | 210 | 17 | 294 | 11 | NA | NA |
| 10 | 20 | 20 | 190 | 16 | 246 | 19 | 267 | 24 | NA | NA |
| 20 | 20 | 4 | 591 | 20 | 497 | 20 | 514 | 21 | NA | NA |
| 20 | 20 | 8 | 332 | 21 | 340 | 23 | 385 | 23 | NA | NA |
| 20 | 20 | 12 | 263 | 23 | 304 | 24 | 370 | 24 | NA | NA |
| 20 | 20 | 16 | 253 | 24 | 274 | 25 | 362 | 30 | NA | NA |
| 20 | 20 | 20 | 294 | 31 | 327 | 33 | 365 | 27 | NA | NA |

Figure 5.2: **Learning with Noisy Data**

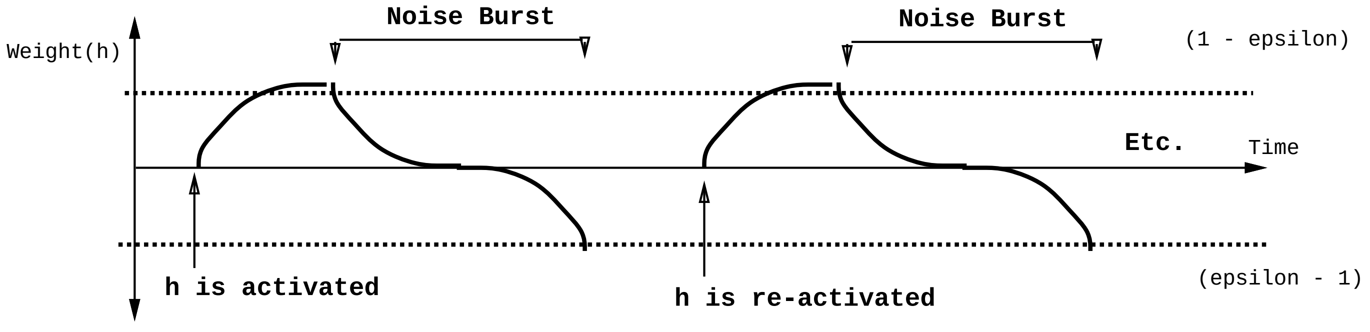

Figure 5.3: **Effect of Recurring Noise Bursts on Hypothesis Weight**

# Chapter 6

# Alternative Models

## 6.1 Bayesian Models

The algorithm presented in this thesis is based on the idea that a more "statistical" approach to gathering data may be necessary when learning in the presence of noisy examples. This raises the issue of whether existing models of statistical inference could be applied to the same problem.

Bayesian models can be used to calculate the probabilities of a set of hypotheses, given some notion of their prior probability and their probability given a particular input datum. There is an intuitive correlation between the elements of Bayes' Theorem and the components of the learning algorithm presented in this thesis, which is most apparent if we consider a multiplicative form of the Theorem.

$$P(h_i|x_1...x_n) \propto P(x_n|h_i) * P(h_i|x_1...x_{n-1}) \tag{6.1}$$

$$\propto P(x_n|h_i) * P(x_{n-1}|h_i) * P(x_{n-2}|h_i) * ... * P(x_1|h_i) * P(h_i) \tag{6.2}$$

The derivation captures the notion that the probability of a hypothesis $h_i$ given a set of data $x_1...x_n$ is the product of the probability of each datum given $h_i$, times the prior probability of $h_i$. The prior probability of a hypothesis can be related to its markedness in my model, and the probability of a datum given a hypothesis can be related to the results of the PARSE function; some probability can be assigned when the datum is correctly parsed by the hypothesis, and some other smaller probability can be assigned when the datum is not parsed correctly.

Given this link between Bayesian systems and the learning algorithm presented here, it is possible to imagine a Bayesian system that would update the probability of each hypothesis after each input datum was processed. The probability $P(h_i)$ of each hypothesis would initially be its prior probability; after each input datum was processed, $P'(h_i) = P(h_i) * P(x_n|h_i)$[1].

Although Bayesian models have well-known formal properties that might make this application desirable, there are several problematic issues to be dealt with:

- *Independence.* The straightforward application of Bayes' Theorem in this case depends on the assumption of independence, that is to say that $P(x_n|h_i)$ is independent of the presence or absence of any other $x_i$ in the data. This is definitely not a valid assumption for natural languages, where sentences can exhibit sharp conditional probabilities[2].

- *Cognitive Plausibility.* The intuitive application of a Bayesian system to the problem of parameter learning would involve manipulating the probabilities of all the hypotheses in the learning space. This is clearly undesirable from a cognitive perspective, since it is not feasible for the child learner to manipulate several hundred hypotheses all at once. It is possible that one could limit Bayesian learning to some sort of "cognitive approximation", in which only a certain number of hypotheses with the highest probability are considered for each input example[3];

[1]For example, Horning (1969) presented an application of the Bayesian model to the problem of learning context-free grammars by computing the *a posteriori* probability of each grammar given a set of data. A clear summary and criticism of Horning's work is given in (Kazman, 1991).

[2]Kevin Kelly, personal communication.

[3]Kevin Kelly, personal communication.

it is left to future work to explore this alternative.

- *Probability Function.* It is necessary to have some function that assigns meaningful values to $P(x_n|h_i)$. A realistic probability function would be required, but it is unclear how one might derive such a function for all input examples and all languages. In order to construct such a function, one must take into account the relative frequency of sentence types in a language, the probability of noisy examples, and the relationships (if any) between the occurrences of different types of sentences.

Despite these problems, Bayesian learning seems promising as an avenue of continued research in language learning, since it promotes the formalization of both the noise model and the distribution of sentences in the language, which are important considerations for any model which attempts to learn in the presence of both noisy examples and infrequently occurring examples. Nevertheless, it remains an open research question as to whether a computationally feasible cognitive approximation of a Bayesian application exists for parameter setting. A more detailed investigation of this alternative is beyond the scope of the present work.

## 6.2 Connectionist Models

Connectionist or "neural-net" models have been the subject of significant research in cognitive science (for example, see (McClelland and Rumelhart, 1986) and the references cited there). There has been recent interest in applying connectionist models to the problem of stress assignment in phonology (Wheeler and Touretzky, 1989) as well as the problem of learning stress assignment rules (Gupta and Touretzky, 1991). However, connectionist models have been the subject of great controversy when applied to language processing and the problem of language learning (cf. Pinker and Mehler, eds., 1988, and the references cited there).

I will not review all of the arguments that have been raised for and against connectionist models. I will briefly mention what I feel are the main criticisms of current connectionist approaches to language learning.

- *Naturalness*. Connectionist models can learn just about anything, given enough hidden layers of units, fine-tuning of the network, and massive amounts of training data (this approach has been referred to as "beat it to death with backprop")[4]. A disadvantage to this approach is that such models often lack a plausible definition of what can and cannot be a natural language[5]. In addition, even when learning natural languages, connectionist models exhibit behavior that is never observed in children[6].

- *Complexity*. The computational complexity of connectionist networks is considerable, especially when they are intended to learn the entire processing mechanism (e.g., the stress processor) from scratch given just a set of training data. One could imagine a hybrid model that used a connectionist network to set the values of parameters in a pre-existing symbolic processor; nevertheless, with no clear understanding of the relative benefit of such an approach, it would fall prey to Occam's Razor. In addition, the convergence time of a connectionist network is often quite long (for example, Gupta's perceptron model of stress assignment required over 580,000 examples to learn some of the stress patterns in its search space).

[4]Touretzky, personal communication.

[5]This follows from the operating assumption that connectionist models don't require an explicit "innate" notion of structure, but rather derive their implicit notion of structure by configuring the network weights during learning trials. It follows, then, that a connectionist network trained on unnatural data would be able to learn an unnatural language. It has been claimed that this is not conclusive evidence against connectionist models; since we cannot perform experiments in which we present a child with only examples from an unnatural language, we cannot gather empirical confirmation that the innate structure of the mind disallows the learning of "unnatural" languages (Prahlad Gupta, personal communication).

[6]The examples cited by Pinker and Prince (1988) in discussion of the Rumelhart and McClelland verb learning model include such errors as *membled*, as the past tense of *mail*, *typeded* as the past tense of *type*, etc. For further discussion of the lack of empirical fidelity in connectionist models, see (Kazman, 1991) or (Pinker and Prince, 1988).

The lack of naturalness makes connectionist models less satisfying from the viewpoint of cognitive plausibility. Although it would be interesting to see how the basic backpropagation architecture could be limited in ways that would produce more natural behavior (and predictable errors), the complexity of connectionist applications makes them less preferable to other methods (e.g., numerically-weighted symbolic methods) that are computationally more tractable and cognitively more plausible.

## 6.3 Other Learning Models

So far in this thesis I have discussed other learning models that fall into the following categories:

- Alternative models of parameter setting (e.g. YOUPIE, DARWIN);
- Applications in a related domain (such as Gupta's perceptron model of stress assignment);
- Historical predecessors of the parameter-setting model (e.g., Berwick's model, the work of Wexler and Culicover, etc.);
- Potentially applicable models of statistical inference (e.g., Bayesian models).

For the sake of completeness, I will briefly mention some other language learning models. Since they do not relate directly to the narrow focus of this thesis (parameter setting), I will not discuss them in any detail. These brief descriptions are offered to the reader who is interested in a broader investigation of topics in language learning.

- *Learning Phrase Structure Rules*. In addition to previously mentioned work on learning phrase structure rules (Berwick, 1985) and transformational grammar (Wexler and Culicover, 1980), it is also worthwhile to mention the work of Pinker (1984), which proposes a learning system for LFG grammars from a developmental perspective. Pinker's work defines a set of deductive procedures that are used to learn phrase structure rules and other elements of syntactic processing (inflection, control, etc.) given information about the semantics and grammatical relations in the input.

- *Lexical Learning*. Kazman (1991) has focussed not on the learning of parameters but rather the induction of the lexicon and its properties in early child language. In some sense Kazman's work is a valuable counterpart to all work in parameter setting, which has abstracted away from lexical learning at some level. Kazman's system, BABEL, first learns the structure and contents of the lexicon through statistical induction, and then learns the syntactic structure of a lexical category following a process of deduction. BABEL makes empirically valid predictions about the acquisition of functional categories in English and several other languages.

  Bates and MacWhinney (1987) have developed the "competition" model, in which the lexicon is taken to be the comprehensive repository of all linguistic entities and relations. Learning takes place as connections between units of lexical structure are re-calulated in a sort of marker-passing, spreading-activation scheme as new input examples are processed.

- *Semantic Models*. Anderson's LAS system (1974, 1975) and Selfridge's CHILD system (1980) are models which infer the syntactic structure of the target language when provided with complete knowledge of the meaning of each sentence. Anderson's system carried successive refinements of an ATN grammar; Selfridge's system associated rules of surface realization with each slot in a semantic case frame. Although intended as comprehensive models of acquisition, both systems suffer from a lack of a complete theory of syntax and lack of adherence to the cognitive criteria presented in Section 1.3 (see (Kazman, 1991) for a criticism of these two models from this perspective).

- *Machine Learning and Artificial Intelligence.* Langley and Carbonell (1987) summarize the state of language learning research in Artificial Intelligence, and discuss several systems, including Wolff's SNPR system (1978), which focussed on the segmentation and aggregation of the input stream during learning, and Langley's AMBER system, which learns mappings from meaning (represented using semantic networks) to sentences; these mappings are represented as production rules in the PRISM production system (Langley and Neches, 1981).

# Chapter 7

# Conclusions

In this chapter I summarize the accomplishments of the thesis and sketch the possibilities for future work based on the model and results presented here.

## 7.1 Summary of the Thesis

### 7.1.1 General Contributions

This thesis makes the following general contributions to the field of computational linguistics:

- A simulation model that brings together linguistic theory and a learning algorithm in an empirically testable model;
- A concrete investigation of what is necessary to construct a particular learning model and test its computational and cognitive plausibility;
- A clear, concise definition of a particular learning architecture and a learning algorithm;
- A computational implementation of the model which can be tuned, extended, etc., and serves as a experimental testbed for algorithm refinement and learning

simulation in future work.

### 7.1.2 Linguistic Theory and Parameter Setting

This thesis raises particular questions about linguistic theory and parameter setting, which reflect the usefulness of this study as a source of feedback to both fields:

- *What is an Adequate Definition of a Parameter?* Although (Dresher and Kaye, 1990) presented a detailed account of phonological parameters and their dependencies, the implementation and subsequent testing of the stress learner presented in this thesis uncovers the inadequacy of their parameter definitions; specifically, the system of parameters and values admits several sets of languages that are distinct in their parameter settings but identical in their stress patterns. Since it is an underlying assumption of generative linguistics that adults who speak the same language share similar if not identical knowledge states, it is unsatisfying if a theory allows several different ways to account for the same data. Future work in parameter setting (especially work which proposes a comprehensive system of parameters) should be prepared to address this issue and show that the parameters proposed do not allow variation that is not evidenced in the data for a set of languages.

- *What is an Adequate Description of Triggering Data?* Most work in parameter setting has focussed on whether or not there exist triggering examples for a particular parameter value, and has avoided the issue of frequency of triggering data. The model and experimental results presented here clearly illustrate the need for a precise definition of triggering data which includes a frequency distribution, especially when learning in noisy environments.

- *How Does Expressibility Interact with Learnability and Diachronic Change?* As noted in Section 3.6, there are some language pairs in the Dresher and Kaye parameter space that overlap by greater than 98%. This raises the issue of learnability, specifically, whether or not such distinctions are learnable by humans. Since language learners take longer to learn constructions that are infrequently expressed in the input data, it follows that some constructions that are sufficiently infrequent might not be learnable at all. Recent work has proposed that such "ambiguous" parameter settings are a mechanism for diachronic language change (Clark, 1991). Future work should address the plausibility of parameter values with very low expressibility and how they interact with learnability.

### 7.1.3 Computational Tractability

In contrast to models which require hundreds of thousands or millions of input examples to select a target language, the model presented in this thesis can select a target language in fairly large domains (up to 30 parameters) after less than 1000 input examples, even when learning in noisy environments.

With up to a 12.5% uniform distribution of noisy examples, the learning algorithm successfully selects a target language in spaces of up to 20 parameters. Since adults make speech errors much less frequently than this, the result implies that the learning algorithm is resilient to levels of noise in the input which are far greater than the uniform distribution of noisy examples in the child's environment.

### 7.1.4 Cognitive Plausibility

I now return to Pinker's cognitive criteria in order to evaluate the plausibility of the model presented in the thesis:

- *The Learnability Condition*. The learning algorithm is able to learn the target language in a wide variety of learning environments. Future work should focus on the definition of the learning environments actually experienced by children.

- *The Equipotentiality Condition*. The learning algorithm can learn any of the languages in its search space (with the exception of those degenerate examples discussed in Section 3.6). The learning algorithm is language-independent, and does not contain any learning mechanism that is particular to the constructions of a certain language.

- *The Time Condition*. The learning algorithm learns each target language in a short finite time span (i.e., on the order of 1000 examples in the domain tested). Future work should attempt to determine the normal time span required by children learning the rules of stress assignement.

- *The Input Condition*. The learning algorithm does not require types of input information that are not available to the child; in particular, it does not require the use of indirect negative evidence. Future work should address the question of what are "natural" distributions of grammatical and noisy examples in the learning environment.

- *The Developmental Condition*. The learning model makes predictions about the intermediate stages of acquisition that are based on the assumptions of the linguistic model (its default parameter settings) and the data encountered by the learner. Future work should address the collection of empirical findings concerning the developmental stages of metrical phonology.

- *The Cognitive Condition*. The learning algorithm places a well-defined bound on the number of active hypotheses ($n$, where $n$ is the number of active parameters) and performs a minimum of processing for each active hypothesis, updating the hypothesis weight once the example has been parsed.

The model presented in this thesis has a combination of characteristics (no use of indirect evidence, robustness in noisy environments, ability to avoid hypothesis ordering problems, limited non-determinism, conservative hypothesis selection) that make it preferable on grounds of both computational tractability and cognitive plausibility when compared to existing parameter-setting models. Nevertheless, there are many areas that are open to improvement; I will return to a discussion of some of these in Section 7.2.

### 7.1.5 Evaluation

- The learning model was successfully tested on a large volume of test data in two different learning applications (metrical phonology and abstract syntax) (cf. Sections 3.6 and 4.3);
- The learning model was also tested for learning in noisy environments (cf. Section 5.2);
- The learning algorithm successfully solves "tough" learning problems that have been identified in the literature (cf. Section 2.8);
- The learning model satisfies the proposed cognitive constraints to a higher degree than existing parameter-setting models (cf. Section 7.1.4).

## 7.2 Future Work

### 7.2.1 Immediate Extensions to the Present Work

Every thesis is limited in scope; nevertheless, there are several possible extensions to this thesis which follow directly from the work presented here:

- *Learning on Infinite Texts with Noise Bursts*. As discussed in Section 5.3.1, the model could be modified to "remember" the net positive evidence for pruned hypothesis, and assign that weight to the hypothesis if it were reactivated, ensuring convergence in the limit.

- *Avoiding Overgeneralization and Bursts of Noisy Examples*. As discussed in Section 5.3.2, there are at least two ways in which the model could be revised to avoid the problem of overgeneralization when a burst of noisy examples is presented to the learner at precisely the moment when a subset has been introduced by the Relaxed Subset Principle. Future work should consider this problem and an appropriate solution.

- *Experimenting with the Algorithm Coefficients*. The weighting coefficients, damping factor $k$, and $\epsilon$ were all kept constant during empirical testing of the learning algorithm. It would be interesting to consider changes in the learner's behavior that might result from varying these coefficients.

- *Further Refinement of the Phonological Parameters*. The stress clash avoidance parameters and non-iterativity parameters are subject to further refinement (through revised definition, parameterized ordering in processing, or the addition of parameter dependencies), which would decrease the ambiguity of the parameter model (cf. Section 3.6.4).

- *Testing the Learner on Noisy Data with Phonological Parameters*. For the sake of expediency, I tested the learner on noisy data in the domain of abstract syntactic parameters only. Future work should replicate the same experiments in the phonological domain.

- *Testing Other Learning Models in the Same Domain*. It would be revealing to test the DARWIN genetic algorithm learner on the same phonological domain explored by the limited non-deterministic learning algorithm, for the sake of direct comparison. Similarly, it would be revealing to test other available learning programs that are as easily adaptable to the domain.

### 7.2.2 Long-Range Investigation

The work presented in this thesis also suggests the following areas of long-term investigation:

- *Empirical Data and Stages in Phonological Development*. As mentioned earlier in Section 3.6.6, there is a dearth of material in the literature concerning developmental stages in phonological learning. The extent to which more stringent cognitive claims can be evaluated for parameter models depends directly on the collection of longitudinal data concerning the stages of phonological development across several languages. It would also be useful to evaluate the claims made by Archibald concerning the "interlanguage" of second language learners, through computational simulation using the present framework.

- *A Complete Model of Syntax*. Application of the learning algorithm to a completely implemented model of syntactic processing (like that described in (Gibson, 1991)) would most likely yield useful results beyond those obtained by considering abstract parameters only.

- *An Empirically Validated Noise Model for Natural Language*. In Section 5.4, I outlined the possible structure of a more comprehensive noise model for natural language. To complete such a model, and to improve the characteristics of the present learning algorithm when faced with bursts of noisy examples, a more thorough characterization of the child's language learning environment would be necessary. This would entail longitudinal data gathering, taking into account the child's utterances, adult utterances focussed at the child, adult utterances not focussed at the child but attended to by the child, adult utterances focussed at the child but not attended to by the child, etc. Only in this way could the notion of "salience" be made concrete and become part of a theory of filtering, which would more adequately explain why children are resilient to large bursts of noisy examples in the environment.

- *An Empirically Validated Frequency Model for Natural Language*. The learning algorithm presented here achieves certain results by assuming a uniform distribution of examples in the sample data; i.e., longer examples are just as frequent as shorter ones, and the internal structure of examples (e.g., the presence/absence of heavy syllables) is also uniformly distributed. Future work should address the issue of more realistic distributions of input examples with respect to each target language. Ultimately, this data should be linked to a continuing investigation of the role of triggering data and the notion of a "minimal bound" on data frequency to ensure learnability[1].

- *Parameter-Setting and Models of Statistical Inference*. Further investigation of Bayesian models, beyond the learning of context-free grammars, might yield a plausible cognitive approximation of a Bayesian learner for parameter-based language learning.

[1]Robin Clark, personal communication

# Bibliography

[1] Anderson (1974) "Language acquisition by computer and a child," Human Performance Center Technical Report No. 55., Ann Arbor: University of Michigan.

[2] Anderson (1975) "Computer simulation of a language acquistion system: A first report," in R. Solso (ed.), *Information Processing and Cognition: The Loyola Symposium*, Washington, D.C.: Erlbaum.

[3] Angluin, D. (1978) "Inductive inference of formal languages from positive data," *Information and Control*, 45, pp. 117-135.

[4] Archibald, J. (1989) "Metrical Parameters: A Neglected Area of Research," Boston University Conference on Language Development.

[5] Archibald, J. (1990) "Acquisition of English Metrical Parameters by Polish and Hungarian Speakers," Boston University Conference on Language Development.

[6] Archibald, J. (1991) "Quantity-Sensitivity and the Word Tree in the Interlanguage of Adult Hungarian Speakers," Boston University Conference on Language Development.

[7] Baker, C.L. (1979) "Syntactic Theory and the Projection Problem," *Linguistic Inquiry*, 10:4, pp. 533-82.

[8] Baker, C.L. and J. McCarthy, eds. (1981) *The Logical Problem of Language Acquisition*, Cambridge: MIT Press.

[9] Bates, E. and B. MacWhinney (1987) "Competition, Variation and Language Learning," in B. MacWhinney, ed., *Mechanisms of Language Acquisition*, Hillsdale, NJ: Lawrence Erlbaum.

[10] Berwick, R. (1985) *The Acquisition of Syntactic Knowledge*, Cambridge: MIT Press.

[11] Bloom, P. (1988) "Evidence for a Predication Parameter in the Child's Grammar," ms., MIT.

[12] Booker, L., D. Goldberg and J. Holland (1990) "Classifier Systems and Genetic Algorithms," in J. Carbonell, ed., *Machine Learning: Paradigms and Methods*, Cambridge, MA: MIT Press.

[13] Borer, H., and Wexler, K. (1987) "The Maturation of Syntax," in Williams and Roeper, eds., *Parameter Setting*, Dordrecht: Reidel.

[14] Brown, R., C. B. Cazden, and U. Bellugi (1969) "The Child's Grammar from I to III," in J. P. Hill, ed., *Minnesota Symposium on Child Psychology*, Volume 2, Minneapolis: University of Minnesota Press.

[15] Brown, R., and Hanlon, C. (1970) "Derivational Complexity and the Order of Acquisition of Child Speech," in J. R. Hayes, ed., *Cognition and the Development of Language*, New York: Wiley and Sons.

[16] Brown, R. (1973) *A First Language: The Early Stages*, Cambridge: Harvard University Press.

[17] Chomsky, N. (1965) *Aspects of the Theory of Syntax*, Cambridge: MIT Press.

[18] Chomsky, N. (1981) "Principles and Parameters in Syntactic Theory," in N. Hornstein and D. Lightfoot, eds., *Explanation in Linguistics: The Logical Problem of Language Acquisition*, New York: Longman.

[19] Chomsky, N. (1985) *Knowledge of Language*, New York: Praeger.

[20] Clark, R. (1988) "On the Relationship Between the Input Data and Parameter Setting," *Proceedings of the 19th North East Linguistics Society*, Cornell University.

[21] Clark, R. (1988) "The Problem of Causality in Models of Language Learnability," paper presented at the 13th Boston University Conference on Language Development, Boston, MA.

[22] Clark, R. (1990a) "Some Elements of a Proof for Language Learnability," ms., Université de Genève.

[23] Clark, R. (1990b) "Parameter Setting and Natural Selection: A Computer Simulation," ms., Université de Genève.

[24] Clark, R. (1990c) "Causality, Natural Selection and Parameter Setting," paper presented at the Boston University Conference on Language Development.

[25] Clark, R., and I. Roberts (1991) "A Computational Model of Language Learnability and Language Change," ms., Université de Genève.

[26] Dresher, B. E. (1989) "A Parameter-Based Learning Model for Metrical Phonology," paper presented at the 14th Boston University Conference on Language Development, Boston, MA.

[27] Dresher, B. E. and J. D. Kaye (1990). "A Computational Learning Model for Metrical Phonology," *Cognition*, 34, pp. 137-195.

[28] Everett, D. L. (1988) "On metrical constituent structure in Piraha phonology," *Natural Language and Linguistic Theory*, 6:2, pp. 207-246.

[29] Everett, D. L. (1989) "The Minimal Word Constraint and Levels of Representation in Kama and Banawa," manuscript, University of Pittsburgh.

[30] Ferguson, C. A. and D. I. Slobin (1973) *Studies of Child Language Development*, Holt, New York: Rinehart and Winston.

[31] Flynn, S. (1987) *A Parameter-Setting Model of L2 Acquisition: Experimental Studies in Anaphora*, Dordrecht: Reidel.

[32] Fong, S. (1990) "The Computational Implementation of Principle-Based Parsers," in C. Tenny, ed., *The MIT Parsing Volume 1988-1989*, Center for Cognitive Science, Massachusetts Institute of Technology.

[33] Gerken, L. A. (1987) "Function morphemes in young children's speech perception and production," PhD dissertation, Columbia University.

[34] Gerken, L. A. (1990) "A metrical account of children's subjectless sentences," ms., State University of New York at Buffalo.

[35] Gibson, E. (1987) "Garden Pathing in a Parser with Parallel Architecture," *Proceedings of the Eastern States Conference on Linguistics*, Ohio State University, Columbus, OH.

[36] Gibson, E. (1991). "A Computational Theory of Human Linguistic Processing: Memory Limitations and Processing Breakdown," Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA.

[37] Goldberg, D. E. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*, Reading, MA: Addison-Wesley.

[38] Graham, R. L., D. Knuth, and O. Patashnik (1989) *Concrete Mathematics: A Foundation for Computer Science*, Reading, MA: Addison-Wesley.

[39] Gross, K. (1989) "CLUE: Concept Learning Using Experimentation," PhD Thesis Proposal, Carnegie Mellon University, School of Computer Science.

[40] Gupta, P., and D. Touretzky (1991) "What a perceptron reveals about metrical phonology," ms., Carnegie Mellon University.

[41] Haider, H., and M. Prinzhorn, eds. (1986) *Verb Second Phenomena in Germanic Languages*, Dordrecht: Foris.

[42] Halle, M., and R. Vernaud (1987) *An Essay on Stress*, Cambridge: MIT Press, Cambridge.

[43] Halle, M. (1990) "Respecting Metrical Structure," *Natural Language and Linguistic Theory*, Vol. 8, No. 2, pp. 149-176.

[44] Hammond, M. (1986) "The obligatory branching parameter in metrical theory," *Natural Language and Linguistic Theory*, 4, pp. 185-228.

[45] Hammond, M. (1990a). "On deriving ternarity," ms, Univ. of Arizona.

[46] Hammond, M. (1990b). "Parameters of Metrical Theory and Learnability," ms, Univ. of Arizona.

[47] Hayes, B. (1981). *A Metrical Theory of Stress Rules*, MIT Doctoral Dissertation, distributed by IULC, Bloomington, IA.

[48] Horning, J. J. (1969) "A Study of Grammatical Inference," PhD dissertation, Dept. of Computer Science, Stanford University, Stanford, CA.

[49] Horwich, P. (1982) *Probability and Evidence*, New York: Cambridge University Press.

[50] Howson, C. and P. Urbach (1989) *Scientific Reasoning: The Bayesian Approach*, La Salle, IL: Open Court.

[51] Hyams, N. (1986) *Language Acquisition and the Theory of Parameters*, Dordrecht: Reidel.

[52] Kazman, R. (1988) "Null Arguments and the Acquisition of Case and Infl," paper presented at the 13th Boston University Conference on Language Development, Boston, MA.

[53] Kazman, R., and E. Gibson (1988). "An Implementation of a Language Acquisition Model," talk presented at the MIT Parsing Seminar.

[54] Kazman, R. (1991) "The Induction of the Lexicon and The Early Stages of Grammar," PhD Thesis, Carnegie Mellon University.

[55] Koopman, H. (1983) *The Syntax of Verbs: From Verb Movement Rules in the Kru Languages to Universal Grammar*, Dordrecht: Foris.

[56] Langley, P. and J. Carbonell (1987) "Language Acquisition and Machine Learning," in MacWhinney, ed., *Mechanisms of Language Acquisition*, Hillsdale, NJ: Lawrence Erlbaum.

[57] Langley, P. and R. Neches (1981) PRISM User's Manual (technical report) Computer Science Department, Carnegie Mellon University, Pittsburgh, PA.

[58] Lenneberg, E. (1967) *Biological Foundations of Language*, New York: Wiley.

[59] Levin, J, (1985) *A Metrical Theory Of Syllable Structure*, Unpublished MIT Dissertation.

[60] Liberman, M. and A. S. Prince (1977). "On Stress and Linguistic Rhythm," *Linguistic Inquiry*, 8, pp. 249-336.

[61] Lightfoot, D. (1989) "The child's trigger experience: Degree-0 learnability," ms. University of Maryland.

[62] Lindley, D. V. (1965) *Introduction to Probability and Statistics from a Bayesian View Point*, Cambridge: University Press.

[63] Lust, B. (ed.) (1986) *Studies in the Acquisition of Anaphora: Vol 1, Defining the Constraints*, Dordrecht: Foris.

[64] Lust, B. (ed.) (1986) *Studies in the Acquisition of Anaphora: Vol 2, Applying the Constraints*, Dordrecht: Foris.

[65] Lyons, J. (1968) *Introduction to Theoretical Linguistics*, Cambridge University Press.

[66] Mehler, J., P. Jusczyk, G. Lambertz, N. Halsted, J. Bertoncini and C. Amiel-Tison (1988) "A precurson of language acquisition in young infants," *Cognition*, 29:143-178.

[67] Newport, E., Gleitman, H., and Gleitman, L. (1977) "Mother, please, I'd rather do it myself: Some effects and non-effects of maternal speech style," in Snow and Ferguson, eds., *Talking to Children: Language Input and Acquisition*, New York: Cambridge University Press.

[68] Nishigauchi, T., and T. Roeper (1987) "Deductive Parameters and the Growth of Empty Categories," in Roeper and Williams, eds., *Parameter Setting*, Dordrecht: Reidel.

[69] Nyberg, E. (1987) "Parsing and the Acquisition of Word Order," *Proceedings of the Eastern States Conference on Linguistics*, Ohio State University, Columbus, OH.

[70] Nyberg, E. (1989) "Weight Propagation and Parameter Setting: A Computational Model Of Language Learning," Phd Thesis Proposal, Carnegie Mellon University.

[71] Nyberg, E. (1990) "A Limited Non-Deterministic Parameter-Setting Model," *Proceedings of the North East Linguistics Society*, Université de Québec à Montréal.

[72] Ochs, E. (1985) "Variation and Error: A Sociolinguistic Approach to Language Acquisition in Samoa" in Slobin, D. (ed) *The Crosslinguistic Study of Language Acquisition, Volume 1: The Data*. Lawrence Erlbaum, Associates, Hillsdale, New Jersey.

[73] Osherson, D., M. Stob and S. Weinstein (1986) *Systems that Learn*, Cambridge: MIT Press.

[74] Peters, S. (1972) "The Projection Problem: How is a Grammar to be Selected?," in S. Peters, ed., *Goals of Linguistic Theory*, Englewood Cliffs, NJ: Prentice-Hall.

[75] Phinney, M. (1987) "The Pro-Drop Parameter in Second Language Acquisition," in Roeper and Williams, eds., *Parameter Setting*, Dordrecht: Reidel.

[76] Pinker, S. (1979) "Formal Models of Language Acquisition," *Cognition*, 7:217-283.

[77] Pinker, S. (1984) *Language Learnability and Language Development*, Cambridge: Harvard University Press.

[78] Pinker, S. and J. Mehler, eds. (1988) *Connections and Symbols*, Cambridge: MIT Press.

[79] Pinker, S. and A. Prince (1988) "On language and connectionism: Analysis of a parallel distributed processing model of language acquisition," in Pinker and Mehler, eds., Connections and Symbols, Cambridge: MIT Press.

[80] Pollock, J. Y. (1989) "Verb Movement, Universal Grammar, and the Structure of IP," *Linguistic Inquiry*, 20.

[81] Prince, A. S. (1983). "Relating to the Grid," *Linguistic Inquiry*, 14, pp. 19-100.

[82] Quine, W. V. O. (1960). *Word and Object*, Cambridge, MA: MIT Press.

[83] Rumelhart, D., J. McClelland and the PDP Research Group (1986). *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 2: Psychological and biological models.* Cambridge, MA: Bradford Books.

[84] Selfridge (1980) "A computer model of child language leraning," *Proceedings of the 1st annual conference of the American Association for Artificial Intelligence*.

[85] Solan, L. (1987) "Parameter Setting and the Development of Pronouns and Reflexives," in Roeper and Williams, eds., *Parameter Setting*, Dordrecht: Reidel.

[86] Travis, L. (1984) *Parameters and Effects of Word Order Variation*, unpublished MIT PhD thesis.

[87] Ullman, M., S. Pinker, M. Hollander, A. Prince, T. Rosen (1989) "Growth of Regular and Irregular Vocabulary and the Onset of Overregularization," paper presented at the 14th Boston University Conference on Language Development.

[88] Wexler, K. (1987) "Maturation, Continuity, Parameters and Reorganization in Language Acquisition," paper presented at the 12th Annual Boston University Conference on Language Development.

[89] Wexler, K., and Culicover, P. (1980) *Formal Principles of Language Acquisition*, Cambridge: MIT Press.

[90] Wexler, K., and Manzini, R. (1987) "Parameters and Learnability in Binding Theory," in Roeper and Williams, eds., *Parameter Setting*, Dordrecht: Reidel.

[91] Wheeler, D. and D. Touretsky (1989) "A connectionist implementation of cognitive phonology," Technical Report CMU-CS-89-144, Department of Computer Science, Carnegie Mellon University.

[92] Wolff, J. G. (1978) "Grammar discovery as data compression," *Proceedings of the AISB/GI Conference on Artificial Intelligence*, hamburg, West Germany.

# Appendix A

# A Sample Trace of the Learning Algorithm

```
      Activating 0000

[1]   Active Nodes: 0000
      Datum: 0101
      Adding -5 to 0000 --> -0.2449188

[2]   Active Nodes: 0000
      Datum: 0101
      Adding -5 to 0000 --> -0.4621162

[3]   Active Nodes: 0000
      Datum: 0101
      Adding -5 to 0000 --> -0.635147

[4]   Active Nodes: 0000
      Datum: 0101
      Adding -5 to 0000 --> -0.761591

[5]   Active Nodes: 0000
```

```
      Datum: 0101
      Adding -5 to 0000 --> -0.848281

[6]   Active Nodes: 0000
      Datum: 0101
      Adding -5 to 0000 --> -0.905146
      Pruning 0000

      No active nodes. 1 pruned node(s).
      Activating nodes 1-adjacent to 0000.
      Activating: 0001 0010 0100 1000

[7]   Active Nodes: 0001 0010 0100 1000
      Datum: 0101
      Adding -5 to 0001 --> -0.2449188
      Adding -5 to 0010 --> -0.2449188
      Adding -5 to 0100 --> -0.2449188
      Adding -5 to 1000 --> -0.2449188

[8]   Active Nodes: 0001 0010 0100 1000
      Datum: 0101
      Adding -5 to 0001 --> -0.4621162
      Adding -5 to 0010 --> -0.4621162
      Adding -5 to 0100 --> -0.4621162
      Adding -5 to 1000 --> -0.4621162

[9]   Active Nodes: 0001 0010 0100 1000
      Datum: 0101
      Adding -5 to 0001 --> -0.635147
      Adding -5 to 0010 --> -0.635147
      Adding -5 to 0100 --> -0.635147
      Adding -5 to 1000 --> -0.635147

[10]  Active Nodes: 0001 0010 0100 1000
```

```
      Datum: 0101
      Adding -5 to 0001 --> -0.761591
      Adding -5 to 0010 --> -0.761591
      Adding -5 to 0100 --> -0.761591
      Adding -5 to 1000 --> -0.761591

[11]  Active Nodes: 0001 0010 0100 1000
      Datum: 0101
      Adding -5 to 0001 --> -0.848281
      Adding -5 to 0010 --> -0.848281
      Adding -5 to 0100 --> -0.848281
      Adding -5 to 1000 --> -0.848281

[12]  Active Nodes: 0001 0010 0100 1000
      Datum: 0101
      Adding -5 to 0001 --> -0.905146; Pruning 0001
      Adding -5 to 0010 --> -0.905146; Pruning 0010
      Adding -5 to 0100 --> -0.905146; Pruning 0100
      Adding -5 to 1000 --> -0.905146; Pruning 1000

      No active nodes. 4 pruned node(s).
      Randomly picking one pruned node: 0100
      Activating nodes local to 0100.
      Activating: 0101 0110 1100

[13]  Active Nodes: 0101 0110 1100
      Datum: 0101
      Adding 1 to 0101 --> 0.0499573
      Adding -5 to 0110 --> -0.2449188
      Adding -5 to 1100 --> -0.2449188

[14]  Active Nodes: 0101 0110 1100
      Datum: 0101
      Adding 1 to 0101 --> 0.0996666
      Adding -5 to 0110 --> -0.4621162
      Adding -5 to 1100 --> -0.4621162

[15]  Active Nodes: 0101 0110 1100
      Datum: 0101
      Adding 1 to 0101 --> 0.1488838
      Adding -5 to 0110 --> -0.635147
      Adding -5 to 1100 --> -0.635147

[16]  Active Nodes: 0101 0110 1100
      Datum: 0101
      Adding 1 to 0101 --> 0.1973743
      Adding -5 to 0110 --> -0.761591
      Adding -5 to 1100 --> -0.761591

[17]  Active Nodes: 0101 0110 1100
      Datum: 0101
      Adding 1 to 0101 --> 0.244917
      Adding -5 to 0110 --> -0.848281
      Adding -5 to 1100 --> -0.848281

[18]  Active Nodes: 0101 0110 1100
      Datum: 0101
      Adding 1 to 0101 --> 0.291313
      Adding -5 to 0110 --> -0.905146; Pruning 0110
      Adding -5 to 1100 --> -0.905146; Pruning 1100

[19]  Active Nodes: 0101
      Datum: 0101
      Adding 1 to 0101 --> 0.336376

[20]  Active Nodes: 0101
      Datum: 0101
      Adding 1 to 0101 --> 0.3799496
```

```
[21]  Active Nodes: 0101
      Datum: 0101
      Adding 1 to 0101 --> 0.4219

[22]  Active Nodes: 0101
      Datum: 0101
      Adding 1 to 0101 --> 0.462118

[23]  Active Nodes: 0101
      Datum: 0101
      Adding 1 to 0101 --> 0.500523

[24]  Active Nodes: 0101
      Datum: 0101
      Adding 1 to 0101 --> 0.537052

[25]  Active Nodes: 0101
      Datum: 0101
      Adding 1 to 0101 --> 0.571672

[26]  Active Nodes: 0101
      Datum: 0101
      Adding 1 to 0101 --> 0.604368

[27]  Active Nodes: 0101
      Datum: 0101
      Adding 1 to 0101 --> 0.635149

[28]  Active Nodes: 0101
      Datum: 0101
      Adding 1 to 0101 --> 0.664038

[29]  Active Nodes: 0101
```

```
      Datum: 0101
      Adding 1 to 0101 --> 0.691071

[30]  Active Nodes: 0101
      Datum: 0101
      Adding 1 to 0101 --> 0.716301

[31]  Active Nodes: 0101
      Datum: 0101
      Adding 1 to 0101 --> 0.739784

[32]  Active Nodes: 0101
      Datum: 0101
      Adding 1 to 0101 --> 0.761595

[33]  Active Nodes: 0101
      Datum: 0101
      Adding 1 to 0101 --> 0.781807

[34]  Active Nodes: 0101
      Datum: 0101
      Adding 1 to 0101 --> 0.800501

[35]  Active Nodes: 0101
      Datum: 0101
      Adding 1 to 0101 --> 0.817755

[36]  Active Nodes: 0101
      Datum: 0101
      Adding 1 to 0101 --> 0.833654

[37]  Active Nodes: 0101
      Datum: 0101
      Adding 1 to 0101 --> 0.848284
```

```
[38]  Active Nodes: 0101
      Datum: 0101
      Adding 1 to 0101 --> 0.861725

[39]  Active Nodes: 0101
      Datum: 0101
      Adding 1 to 0101 --> 0.874056

[40]  Active Nodes: 0101
      Datum: 0101
      Adding 1 to 0101 --> 0.885355

[41]  Active Nodes: 0101
      Datum: 0101
      Adding 1 to 0101 --> 0.895695

[42]  Active Nodes: 0101
      Datum: 0101
      Adding 1 to 0101 --> 0.905153

[43]  Active Nodes: 0101

Success! Converged to a target language
         after 42 data examples.
```

# Appendix B

# A Trace of a Subset Learning Problem

```
      Activating H1

[1]   Active Nodes: :H1
      Datum: H7
      Adding 10 to H1 --> 0.462118

[2]   Active Nodes: :H1
      Datum: H7
      Adding 10 to H1 --> 0.761595

[3]   Active Nodes: :H1
      Datum: H7
      Adding 10 to H1 --> 0.905149
        Subsets: (H2 H3)
        Activating inactive subsets: H2 H3
      Adding 10 to H2 --> 0.462118
      Adding 10 to H3 --> 0.462118

[4]   Active Nodes: :H1 :H2 :H3
      Datum: H7
      Adding 10 to H1 --> 0.964029
      Adding 10 to H2 --> 0.761595
      Adding 10 to H3 --> 0.761595

[5]   Active Nodes: :H1 :H2 :H3
      Datum: H7
      Adding 10 to H1 --> 0.986616
      Adding 10 to H2 --> 0.905149
        Subsets: (H4)
        Supersets: (H1)
        Activating inactive subset: H4
        Deactivating active superset: H1
      Adding 10 to H3 --> 0.905149
        Subsets: (H4)
        Supersets: (H1)
        Subset H4 already activated.
        Superset H1 already pruned.
      Adding 10 to H4 --> 0.462118

[6]   Active Nodes: :H2 :H3 :H4
      Datum: H7
      Adding 10 to H2 --> 0.964029
      Adding 10 to H3 --> 0.964029
      Adding 10 to H4 --> 0.761595

[7]   Active Nodes: :H2 :H3 :H4
      Datum: H7
      Adding 10 to H2 --> 0.986616
      Adding 10 to H3 --> 0.986616
      Adding 10 to H4 --> 0.905149
        Subsets: (H5 H8)
        Supersets: (H3 H2)
        Activating inactive subsets: H5 H8
        Deactivating active supersets: H2 H3
      Adding 10 to H5 --> 0.462118
      Adding -10 to H8 --> -0.462117
```

```
[8]   Active Nodes: :H4 :H5 :H8
      Datum: H7
      Adding 10 to H4 --> 0.964029
      Adding 10 to H5 --> 0.761595
      Adding -10 to H8 --> -0.761594

[9]   Active Nodes: :H4 :H5 :H8
      Datum: H7
      Adding 10 to H4 --> 0.986616
      Adding 10 to H5 --> 0.905149
        Subsets: (H6 H7)
        Supersets: (H4)
        Activating inactive subsets: H6 H7
        Deactivating active superset: H4
      Adding -10 to H8 --> -0.905148
      Pruning H8
      Adding -10 to H6 --> -0.462117
      Adding 10 to H7 --> 0.462118

[10]  Active Nodes: :H5 :H6 :H7
      Datum: H7
      Adding 10 to H5 --> 0.964029
      Adding -10 to H6 --> -0.761594
      Adding 10 to H7 --> 0.761595

[11]  Active Nodes: :H5 :H6 :H7
      Datum: H7
      Adding 10 to H5 --> 0.986616
      Adding -10 to H6 --> -0.905148
      Pruning H6
      Adding 10 to H7 --> 0.905149
        Supersets: (H5)
        Deactivating active superset: H5
```

```
[12]  Active Nodes: :H7
Success! Converged to a target language
         after 11 data examples.
T
*
```

# Appendix C

# Examples of Metrical Stress Processing

```
Input segments: #([KO]<0> [KO]<0> [KO]<0>)

Metrical constituents (LEFT-headed, UNBOUNDED feet built from the LEFT):

   [  1        0        0      ]
     / \      / \      / \
    K   *    K   *    K   *
       / \      / \      / \
      O        O        O

Word tree (strong on the LEFT):

   [  2 +      0        0      ]
     / \      / \      / \
    K   *    K   *    K   *
       / \      / \      / \
      O        O        O
```

Figure C.1: **Latvian Example**

```
Input segments: #([KO]<0> [KO]<0> [KO]<0>)

Metrical constituents (LEFT-headed, UNBOUNDED feet built from the LEFT)
  (the LEFT-most segment is extrametrical):

      0       [  1        0      ]
     / \        / \      / \
    K   *      K   *    K   *
       / \        / \      / \
      O          O        O

Word tree (strong on the LEFT):

      0       [  2 +      0      ]
     / \        / \      / \
    K   *      K   *    K   *
       / \        / \      / \
      O          O        O
```

Figure C.2: **Lakota Example**

```
Input segments: #([ME]<0> [RE]<0> [PET]<0>)

Metrical constituents (LEFT-headed, BINARY feet built from the LEFT):

   [  1        0      ][  1      ]
     / \      / \        / \
    M   *    R   *      P   *
       / \      / \        / \
      E        E          E   T

Word tree (strong on the LEFT):

   [  2 +      0      ][  1      ]
     / \      / \        / \
    M   *    R   *      P   *
       / \      / \        / \
      E        E          E   T


Input segments: #([YAN]<0> [GAR]<0> [MA]<0> [TA]<0>)

Metrical constituents (LEFT-headed, BINARY feet built from the LEFT):

   [  1        0      ][  1        0      ]
     / \      / \        / \      / \
    Y   *    G   *      M   *    T   *
       / \      / \        / \      / \
      A   N    A   R      A        A

Word tree (strong on the LEFT):

   [  2 +      0      ][  1        0      ]
     / \      / \        / \      / \
    Y   *    G   *      M   *    T   *
       / \      / \        / \      / \
      A   N    A   R      A        A
```

Figure C.3: **Maranungku Examples**

```
Input segments: #([YI]<0> [WA]<0> [RA]<0> [NA]<0> [E]<0>)

Metrical constituents (LEFT-headed, BINARY feet built from the RIGHT):

   [  1      ][  1        0      ][  1        0      ]
     / \        / \      / \        / \      / \
    Y   *      W   *    R   *      N   *        *
       / \        / \      / \        / \      / \
      I          A        A          A        E

Word tree (strong on the RIGHT):

   [  1      ][  1        0      ][  2 +      0      ]
     / \        / \      / \        / \      / \
    Y   *      W   *    R   *      N   *        *
       / \        / \      / \        / \      / \
      I          A        A          A        E

Final output (weak feet are defooted in clash):

   [  0 -    ][  1        0      ][  2 +      0      ]
     / \        / \      / \        / \      / \
    Y   *      W   *    R   *      N   *        *
       / \        / \      / \        / \      / \
      I          A        A          A        E
```

Figure C.4: **Warao Example**

```
Input segments: #([ME]<0> [RE]<0> [PET]<0>)

Metrical constituents (LEFT-headed, BINARY feet built from the LEFT)
  (branching RIME must be strong):

   [  1        0      ][  1      ]
     / \      / \        / \
    M   *    R   *      P   *
       / \      / \        / \
      E        E          E   T

Word tree (strong on the LEFT):

   [  2 +      0      ][  1      ]
     / \      / \        / \
    M   *    R   *      P   *
       / \      / \        / \
      E        E          E   T


Input segments: #([YAN]<0> [GAR]<0> [MA]<0> [TA]<0>)

Metrical constituents (LEFT-headed, BINARY feet built from the LEFT)
  (branching RIME must be strong):

   [  1      ][  1        0      ][  1      ]
     / \        / \      / \        / \
    Y   *      G   *    M   *      T   *
       / \        / \      / \        / \
      A   N      A   R    A          A

Word tree (strong on the LEFT):

   [  2 +    ][  1        0      ][  1      ]
     / \        / \      / \        / \
    Y   *      G   *    M   *      T   *
       / \        / \      / \        / \
      A   N      A   R    A          A
```

Figure C.5: **Quantity-Sensitive Examples (“Maranungku-QS”)**

```
Input segments: #([E]<0> [GA]<0> [LI]<0> [TE]<0>)

Metrical constituents (RIGHT-headed, UNBOUNDED feet built from the LEFT):

   [  0        0        0        1      ]
     / \      / \      / \      / \
        *    G   *    L   *    T   *
       / \      / \      / \      / \
      E        A        I        E

Word tree (strong on the LEFT):

   [  0        0        0        2 +    ]
     / \      / \      / \      / \
        *    G   *    L   *    T   *
       / \      / \      / \      / \
      E        A        I        E
```

Figure C.6: **French Example**

```
Input segments: #([TE]<0> [LE]<0> [ZEN]<0>)

Metrical constituents (LEFT-headed, UNBOUNDED feet built from the LEFT)
  (branching NUCLEUS must be strong):

   [  1        0        0      ]
     / \      / \      / \
    T   *    L   *    Z   *
       / \      / \      / \
      E        E        E   N

Word tree (strong on the RIGHT)

   [  2 +      0        0      ]
     / \      / \      / \
    T   *    L   *    Z   *
       / \      / \      / \
      E        E        E   N

Input segments: #([SLAA]<0> [PAA]<0> [ZEM]<0>)

Metrical constituents (LEFT-headed, UNBOUNDED feet built from the LEFT)
  (branching NUCLEUS must be strong):

   [  1        1        0      ]
     / \      / \      / \
    SL  *    P   *    Z   *
       / \      / \      / \
      AA       AA       E   M

Word tree (strong on the RIGHT)

   [  1        2 +      0      ]
     / \      / \      / \
    SL  *    P   *    Z   *
       / \      / \      / \
      AA       AA       E   M
```

Figure C.7: **Eastern Cheremis Examples**

```
Input segments: #([KA]<0> [SA?]<0>)

Metrical constituents (UNBOUNDED feet built from the LEFT)
  (branching NUCLEUS must be strong):

   [  0        1      ]
     / \      / \
    K   *    S   *
       / \      / \
      A        A   ?

Word tree (strong on the RIGHT)

   [  0        2 +    ]
     / \      / \
    K   *    S   *
       / \      / \
      A        A   ?

Input segments: #([HAA]<0> [LU?]<0>)

Metrical constituents (UNBOUNDED feet built from the LEFT)
  (branching NUCLEUS must be strong):

   [  1        0      ]
     / \      / \
    H   *    L   *
       / \      / \
      AA       U   ?

Word tree (strong on the RIGHT)

   [  2 +      0      ]
     / \      / \
    H   *    L   *
       / \      / \
      AA       U   ?
```

Figure C.8: **Aguatec Mayan Examples**

# Appendix D

# Stress Learner Test Results

| No. | Hypothesis | Target Activation | Cycles | Expansion |
|---|---|---|---|---|
| 1. | #*00000000000 | 0 | 31 | 0 |
| 2. | #*00000000010 | 0 | 31 | 0 |
| 3. | #*00000001000 | 9 | 39 | 1 |
| 4. | #*00010000000 | 9 | 39 | 1 |
| 5. | #*10000000000 | 11 | 41 | 1 |
| 6. | #*00100000000 | 21 | 53 | 1 |
| 7. | #*00001000000 | 13 | 43 | 1 |
| 8. | #*00000000001 | 10 | 40 | 1 |
| 9. | #*00000001010 | 9 | 40 | 1 |
| 10. | #*00010000010 | 9 | 40 | 1 |
| 11. | #*10000000010 | 11 | 41 | 1 |
| 12. | #*00100000010 | 21 | 53 | 1 |
| 13. | #*00001000010 | 53 | 83 | 2 |
| 14. | #*00010001000 | 34 | 65 | 3 |
| 15. | #*10000001000 | 33 | 64 | 2 |
| 16. | #*00100001000 | 52 | 99 | 2 |
| 17. | #*00001001000 | 129 | 159 | 8 |
| 18. | #*00001100000 | 100 | 130 | 8 |
| 19. | #*10010000000 | 126 | 157 | 11 |
| 20. | #*00110000000 | 144 | 174 | 8 |

| | | | |
|---|---|---|---|
| 21. #*10100000000 | 59 | 90 | 5 |
| 22. #*00011000000 | 9 | 39 | 1 |
| 23. #*01001000000 | 62 | 93 | 5 |
| 24. #*10001000000 | 37 | 67 | 4 |
| 25. #*00101000000 | 23 | 54 | 1 |
| 26. #*00000001100 | 663 | 706 | 43 |
| 27. #*00000000011 | 11 | 41 | 1 |
| 28. #*00000001001 | 23 | 53 | 2 |
| 29. #*00010000001 | 25 | 56 | 2 |
| 30. #*10000000001 | 149 | 180 | 9 |
| 31. #*00100000001 | 11 | 41 | 1 |
| 32. #*00001000001 | 10 | 40 | 1 |
| 33. #*00010001010 | 77 | 124 | 4 |
| 34. #*10000001010 | 47 | 80 | 3 |
| 35. #*00100001010 | 54 | 101 | 2 |
| 36. #*00001001010 | 133 | 164 | 5 |
| 37. #*00001100010 | 96 | 126 | 6 |
| 38. #*10010000010 | 124 | 155 | 10 |
| 39. #*00110000010 | 155 | 187 | 9 |
| 40. #*10100000010 | 102 | 135 | 4 |
| 41. #*00011000010 | 9 | 40 | 1 |
| 42. #*01001000010 | 47 | 78 | 4 |
| 43. #*10001000010 | 68 | 98 | 5 |
| 44. #*00101000010 | 20 | 51 | 1 |
| 45. #*00000001110 | 633 | 675 | 42 |
| 46. #*00001101000 | 193 | 224 | 11 |
| 47. #*10010001000 | 44 | 76 | 4 |
| 48. #*00110001000 | 92 | 140 | 3 |
| 49. #*10100001000 | 73 | 120 | 4 |
| 50. #*00011001000 | 34 | 65 | 3 |
| 51. #*01001001000 | 111 | 146 | 8 |
| 52. #*10001001000 | 88 | 118 | 6 |
| 53. #*00101001000 | 54 | 99 | 2 |
| 54. #*00011100000 | 9 | 39 | 1 |

|  |  |  |  |
|---|---|---|---|
| 55. #*01001100000 | 212 | 246 | 19 |
| 56. #*10001100000 | 156 | 186 | 15 |
| 57. #*00101100000 | 20 | 51 | 1 |
| 58. #*10110000000 | 137 | 168 | 8 |
| 59. #*01011000000 | 54 | 101 | 5 |
| 60. #*10011000000 | 109 | 140 | 9 |
| 61. #*00111000000 | 127 | 157 | 11 |
| 62. #*11001000000 | 134 | 167 | 12 |
| 63. #*10101000000 | 33 | 63 | 2 |
| 64. #*01001010000 | 327 | 392 | 8 |
| 65. #*00010001100 | 451 | 493 | 19 |
| 66. #*10000001100 | 691 | 734 | 48 |
| 67. #*00100001100 | 408 | 445 | 35 |
| 68. #*00001001100 | 464 | 494 | 36 |
| 69. #*00000001011 | 24 | 54 | 2 |
| 70. #*00010000011 | 24 | 54 | 2 |
| 71. #*10000000011 | 123 | 154 | 8 |
| 72. #*00100000011 | 11 | 41 | 1 |
| 73. #*00001000011 | 10 | 40 | 1 |
| 74. #*00010001001 | 58 | 89 | 5 |
| 75. #*10000001001 | 95 | 132 | 6 |
| 76. #*00100001001 | 23 | 53 | 2 |
| 77. #*00001001001 | 22 | 53 | 2 |
| 78. #*00001100001 | 11 | 41 | 1 |
| 79. #*10010000001 | 68 | 103 | 6 |
| 80. #*00110000001 | 157 | 190 | 7 |
| 81. #*10100000001 | 76 | 142 | 6 |
| 82. #*00011000001 | 23 | 53 | 2 |
| 83. #*01001000001 | 320 | 368 | 9 |
| 84. #*10001000001 | 217 | 349 | 9 |
| 85. #*00101000001 | 11 | 41 | 1 |
| 86. #*00000001101 | 11 | 41 | 1 |
| 87. #*00001101010 | 168 | 198 | 7 |
| 88. #*10010001010 | 52 | 85 | 4 |

| | | | | |
|---|---|---|---|---|
| 89. | #*00110001010 | 93 | 144 | 3 |
| 90. | #*10100001010 | 121 | 159 | 5 |
| 91. | #*00011001010 | 75 | 123 | 4 |
| 92. | #*01001001010 | 110 | 147 | 8 |
| 93. | #*10001001010 | 106 | 136 | 6 |
| 94. | #*00101001010 | 51 | 100 | 2 |
| 95. | #*00011100010 | 9 | 40 | 1 |
| 96. | #*01001100010 | 255 | 291 | 23 |
| 97. | #*10001100010 | 160 | 190 | 14 |
| 98. | #*00101100010 | 20 | 51 | 1 |
| 99. | #*10110000010 | 123 | 153 | 7 |
| 100. | #*01011000010 | 53 | 101 | 5 |
| 101. | #*10011000010 | 117 | 148 | 10 |
| 102. | #*00111000010 | 151 | 181 | 12 |
| 103. | #*11001000010 | 136 | 173 | 12 |
| 104. | #*10101000010 | 96 | 132 | 4 |
| 105. | #*01001010010 | 390 | 451 | 9 |
| 106. | #*00010001110 | 389 | 429 | 23 |
| 107. | #*10000001110 | 610 | 656 | 42 |
| 108. | #*00100001110 | 445 | 480 | 38 |
| 109. | #*00001001110 | 595 | 628 | 44 |
| 110. | #*00011101000 | 32 | 64 | 3 |
| 111. | #*01001101000 | 242 | 276 | 18 |
| 112. | #*10001101000 | 174 | 204 | 11 |
| 113. | #*00101101000 | 49 | 93 | 2 |
| 114. | #*10110001000 | 111 | 158 | 5 |
| 115. | #*01011001000 | 97 | 139 | 7 |
| 116. | #*10011001000 | 48 | 80 | 4 |
| 117. | #*00111001000 | 109 | 139 | 7 |
| 118. | #*11001001000 | 151 | 185 | 11 |
| 119. | #*10101001000 | 82 | 128 | 4 |
| 120. | #*01001011000 | 256 | 381 | 7 |
| 121. | #*01011100000 | 240 | 281 | 22 |
| 122. | #*10011100000 | 126 | 157 | 11 |

| | | | | |
|---|---|---|---|---|
| 123. | #*00111100000 | 179 | 209 | 17 |
| 124. | #*11001100000 | 255 | 289 | 23 |
| 125. | #*10101100000 | 37 | 68 | 3 |
| 126. | #*11011000000 | 140 | 181 | 13 |
| 127. | #*10111000000 | 155 | 185 | 12 |
| 128. | #*01001110000 | 444 | 517 | 12 |
| 129. | #*11001010000 | 301 | 385 | 9 |
| 130. | #*00001101100 | 498 | 528 | 39 |
| 131. | #*10010001100 | 441 | 475 | 21 |
| 132. | #*00110001100 | 366 | 415 | 15 |
| 133. | #*10100001100 | 435 | 476 | 36 |
| 134. | #*00011001100 | 508 | 547 | 23 |
| 135. | #*01001001100 | 423 | 456 | 30 |
| 136. | #*10001001100 | 461 | 492 | 38 |
| 137. | #*00101001100 | 378 | 414 | 32 |
| 138. | #*00010001011 | 50 | 85 | 4 |
| 139. | #*10000001011 | 112 | 148 | 6 |
| 140. | #*00100001011 | 22 | 52 | 2 |
| 141. | #*00001001011 | 23 | 54 | 2 |
| 142. | #*00001100011 | 11 | 41 | 1 |
| 143. | #*10010000011 | 69 | 103 | 6 |
| 144. | #*00110000011 | 182 | 214 | 8 |
| 145. | #*10100000011 | 90 | 145 | 6 |
| 146. | #*00011000011 | 22 | 52 | 2 |
| 147. | #*01001000011 | 302 | 351 | 8 |
| 148. | #*10001000011 | 273 | 403 | 10 |
| 149. | #*00101000011 | 11 | 41 | 1 |
| 150. | #*00000001111 | 10 | 40 | 1 |
| 151. | #*00001101001 | 22 | 52 | 2 |
| 152. | #*10010001001 | 76 | 113 | 6 |
| 153. | #*00110001001 | 135 | 179 | 6 |
| 154. | #*10100001001 | 106 | 160 | 6 |
| 155. | #*00011001001 | 55 | 88 | 4 |
| 156. | #*01001001001 | 261 | 366 | 8 |

| | | | |
|---|---|---|---|
| 157. #*10001001001 | 227 | 342 | 9 |
| 158. #*00101001001 | 23 | 53 | 2 |
| 159. #*00011100001 | 23 | 53 | 2 |
| 160. #*01001100001 | 410 | 471 | 11 |
| 161. #*10001100001 | 401 | 558 | 15 |
| 162. #*00101100001 | 11 | 41 | 1 |
| 163. #*10110000001 | 75 | 111 | 6 |
| 164. #*01011000001 | 348 | 458 | 11 |
| 165. #*10011000001 | 74 | 111 | 6 |
| 166. #*00111000001 | 648 | 766 | 10 |
| 167. #*11001000001 | 189 | 358 | 7 |
| 168. #*10101000001 | 69 | 125 | 5 |
| 169. #*01001010001 | 298 | 355 | 8 |
| 170. #*00010001101 | 70 | 228 | 4 |
| 171. #*10000001101 | 487 | 535 | 24 |
| 172. #*00100001101 | 11 | 41 | 1 |
| 173. #*00001001101 | 11 | 41 | 1 |
| 174. #*00011101010 | 77 | 124 | 4 |
| 175. #*01001101010 | 236 | 275 | 17 |
| 176. #*10001101010 | 217 | 247 | 15 |
| 177. #*00101101010 | 54 | 101 | 2 |
| 178. #*10110001010 | 111 | 158 | 5 |
| 179. #*01011001010 | 120 | 160 | 9 |
| 180. #*10011001010 | 46 | 79 | 4 |
| 181. #*00111001010 | 157 | 189 | 8 |
| 182. #*11001001010 | 159 | 194 | 11 |
| 183. #*10101001010 | 131 | 169 | 5 |
| 184. #*01001011010 | 247 | 344 | 7 |
| 185. #*01011100010 | 257 | 300 | 24 |
| 186. #*10011100010 | 128 | 158 | 10 |
| 187. #*00111100010 | 222 | 252 | 18 |
| 188. #*11001100010 | 242 | 279 | 22 |
| 189. #*10101100010 | 101 | 137 | 4 |
| 190. #*11011000010 | 145 | 190 | 13 |

| | | | |
|---|---|---|---|
| 191. #*10111000010 | 114 | 144 | 7 |
| 192. #*01001110010 | 433 | 496 | 13 |
| 193. #*11001010010 | 264 | 372 | 9 |
| 194. #*00001101110 | 540 | 572 | 39 |
| 195. #*10010001110 | 446 | 484 | 20 |
| 196. #*00110001110 | 408 | 454 | 18 |
| 197. #*10100001110 | 331 | 368 | 23 |
| 198. #*00011001110 | 410 | 449 | 24 |
| 199. #*01001001110 | 392 | 427 | 28 |
| 200. #*10001001110 | 263 | 294 | 19 |
| 201. #*00101001110 | 426 | 461 | 36 |
| 202. #*01011101000 | 284 | 322 | 20 |
| 203. #*10011101000 | 52 | 85 | 4 |
| 204. #*00111101000 | 137 | 167 | 9 |
| 205. #*11001101000 | 261 | 298 | 19 |
| 206. #*10101101000 | 90 | 136 | 5 |
| 207. #*11011001000 | 174 | 213 | 13 |
| 208. #*10111001000 | 174 | 220 | 9 |
| 209. #*01001111000 | 366 | 490 | 11 |
| 210. #*11001011000 | 145 | 252 | 6 |
| 211. #*11011100000 | 255 | 296 | 24 |
| 212. #*10111100000 | 149 | 179 | 12 |
| 213. #*11001110000 | 347 | 462 | 13 |
| 214. #*00011101100 | 475 | 514 | 21 |
| 215. #*01001101100 | 435 | 469 | 31 |
| 216. #*10001101100 | 481 | 511 | 40 |
| 217. #*00101101100 | 428 | 467 | 37 |
| 218. #*10110001100 | 92 | 125 | 4 |
| 219. #*01011001100 | 321 | 364 | 23 |
| 220. #*10011001100 | 422 | 461 | 19 |
| 221. #*00111001100 | 311 | 341 | 22 |
| 222. #*11001001100 | 374 | 407 | 27 |
| 223. #*10101001100 | 459 | 500 | 38 |
| 224. #*01001011100 | 339 | 427 | 7 |

| | | | | |
|---|---|---|---|---|
| 225. | #*00001101011 | 22 | 52 | 2 |
| 226. | #*10010001011 | 77 | 115 | 6 |
| 227. | #*00110001011 | 105 | 152 | 5 |
| 228. | #*10100001011 | 104 | 157 | 6 |
| 229. | #*00011001011 | 52 | 85 | 4 |
| 230. | #*01001001011 | 294 | 404 | 8 |
| 231. | #*10001001011 | 234 | 381 | 9 |
| 232. | #*00101001011 | 23 | 54 | 2 |
| 233. | #*00011100011 | 22 | 53 | 2 |
| 234. | #*01001100011 | 488 | 540 | 13 |
| 235. | #*10001100011 | 445 | 592 | 15 |
| 236. | #*00101100011 | 11 | 41 | 1 |
| 237. | #*10110000011 | 62 | 95 | 5 |
| 238. | #*01011000011 | 301 | 409 | 10 |
| 239. | #*10011000011 | 77 | 115 | 6 |
| 240. | #*00111000011 | 415 | 532 | 7 |
| 241. | #*11001000011 | 167 | 358 | 8 |
| 242. | #*10101000011 | 70 | 144 | 5 |
| 243. | #*01001010011 | 274 | 318 | 8 |
| 244. | #*00010001111 | 79 | 253 | 5 |
| 245. | #*10000001111 | 491 | 534 | 24 |
| 246. | #*00100001111 | 11 | 41 | 1 |
| 247. | #*00001001111 | 11 | 41 | 1 |
| 248. | #*00011101001 | 60 | 95 | 5 |
| 249. | #*01001101001 | 392 | 495 | 10 |
| 250. | #*10001101001 | 568 | 716 | 15 |
| 251. | #*00101101001 | 22 | 52 | 2 |
| 252. | #*10110001001 | 78 | 114 | 6 |
| 253. | #*01011001001 | 175 | 281 | 7 |
| 254. | #*10011001001 | 67 | 101 | 6 |
| 255. | #*00111001001 | 315 | 458 | 7 |
| 256. | #*11001001001 | 156 | 299 | 8 |
| 257. | #*10101001001 | 116 | 174 | 7 |
| 258. | #*01001011001 | 250 | 343 | 7 |

| | | | |
|---|---|---|---|
| 259. #*01011100001 | 355 | 467 | 12 |
| 260. #*10011100001 | 72 | 105 | 6 |
| 261. #*00111100001 | 637 | 762 | 11 |
| 262. #*11001100001 | 289 | 503 | 14 |
| 263. #*10101100001 | 68 | 136 | 5 |
| 264. #*11011000001 | 194 | 355 | 8 |
| 265. #*10111000001 | 71 | 107 | 6 |
| 266. #*01001110001 | 441 | 493 | 12 |
| 267. #*11001010001 | 239 | 347 | 8 |
| 268. #*00001101101 | 11 | 41 | 1 |
| 269. #*10010001101 | 66 | 129 | 5 |
| 270. #*00110001101 | 372 | 416 | 14 |
| 271. #*10100001101 | 274 | 313 | 19 |
| 272. #*00011001101 | 70 | 222 | 4 |
| 273. #*01001001101 | 320 | 408 | 8 |
| 274. #*10001001101 | 510 | 622 | 14 |
| 275. #*00101001101 | 11 | 41 | 1 |
| 276. #*01011101010 | 248 | 285 | 18 |
| 277. #*10011101010 | 52 | 85 | 4 |
| 278. #*00111101010 | 262 | 293 | 14 |
| 279. #*11001101010 | 258 | 292 | 19 |
| 280. #*10101101010 | 121 | 159 | 5 |
| 281. #*11011001010 | 150 | 195 | 11 |
| 282. #*10111001010 | 190 | 231 | 7 |
| 283. #*01001111010 | 361 | 468 | 10 |
| 284. #*11001011010 | 175 | 276 | 7 |
| 285. #*11011100010 | 243 | 286 | 22 |
| 286. #*10111100010 | 156 | 186 | 10 |
| 287. #*11001110010 | 355 | 469 | 13 |
| 288. #*00011101110 | 388 | 428 | 23 |
| 289. #*01001101110 | 447 | 480 | 32 |
| 290. #*10001101110 | 306 | 336 | 22 |
| 291. #*00101101110 | 445 | 480 | 38 |
| 292. #*10110001110 | 96 | 132 | 4 |

|  |  |  |  |
|---|---|---|---|
| 293. #*01011001110 | 339 | 382 | 24 |
| 294. #*10011001110 | 472 | 509 | 22 |
| 295. #*00111001110 | 300 | 332 | 20 |
| 296. #*11001001110 | 374 | 408 | 27 |
| 297. #*10101001110 | 320 | 357 | 23 |
| 298. #*01001011110 | 364 | 450 | 8 |
| 299. #*11011101000 | 250 | 288 | 18 |
| 300. #*10111101000 | 257 | 303 | 13 |
| 301. #*11001111000 | 235 | 347 | 10 |
| 302. #*01011101100 | 386 | 427 | 28 |
| 303. #*10011101100 | 445 | 478 | 21 |
| 304. #*00111101100 | 320 | 350 | 23 |
| 305. #*11001101100 | 411 | 446 | 30 |
| 306. #*10101101100 | 447 | 489 | 37 |
| 307. #*11011001100 | 316 | 359 | 24 |
| 308. #*10111001100 | 396 | 432 | 24 |
| 309. #*01001111100 | 486 | 569 | 11 |
| 310. #*11001011100 | 216 | 310 | 8 |
| 311. #*00011101011 | 53 | 85 | 4 |
| 312. #*01001101011 | 367 | 469 | 10 |
| 313. #*10001101011 | 531 | 656 | 14 |
| 314. #*00101101011 | 22 | 52 | 2 |
| 315. #*10110001011 | 75 | 111 | 6 |
| 316. #*01011001011 | 157 | 279 | 6 |
| 317. #*10011001011 | 75 | 111 | 6 |
| 318. #*00111001011 | 331 | 479 | 7 |
| 319. #*11001001011 | 143 | 282 | 7 |
| 320. #*10101001011 | 108 | 154 | 7 |
| 321. #*01001011011 | 264 | 348 | 7 |
| 322. #*01011100011 | 416 | 522 | 14 |
| 323. #*10011100011 | 70 | 104 | 6 |
| 324. #*00111100011 | 617 | 730 | 11 |
| 325. #*11001100011 | 294 | 493 | 14 |
| 326. #*10101100011 | 80 | 148 | 5 |

| | | | |
|---|---|---|---|
| 327. #*11011000011 | 208 | 359 | 9 |
| 328. #*10111000011 | 79 | 113 | 6 |
| 329. #*01001110011 | 448 | 501 | 12 |
| 330. #*11001010011 | 265 | 373 | 9 |
| 331. #*00001101111 | 11 | 41 | 1 |
| 332. #*10010001111 | 86 | 138 | 6 |
| 333. #*00110001111 | 384 | 429 | 14 |
| 334. #*10100001111 | 282 | 320 | 20 |
| 335. #*00011001111 | 86 | 228 | 4 |
| 336. #*01001001111 | 352 | 438 | 8 |
| 337. #*10001001111 | 529 | 627 | 15 |
| 338. #*00101001111 | 10 | 40 | 1 |
| 339. #*01011101001 | 246 | 349 | 10 |
| 340. #*10011101001 | 73 | 106 | 6 |
| 341. #*00111101001 | 476 | 583 | 10 |
| 342. #*11001101001 | 321 | 471 | 13 |
| 343. #*10101101001 | 108 | 156 | 7 |
| 344. #*11011001001 | 303 | 485 | 10 |
| 345. #*10111001001 | 75 | 113 | 6 |
| 346. #*01001111001 | 382 | 499 | 11 |
| 347. #*11001011001 | 160 | 261 | 7 |
| 348. #*11011100001 | 331 | 480 | 14 |
| 349. #*10111100001 | 68 | 101 | 6 |
| 350. #*11001110001 | 345 | 467 | 12 |
| 351. #*00011101101 | 62 | 223 | 4 |
| 352. #*01001101101 | 519 | 593 | 12 |
| 353. #*10001101101 | 590 | 698 | 16 |
| 354. #*00101101101 | 10 | 40 | 1 |
| 355. #*10110001101 | 78 | 146 | 6 |
| 356. #*01011001101 | 266 | 368 | 9 |
| 357. #*10011001101 | 82 | 145 | 6 |
| 358. #*00111001101 | 693 | 843 | 10 |
| 359. #*11001001101 | 341 | 474 | 16 |
| 360. #*10101001101 | 284 | 322 | 20 |

| | | | | |
|---|---|---|---|---|
| 361. | #*01001011101 | 339 | 416 | 8 |
| 362. | #*11011101010 | 272 | 310 | 20 |
| 363. | #*10111101010 | 229 | 269 | 10 |
| 364. | #*11001111010 | 255 | 358 | 10 |
| 365. | #*01011101110 | 407 | 447 | 30 |
| 366. | #*10011101110 | 425 | 462 | 20 |
| 367. | #*00111101110 | 319 | 350 | 21 |
| 368. | #*11001101110 | 460 | 495 | 34 |
| 369. | #*10101101110 | 334 | 376 | 24 |
| 370. | #*11011001110 | 319 | 363 | 24 |
| 371. | #*10111001110 | 334 | 384 | 17 |
| 372. | #*01001111110 | 494 | 580 | 12 |
| 373. | #*11001011110 | 221 | 328 | 8 |
| 374. | #*11011101100 | 365 | 408 | 27 |
| 375. | #*10111101100 | 382 | 418 | 24 |
| 376. | #*11001111100 | 340 | 451 | 12 |
| 377. | #*01011101011 | 266 | 353 | 11 |
| 378. | #*10011101011 | 71 | 106 | 6 |
| 379. | #*00111101011 | 493 | 623 | 10 |
| 380. | #*11001101011 | 255 | 391 | 11 |
| 381. | #*10101101011 | 123 | 185 | 7 |
| 382. | #*11011001011 | 264 | 441 | 9 |
| 383. | #*10111001011 | 82 | 118 | 6 |
| 384. | #*01001111011 | 399 | 520 | 11 |
| 385. | #*11001011011 | 165 | 275 | 7 |
| 386. | #*11011100011 | 361 | 497 | 15 |
| 387. | #*10111100011 | 67 | 101 | 5 |
| 388. | #*11001110011 | 363 | 463 | 12 |
| 389. | #*00011101111 | 74 | 230 | 5 |
| 390. | #*01001101111 | 490 | 568 | 11 |
| 391. | #*10001101111 | 616 | 717 | 16 |
| 392. | #*00101101111 | 10 | 40 | 1 |
| 393. | #*10110001111 | 85 | 137 | 6 |
| 394. | #*01011001111 | 205 | 308 | 7 |

| | | | | |
|---|---|---|---|---|
| 395. | #*10011001111 | 88 | 135 | 6 |
| 396. | #*00111001111 | 635 | 807 | 10 |
| 397. | #*11001001111 | 322 | 446 | 15 |
| 398. | #*10101001111 | 272 | 309 | 19 |
| 399. | #*01001011111 | 345 | 409 | 8 |
| 400. | #*11011101001 | 493 | 650 | 15 |
| 401. | #*10111101001 | 81 | 119 | 6 |
| 402. | #*11001111001 | 272 | 388 | 10 |
| 403. | #*01011101101 | 336 | 421 | 12 |
| 404. | #*10011101101 | 77 | 143 | 5 |
| 405. | #*00111101101 | 707 | 851 | 12 |
| 406. | #*11001101101 | 407 | 524 | 19 |
| 407. | #*10101101101 | 266 | 305 | 18 |
| 408. | #*11011001101 | 416 | 568 | 15 |
| 409. | #*10111001101 | 71 | 128 | 5 |
| 410. | #*01001111101 | 488 | 575 | 11 |
| 411. | #*11001011101 | 256 | 349 | 9 |
| 412. | #*11011101110 | 355 | 393 | 27 |
| 413. | #*10111101110 | 384 | 424 | 20 |
| 414. | #*11001111110 | 330 | 437 | 12 |
| 415. | #*11011101011 | 494 | 682 | 15 |
| 416. | #*10111101011 | 79 | 116 | 6 |
| 417. | #*11001111011 | 258 | 350 | 11 |
| 418. | #*01011101111 | 286 | 397 | 11 |
| 419. | #*10011101111 | 66 | 134 | 5 |
| 420. | #*00111101111 | 728 | 870 | 12 |
| 421. | #*11001101111 | 393 | 518 | 18 |
| 422. | #*10101101111 | 294 | 334 | 20 |
| 423. | #*11011001111 | 430 | 592 | 15 |
| 424. | #*10111001111 | 78 | 125 | 6 |
| 425. | #*01001111111 | 566 | 667 | 12 |
| 426. | #*11001011111 | 217 | 321 | 8 |
| 427. | #*11011101101 | 532 | 690 | 18 |
| 428. | #*10111101101 | 87 | 136 | 6 |

| | | | |
|---|---|---|---|
| 429. #*11001111101 | 372 | 474 | 13 |
| 430. #*11011101111 | 538 | 719 | 18 |
| 431. #*10111101111 | 82 | 153 | 5 |
| 432. #*11001111111 | 305 | 407 | 10 |