MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Memo No. 1575 March, 1996

A Computational Model for the Acquisition and Use of Phonological Knowledge

Kenneth Yip **Gerald Jay Sussman**

This publication can be retrieved by anonymous ftp to publications.ai.mit.edu.

Abstract

Does knowledge of language consist of symbolic rules? How do children learn and use their linguistic knowledge? To elucidate these questions, we present a computational model that acquires phonological knowledge from a corpus of common English nouns and verbs. In our model the phonological knowledge is encapsulated as boolean constraints operating on classical linguistic representations of speech sounds in term of distinctive features. The learning algorithm compiles a corpus of words into increasingly sophisticated constraints. The algorithm is incremental, greedy, and fast. It yields one-shot learning of phonological constraints from a few examples. Our system exhibits behavior similar to that of young children learning phonological knowledge. As a bonus the constraints can be interpreted as classical linguistic rules. The computational model can be implemented by a surprisingly simple hardware mechanism. Our mechanism also sheds light on a fundamental AI question: How are signals related to symbols?

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for this research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-92-J-4097, by the National Science Foundation under grant number MIP-9001651, and by an National Science Foundation NYI Award ECS-9357773.

1 Introduction

Almost every child learns how to speak and to understand his native language. It must be easy. However, we do not have effective theories that explain the phenomena. In this paper we attempt to illuminate a small dark corner of the problem: the acquisition of phonological knowledge.

Children learn the vocabulary of their native language with amazing speed, once they get to the right stage of development. Children typically learn many new words, and their correct usage, each day. They do not need to hear the same words repeated over and over again. They do not need to be corrected very often.

The mystery deepens when we notice that children learn many new words without ever hearing them. In a classic experiment by Berko [1], a number of English-speaking children were shown representations of a fanciful being called a "wug." When asked to say something about a situation with more than one of these beings, the children correctly pluralized the novel word to make "wugz" (not "wugs"). In another experiment [4], Ervin showed that young children who first use an irregular verb properly (such as "came") would later err on the same verb (such as "comed") before they use the verb correctly again. In this way children reliably exhibit behavior that indicates that they have made generalizations that linguists describe with rules.

We will present a simple mechanism that makes and uses similar generalizations. The generalizations are derived from a small corpus of common English words. The behavior of our mechanism exhibits many aspects of the behavior of children. It needs only a few, carelessly chosen examples. It requires no repetition. In intermediate stages of development it makes the same kinds of errors that children make. And, when we pop off the cover and look inside, we find that the internal representations constructed by this mechanism can be read out as the rules that are found in classical books of linguistics.

For example, after seeing a dozen common nouns and their plurals, our system learns three pluralization rules: (1) Nouns ending in one of the "hissing" sounds ([s], [z], [sh], [ch], [zh] and [j]) are pluralized by adding an additional syllable [I.z] to the root word, (2) Nouns ending in a voiced phoneme (other than the hissing sounds) are pluralized by adding a [z] sound, and (3) Nouns ending in a voiceless consonant (other than the hissing sounds) are pluralized by adding a [s] sound.

We do not attack the problem of how an acoustic waveform is processed. We start with an abstraction from linguistics (as developed by Roman Jakobson, Nikolai Trubetzkoy, Morris Halle, and Noam Chomsky) [2]: Speech sounds (phonemes) are not atomic but are encoded as combinations of more primitive structures, called *distinctive features*, that control the configuration of the major speech organs (such as tongue, lips, and vocal cords). The representation of linguistic sounds as sequences of phonemes that are made up of distinctive features, and the arguments and experiments establishing the psycholinguistic reality of such a representation, is one of the great achievements of linguistics.

In our theory the fundamental representation of all of the information about a linguistic event is in the form of bit vectors. Part of the information is the sequence of phonemes, which are themselves represented as bit vectors of the values of the distinctive features. Part of the information is grammatical, such as whether or not we have a noun, and if it is a noun whether or not it is plural. There is also a set of bits intended to identify the meaning. These parts do not have a sharp boundary: The grammatical status of a word surely overlaps with its meaning. In any linguistic event some of the bits in the bit vectors may be known, and others may be unknown.

Our mechanism consists of a performance model and an acquisition procedure. The job of the performance model is to fill in the details of a linguistic event, and enforce phonological constraints. The constraints can run in any direction; there are no distinguished input or output bits. The implementation substrate is simple: a few registers and bit vectors encoding both data and control operations (like shift and lock). The registers can be written by an external linguistic event or by constraints. The registers are continuously monitored for changes. Special correlations among registers are recorded and accumulated for later summarization. Conflicts in the assignment of bit values might trigger additional constraints that attempt to resolve the conflicts. When a conflict cannot be resolved, the acquisition procedure kicks in and tries to modify existing constraints or construct new ones.

The goal of the acquisition procedure is to compile a corpus of English words into a series of increasingly sophisticated phonological constraints. Our theory of acquisition differs significantly from those based on statistics (such as [12, 6]). The acquisition process is rather simple. It is incremental, greedy, and fast. It has almost no parameters to adjust. It makes falsifiable predictions about the learning of phonological constraints: (1) That learning requires very few examples, (2) That the same target constraints are learned independent of the presentation order of the corpus, (3) That learning is insensitive to token frequency, and (4) That learning is more effective as more constraints are acquired.

This paper is structured as follows. We first present a simple hardware mechanism that is sufficient to implement our performance model. After a brief section on linguistic background, we illustrate the operations of the performance model. We then explain the acquisition procedure, and present experimental results on learning pluralization and past tense inflection rules. Finally, we discuss the implications of this work in the context of understanding the general problem of signal-to-symbol transformation.

2 Hardware Vision

Our theory is expressed in terms of a mechanism that implements the performance model. By contrast to unrestricted computer programs, mechanisms are limited in the kinds of parts that we may postulate and in the ways that they may be interconnected. By restricting ourselves to a simple mechanism we construct a stronger and more robust theory.

Our mechanism uses a few registers. We imagine that the sequence of phonemes is arranged in the phoneme register. There are also registers that hold bits describing grammatical features and other components of meaning.

The phoneme register is a shift register: the slots in the phoneme register reflect the temporal sequence of the phonemes.[1] See Figure 1. The bus that brings phonemes in when they are heard and takes phonemes out to the muscle controllers when speaking terminates in the "current phoneme" slot of the phoneme register. The register shifts when a new phoneme is heard or is spoken. There are slots in the register for both future and past phonemes. Also, certain segments of the register may shift while others remain fixed. This is useful when the linguistic data require the expansion of time by duplication of a phoneme or the compression of time by deletion of a phoneme. For example, a phonological constraint might specify that a long vowel is to be shortened in specific phonological contexts.

The distinctive feature bits of the phonemes in the shift register are constrained by boolean relations. Propagation of values through these constraints is the mechanism by which unknown details are filled in, when determined by other known information. Note that there may be constraints among distinctive feature bits in phonemes in different time slots. These constraints may involve grammar and meaning bits, and they also involve control bits, such as those that enable segments of the phoneme register to shift.

For example, if "wugz" has just been heard, the phoneme representation of the sounds is assumed to be in time slots -3 through 0. If there is an expectation of hearing a noun at this time, the noun bit in the grammar register will be on. This data—the terminal "z" and the noun bit turned on—will propagate through some boolean constraints implementing the phonological generalizations (details are in section 4) to force the plural bit to turn on, if it is unknown. The same constraints

[1]Our shift register model of phoneme is only a crude approximation to what physically takes place in speech. We make two idealizations. First, the distinctive features are discretized to be either 0 or 1. As we shall see in section 3, the distinctive features are really analog signals controlling the articulatory gestures of speech organs. Second, the distinctive features are assumed to change synchronously. In hearing or speaking, the analog signals overlap in time and their durations need not be aligned perfectly.

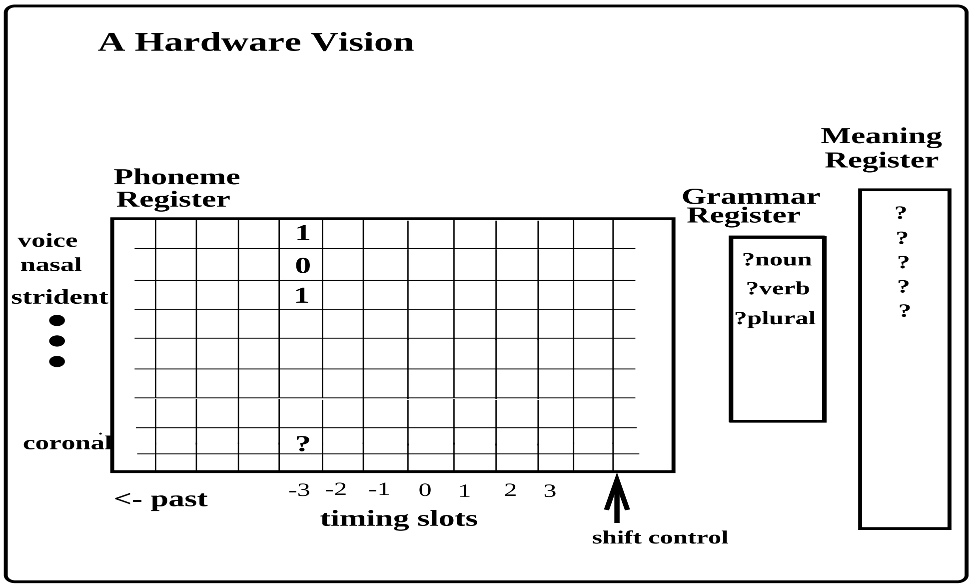

Figure 1: The hardware mechanism consists of three data registers. In the Phoneme Shift Register, each vertical stripe represents a time slot. There are slots for future phonemes (positive time labels) as well as past phonemes (negative time labels). Each horizontal stripe represents a distinctive feature bit. For example, if the phoneme in time slot -3 is known to be a voiced non-nasal then in the column labeled -3 the voice entry would be 1 and the nasal entry would be 0. If the phoneme is known to be strident, the strident entry would be 1. If a feature is unknown, as in the coronal bit, we leave a question mark. The Grammar Register contains bits describing the grammatical status of the phoneme sequence. The Meaning Register contains meaning bits.

may be used in a different direction. If "wug" is heard, and we expected a plural noun, then the same constraints will force the unknown terminal phoneme to be "z."

In our theory, the mechanism implementing phonological knowledge is boolean constraints among the various bits in the phoneme shift register and the associated grammar, meaning, and control bits. Acquisition of such knowledge is accomplished by creating and incrementally modifying these constraints. These modifications are made as part of a process that summarizes correlations discovered in the corpus.

3 Linguistic Background: Phonemes and distinctive features

Linguists idealize speech by breaking speech sounds into discrete time segments called *phonemes*. This decomposition allows the compact expression of certain regularities that are observed in language—regularities that would not be apparent if words were considered atomic. Similarly, the phonemes themselves are thought to be combinations of *distinctive features*.[2] Each distinctive feature is a binary-valued variable. Distinctive features represent the gestures that the speech organs – such as the tongue, lips, glottis – execute during the speaking process. By altering the size and shape of the air cavities inside the vocal tract, these gestures produce specific acoustic effects: they might amplify or attenuate certain frequencies of the speech sound. The distinctive features not only provide a means for classifying speech sounds, but also allow linguists to build elegant phonological theories that describe what speakers must know about their language.

For example, pay attention to how you pronounce the phoneme [z] as in "dogs" and the phoneme [s] as in "cats." (It is customary to use the square brackets to enclose the phonetic symbols that linguists use to notate the phonemes.) If you put a finger in each ear, you will hear a buzz and feel the vibrations of the vocal cords when you say [z]. Thus [z] is said to have the voicing feature. You can hear only a hissing sound and feel no vibrations when you say [s]. So [s] lacks voicing. The voicing feature distinguishes between sound pairs such as [d] and [t], [b] and [p], [v] and [f]. The first member of each pair is said to be [+voice], while its partner is [−voice].

Speech sounds that share common distinctive features are often grouped into natural classes. For example, when you produce [f] ("off"), [v] ("give"), [ch] ("watch"), [j] ("judge"), [sh] ("wish"), and [zh] ("garage"), you will hear a hissing or buzzing sound much like that of [s] and [z]. When you produce these sounds, a stream of air is forced through a small opening, causing air to vibrate violently and creating an acoustic turbulence. These eight sounds are said to be [+strident].

The most widely used distinctive feature system is the one described in *The Sound Pattern of English* [2]. This feature system uses 14 distinctive features. Table 1 below shows the distinctive features for a subset of English vowels and consonants. Each phoneme is a particular combination of the 14 features.

Languages arrive at different inventories of phonemes by using some subset of these 2^{14} possibilities. Actually not all of the 2^{14} combinations are possible. For example, no phoneme can be both [+high] and [+low] because the tongue position cannot be high and low at the same time. However, a phoneme can be both [-high] and [-low], meaning that the tongue position is in the middle.

It might seem that languages are tremendously wasteful in using the distinctive features. No human language uses many more than 100 phonemes. English uses 40.

[2]In recent phonological theories, the distinctive features of a speech sound are not simply an unordered bundle; they are organized in a hierarchical tree structure. The hierarchical grouping of features is used to explain observed restrictions on feature combinations in phonological processes. See [5] for a discussion of some recent feature models.

feature	[i]	[ae]	[k]	[t]	[s]	[z]
syllabic	1	1	0	0	0	0
consonantal	0	0	1	1	1	1
sonorant	1	1	0	0	0	0
high	1	0	1	0	0	0
back	0	0	1	0	0	0
low	0	1	0	0	0	0
round	0	0	0	0	0	0
tense	1	0	1	1	1	0
anterior	0	0	0	1	1	1
coronal	0	0	0	1	1	1
voice	1	1	0	0	0	1
continuant	1	1	0	0	1	1
nasal	0	0	0	0	0	0
strident	0	0	0	0	1	1

Table 1: Distinctive features for some English vowels and consonants. Phonetic symbols are enclosed in square brackets. For example, [i] is the symbol for the high [+high], front [−back], tense [+tense] vowel in "beat," and [ae] is the low [+low], front [−back], lax [−tense] vowel in "cat."

However, our learning theory will show that this sparseness of the representation is essential for one-shot learning of phonological knowledge.

Given a sequence of phonemes, we can read off its distinctive feature representation from the table. For example, the word "cats" [k ae t s] can be arranged in a matrix with time on the horizontal axis and distinctive feature on the vertical. See Figure 2.

4 Performance model

The target of learning is a performance model consisting of three data registers and a continuously running constraint propagator.

The phoneme register contains bits detailing the phonemes that describe the linguistic sound pattern of a word. The grammar register holds the grammatical status of the word, and the meaning register contains meaning features. The bits in registers have four possible states {0, 1, ?, *}. If the value of a bit is currently unknown it contains an unknown symbol (?). Such a bit can be set to a known value by a constraint that infers the value from other known bits. If a bit is asserted to be both 1 and 0 because of a disagreement among the constraints it participates in, it is in

	[k]	[ae]	[t]	[s]	“cats”
syllabic	0	1	0	0	
consonantal	1	0	1	1	
sonorant	0	1	0	0	
high	1	0	0	0	
back	1	0	0	0	
low	0	1	0	0	
round	0	0	0	0	
tense	1	0	1	1	
anterior	0	0	1	1	
coronal	0	0	1	1	
voice	0	1	0	0	
continuant	0	1	0	1	
nasal	0	0	0	0	
strident	0	0	0	1	
TIME ···	-3	-2	-1	0	1 ···

Figure 2: Phonemes are represented by vectors of distinctive features. Each vector is indexed by a time instant. The column labeled by time = 0 is the most recently heard phoneme. Phoneme with negative time indices are already heard.

the conflict state, which we denote by (*).

Boolean constraints continuously monitor the data registers. A constraint is excited when a sufficient number of its bits match those in the registers. The strength of excitation of a constraint is measured by the number of conflicting bits: the fewer the conflicting bits, the stronger the excitation. The most excited constraint takes control over the data registers, and enforces its constraint on slots that it is connected to by setting bits in slots containing “?” to definite values. The assignment of definite values to these slots may trigger other constraints, and thus the propagation may fill in many unknown values.

Conflicts arise when a constraint finds that its constraint is inconsistent with the data in the slots that it is connected to. The inconsistency may be a direct conflict with externally imposed data, or it may arise because two constraints try to require a slot to have different boolean values. Such conflicts lead to learning opportunities. Constraints are modified by either generalizing (i.e., a constraint disconnects from certain bits) or specializing (i.e., a constraint connects to additional bits), avoiding the conflict situation. We will see how this works in section 5.

4.1 Classifiers: How constraints are represented and activated

Boolean constraints are multi-directional: There are no distinguished input and output bits. Our system represents constraints and data uniformly as bit vectors.

We refer to these bit vectors as *classifiers*. A classifier is a finite string over the three-symbol alphabet $0, 1, -$, where "$-$" is the don't care symbol. A "1" (or "0") typically represents the presence (or absence) of a characteristic. A "$-$" means don't care, i.e., the bit's actual value does not matter.

It would be unintelligible to describe a classifier by its actual bit vector. Instead we will use symbolic notations to abbreviate the various components of a classifier. For example, the notation [ae.p.l] refers to the 42 bits (3×14) representing the distinctive features of the phonemes enclosed in the square brackets. The symbol "dc" for a phoneme abbreviates a string of 14 don't-care bits. The notation [+noun] indicates that the noun bit in the classifier is on.

There are two types of classifiers: *rote-classifier* and *rule-classifier*. The rote-classifier represents special constraints among the phoneme, grammar, and meaning registers. For example, a rote-classifier for "apple" enforces a certain constraint among the phoneme slots containing [ae.p.l], the [+noun,-verb,-plural...] features in the grammar register, and the bits in the meaning register. A rote-classifier for "apple" might have the following form[3]:

```
rote-classifier-apple
  Phonemes: ae.p.l
  Grammar:  [+noun -verb -plural ...]
  Meaning:  [+red +round +edible +fruit ... ]
```

The rule-classifier represents a more general constraint between the phoneme and the grammar registers. For example, suppose we already have the rote-classifiers for the words "apple" and "apples." As "apples" is heard, a temporal pattern is observed when the two rote-classifiers attempt to control the data registers (more on this in section 4.3). This pattern of change is observed and recorded. When several examples of such temporal change have been accumulated, the common pattern of change is abstracted in a rule-classifier which says roughly that to resolve the conflict in the plural bit, the phoneme register is shifted left by one phoneme unit and the empty slot is filled with unknowns (?). These unknowns are then filled by the [z] phoneme. This pattern of behavior corresponds to the English pluralization rule that nouns ending in a voiced phoneme are pluralized by appending the [z] phoneme. In classifier notation, this voiced-plural rule might be described as follows:

[3] We do not claim to have an adequate theory of meaning. The crude representation of meaning in terms of discrete features is sufficient for our purpose. Our performance model and learning theory do not depend on the details of meaning representation.

```
rule-classifier-voiced-plural
  Phonemes: dc.dc.dc.[+voice].z
  Source-Grammar:  [+noun -plural]
  Target-Grammar:  [+noun +plural]
  Control:  [shift
                  [direction  : left
                   start loc  : 0
                   unit       : 1
                   fill symbol: ?]]
            [unlock
              phoneme slot  0: no
              phoneme slot -1: no
              phoneme slot -2: no]
```

The first component of a rule-classifier is the phoneme slots. We restrict the number of slots to a small number because we expect the constraints on the phoneme slots to be local in space and time. The voiced-plural rule-classifier has 5 phoneme slots. The first three slots are all don't-cares, indicating that the classifier is not connected to the slots corresponding to time = $-4, -3$, and -2 in the phoneme register. The time -1 slot has the voicing bit on. The time 0 (most recent) slot contains the distinctive features for the [z] phoneme. The source-grammar describes the grammatical features of the rote-classifier (such as "apple") before the conflict in the plural bit is observed. The target-grammar describes the grammatical features of the rote-classifier (such as "apples") after the conflict.

The control part of the rule-classifier has two components. First, it specifies a shifting action on the phoneme register. The content of the phoneme register is shifted left starting at location 0 by one unit with "?" as the fill symbol. Second, the unlock mask specifies whether the three most recent phoneme slots can be unlocked and written by the classifier. In any phoneme slots the unknowns can always be filled. However, to overwrite the bits that have definite values, a classifier must have the unlock privilege on the slots to which the bits belong. For example, the voiced-plural rule-classifier cannot overwrite any of the three phoneme slots because it does not have unlock privileges on these slots. Other rule-classifiers (such as those describing irregular past tense formation) can have unlock privileges to overwrite some of these phoneme slots.

Rule-classifiers are intended to capture phonological constraints. Most phonological constraints deal with predictable regularities of the sound structure that are blind to semantic information. That the plural of "cat" is pronounced as cat[s] but not *cat[z] has nothing to do with the fact a cat is a domesticated carnivorous mammal. It is convenient to assume that whatever semantic information that might be relevant to the applicability of a phonological constraint is already abstracted into the grammatical information. This assumption simplifies the description of a rule-classifier: a rule-classifier is not connected to the meaning register and therefore its description does not contain any meaning component.

A rule-classifier can run in both the forward direction (from source to target) and the backward direction (from target to source). For example, given the phoneme sequence [ae.p.l] of the word "apple," and the [+noun, +plural] grammatical features, the voiced-plural rule-classifier can cause the content of the phoneme register to be shifted left. On the other hand, given the phoneme sequence [ae.p.l.z] and the [+noun,−plural] features, the same rule-classifier can cause the phoneme register to be shifted right to get rid of the terminal [z] phoneme.

The core of the performance model is a collection of classifiers. These classifiers continuously monitor the data registers. When a linguistic event appears, the classifiers will be excited to various degrees. We define the state vector of the performance machine to be the bit vector formed by concatenating the contents of the phoneme, grammar, and meaning registers. The excitation of a rote-classifier is defined as the difference between the number of bits of the rote-classifier and the Hamming distance between the rote-classifier and the machine state vector.

A rule-classifier is *applicable* if either its source-grammar bit or its target-grammar bits matches those in the grammar register. The excitation strength of a rule-classifier depends on whether the shift action is executed or not. We define the data vector of a rule-classifier to be the concatenation of its phoneme and grammar bits. The data vector can be unshifted or shifted. The unshifted data vector of a rule-classifier is simply the concatenation of the phoneme bits of the rule-classifier and the target-grammar bits. The shifted data vector of a rule-classifier is defined for two cases. First, if the rule-classifier runs in the forward direction[4] (e.g., pluralizing a singular noun), the shifted-data-vector is the concatenation of the right-shifted phoneme bits of the classifier and the target-grammar bits. Second, if the rule-classifier runs in the backward direction[5] (e.g., giving the singular form of a plural), the shifted-data-vector is the concatenation of the phoneme bits of the classifier and the source-grammar bits.

The excitation of a rule-classifier is defined as:

$$\begin{aligned}\text{excitation (rule-classifier)} = \text{minimum} (\ & \text{Hamming-distance (shifted-data-vector,machine-state-vector)},\\ & \text{Hamming-distance (unshifted-data-vector,machine-state-vector)})\end{aligned}$$

The constraint propagation process consists of three steps:

[4]We recognize this situation if the target-grammar bits of the rule-classifier match those in the grammar register.

[5]We recognize this situation if the source-grammar bits of the rule-classifier match those in the grammar register.

1. Rank the classifiers according to their excitation strength.
2. Among those classifiers whose excitation exceeds a certain threshold, activate the ones that have the largest number of matching definite bits (i.e., bits that are either 1 or 0). (If none of the classifier has a strong excitation, then no classifier will be activated.)
3. The activated classifiers fill in the contents of the data registers according to the constraints they enforce.

The propagation process continues until there are no further changes in the data registers.

4.2 Constraint Propagation: Meaning to Sound

To illustrate how constraint propagation is used to fill in details we examine a simple situation. Assume that at some time the meaning identifier describing a red, round, edible fruit appears in the meaning register. These bits might have been set by a vision module that has recognized an apple or a picture of an apple, or perhaps by an olfactory module that has recognized the smell of an apple. We also assume that for some reason the plural bit of the grammar register is set, perhaps because the picture shows two apples.

Suppose also that at this point the performance model has two classifiers: a rote-classifier for the apple constraint, which captures the correlation between the phoneme sequence [ae.p.l], the [+red +round +edible +fruit] meaning, and the [+noun −verb −plural ...] grammar, and a rule-classifier for the voiced-plural rule, which captures the phonological rule that the plural of a noun ending in a voiced phoneme is formed by appending a [z] phoneme to the noun.

In general there may be many other rote-classifiers that capture correlations among the data registers for other words, and many other rule-classifiers that capture other phonological rules. These classifiers constitute the representation of the main body of the lexicon.

The situation at the initial time is depicted in Figure 3. The initial situation triggers the following sequence of events. The content of the meaning register is sufficient to activate the constraint described by the classifier for apple. This constraint then attempts to set as many unknown bits as it can. It asserts the bits describing the phoneme sequence into the phoneme register. This encounters no resistance because all of those bits were initially unknown. The apple constraint also sets some grammar bits. The noun bit is turned on and the verb bit is turned off. However, a conflict arises over the setting of the plural bit. The picture of two apples forced the plural bit on, but the apple constraint is trying to assert a singular. Figure 4 shows the contents of the registers at this point.

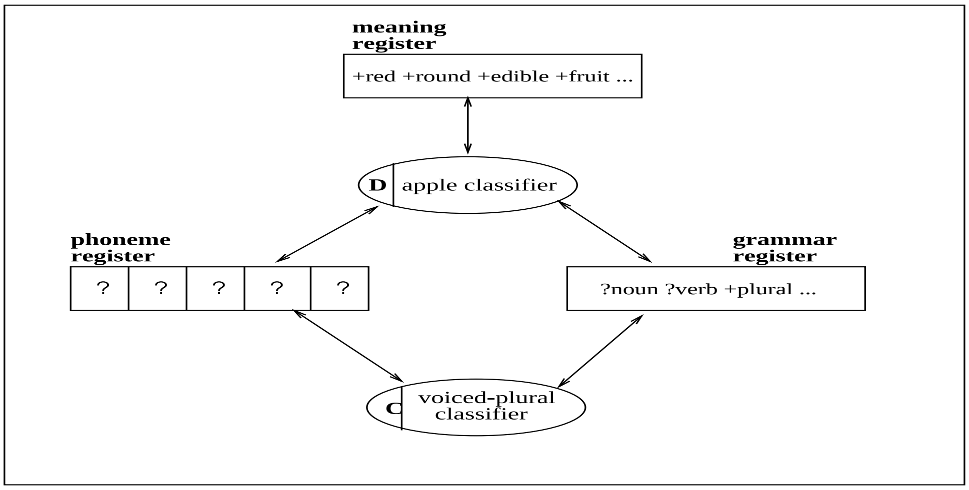

Figure 3: *Meaning to Sound snapshot 1.* Initial situation: An event fills the meaning register with features describing an apple, and the grammar register with the [+plural] feature. The performance model has two classifiers: a rote-classifier representing the apple constraint and a rule-classifier representing the voiced-plural rule. The job of the performance model is to fill the slots of the phoneme register with the sound sequence corresponding to the plural of "apple."

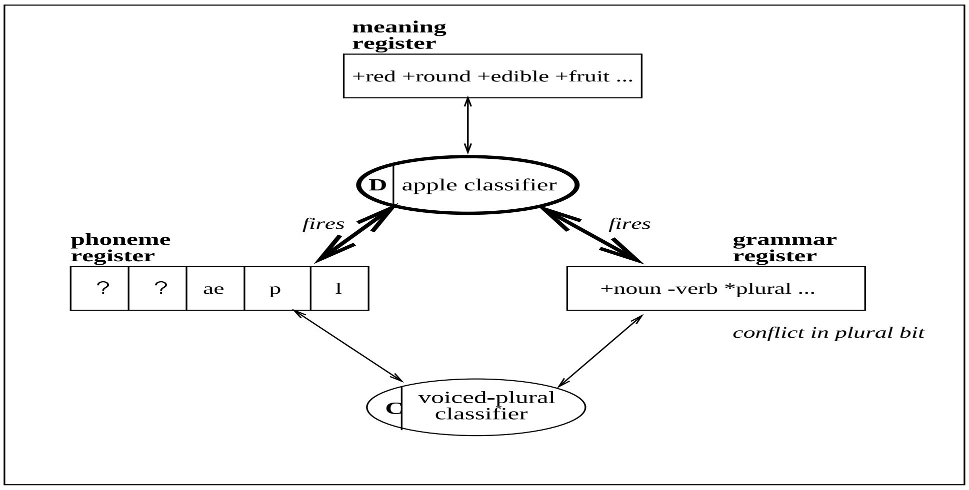

Figure 4: *Meaning to Sound snapshot 2.* The apple classifier is excited by the content of the meaning register. It fires and sets the phoneme register with the sound sequence [ae.p.l], and additional bits in the grammar register (e.g. [-verb]).

All the phoneme bits from the apple constraint are now in the phoneme register. The fact that there is a noun under consideration (+noun in the grammar register), that there is a conflict over the plural bit, and that the terminal [l] phoneme is [+voice] is a sufficient trigger to activate the constraint represented by the voiced-plural classifier. It sends a shift left signal to the phoneme register, moving the phonemes ae.p.l to less recent positions, and locking the determined phonemes so that they cannot change. The most recent phoneme slot is filled with unknowns, which are certainly allowed to change. The apple constraint now becomes less excited because the values it would like in the phoneme register are all in conflict with the ones that are there. The voiced-plural classifier now fills the unknowns in the current phoneme slot with the phoneme [z]. See Figure 5.

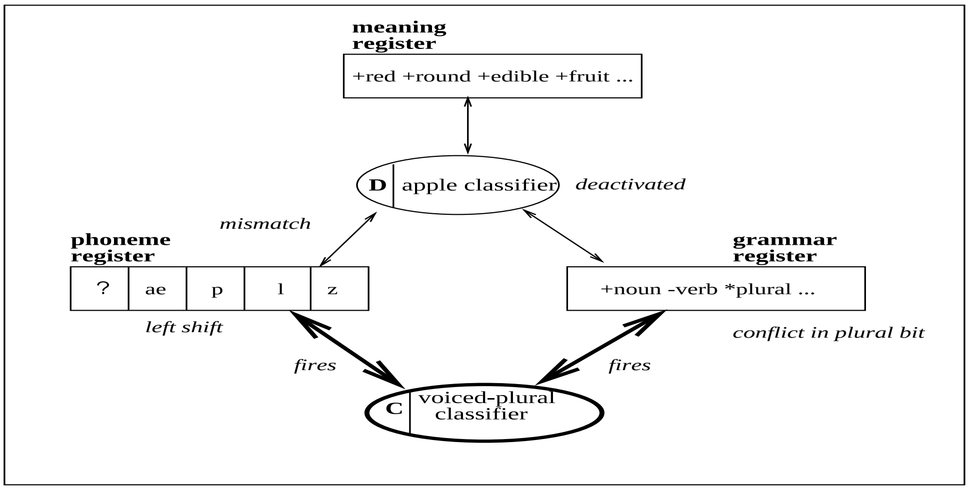

Figure 5: *Meaning to Sound snapshot 3.* The [+noun,*plural] features in the grammar register are sufficient to activate the voiced-plural rule-classifier. The rule-classifier sends a shift left signal to the phoneme register, and fills the unknown terminal slot with the [z] phoneme. The apple classifier is no longer excited because of the mismatch in the phoneme register. So it gives up control over the data registers.

As the apple classifier is deactivated, it drops its attempt to set the plural bit to 0. The noun, the verb, and the plural bits retain their last values. The plural bit is still in conflict, but it will put up no resistance if another constraint tries to turn it on. In particular, the excited voiced-plural rule-classifier restores the plural bit to 1. At this point the system reaches a quiescent state (Figure 6) with a consistent representation of the plural noun pronounced [ae.p.l.z] in the phoneme register.

4.3 Constraint Propagation: Sound to Meaning

Classifiers may also be run in other directions. Suppose the performance model has the same two classifiers as before. If the word "apples" is heard, the sequence of phonemes is shifted into the phoneme register. The situation looks like Figure 7.

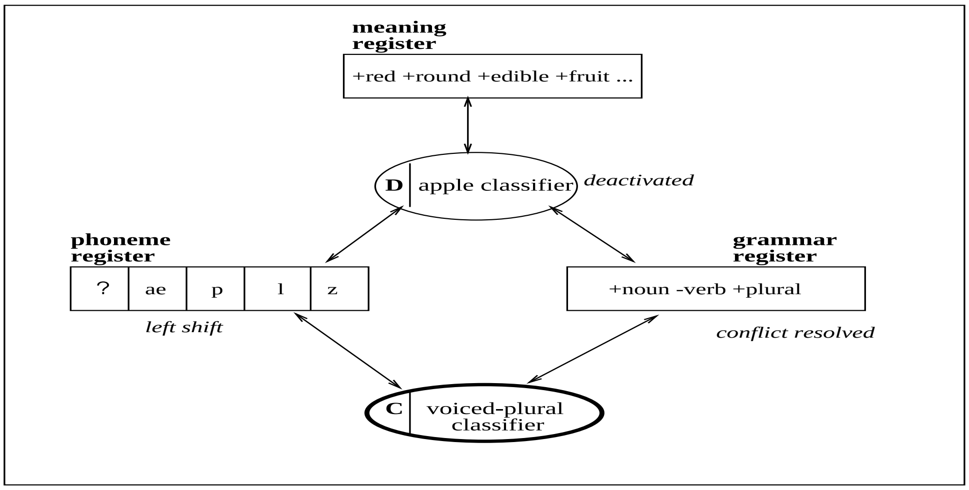

Figure 6: *Meaning to Sound snapshot 4.* No longer excited, the apple classifier drops its attempt to set the data registers. The registers retain their past values. In particular, the plural bit in the grammar register is restored to 1 by the voiced-plural classifier. A quiescent state is reached.

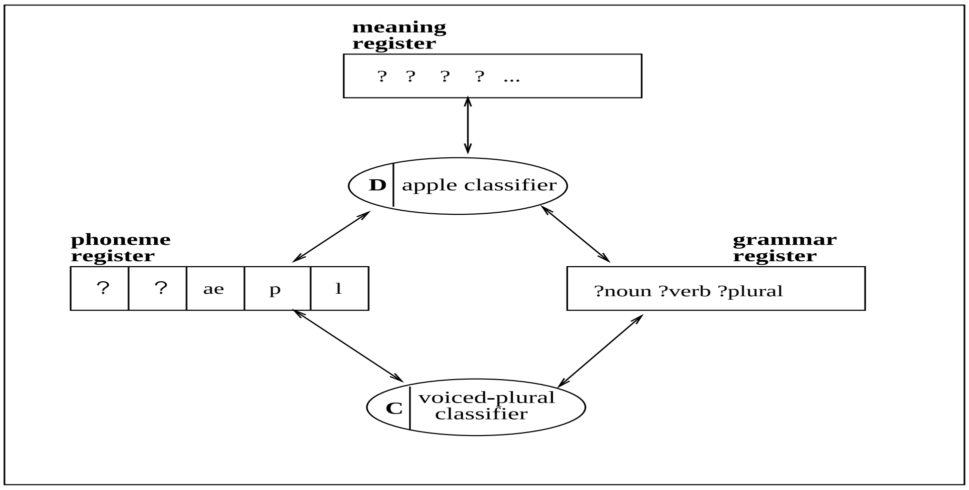

Figure 7: *Sound to Meaning snapshot 1.* Initial situation: The word "apples" is heard. The sound sequence [ae.p.l.z] is being shifted into the phoneme register. The figure shows the situation when the first three phonemes have been heard.

The content of the phoneme register is now sufficient to activate the apple classifier, which sets the meaning and grammar registers (Figure 8).

As the terminal [z] phoneme is shifted into the phoneme register, the apple classifier is deactivated (due to the mismatched phonemes), while the voiced-plural constraint is strongly excited by the present situation. The deactivated apple classifier

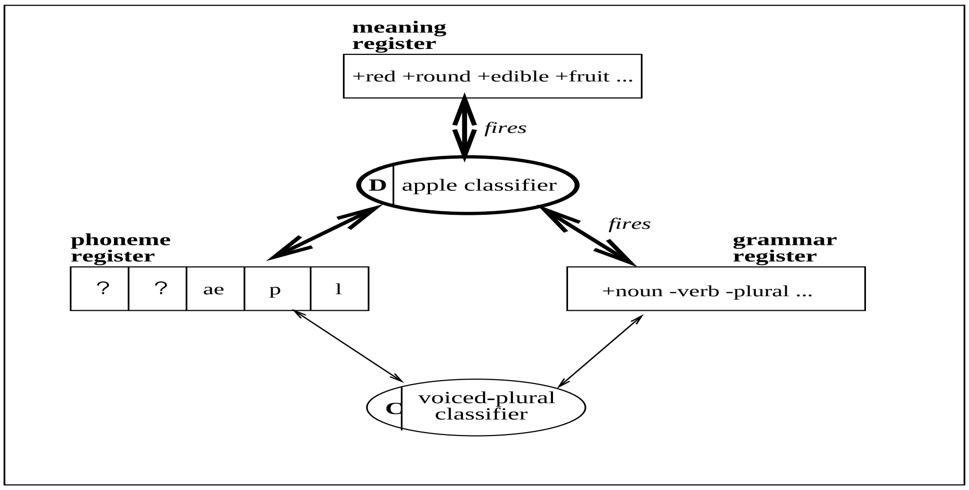

Figure 8: *Sound to Meaning snapshot 2.* The apple classifier is excited by the current content of the phoneme register. It sets the meaning and the grammar registers.

withdraws its hold on the meaning and grammar registers. The excited voiced-plural rule-classifier sets the grammar register and encounters no resistance because the deactivated apple classifier has withdrawn its control over the data registers. In particular, the plural bit is turned on. A consistent and quiescent state is reached. Our mechanism has filled in the meaning and grammar details for the sound sequence [ae.p.l.z] (Figure 9).

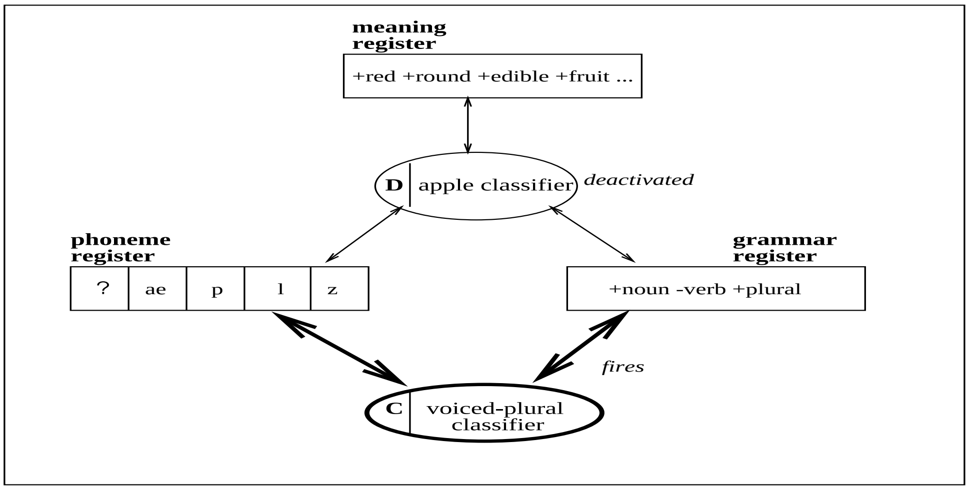

Figure 9: *Sound to Meaning snapshot 3.* As the terminal [z] phoneme is shifted into the phoneme register, the apple classifier is no longer excited and withdraws its control over the data registers. The voiced-plural classifier is activated, turning the plural bit on. A quiescent state is reached.

5 Learning Classifiers

5.1 Origin of Constraints

We could imagine that in a full system there are many classifiers. Some of them (rote-classifiers) embody the special correlations for each word that the system knows. Others (rule-classifiers) are generalizations that allow phonological deductions to be made to fill in properties of words that have not been previously encountered. How are the classifiers constructed?

The data registers are continuously monitored for correlations. Correlations are compiled into rote-classifiers that enforce particular bit patterns among the contents of the phoneme register, the grammar register, and the meaning register. For example, when the phoneme sequence [ae.p.l] is observed to be correlated with the meaning [+red +round +edible +fruit] and grammar [+noun −plural], the apple rote-classifier is constructed to enforce these particular bit patterns.

Once constructed, these rote-classifiers become the primitive objects from which higher order correlations are made. For example, the voiced-plural rule-classifier is constructed from a small number of correlations between rote-classifiers representing nouns and their plurals. Thus the rote-classifiers that summarize correlations among the data registers encode first-order correlations. Similarly, the more general constraints that summarize classes of first-order correlations encode second-order correlations.

To see how such a generalization might be constructed, consider a specific scenario. Suppose that we already have special constraints for "apple," "apples," "dog," and "dogs," and a few others. Comparing the bit patterns enforced by "apple" and "apples," the learner notices that most of the meaning and grammar bits remain constant. The notable exception is the change of the plural bit from -plural to +plural. This change is correlated with the shifting of the phoneme register one unit to the left and the filling of the newly created unknowns by the [z] phoneme bits.

The same transition occurs for "dog" and "dogs," and "gun" and "guns." In each case, the learner observes that (1) the voicing bit of the penultimate phoneme of the plural is on, as well as a number of probably irrelevant bits, and (2) the plural is formed by appending a terminal [z] phoneme to the singular. This observation is compiled and stored into a rule-classifier. The initial rule-classifier could be too specific or too general, but it is then incrementally refined by consideration of further examples.

Generalizations of temporal correlations, such as the voiced-plural classifier just mentioned, is rather simple. But essentially the same mechanism can produce generalizations of spatial-temporal correlations. Shift registers are a means of transforming temporal patterns into spatial ones, and the correlations are actually observed in the spatial representation. Thus, correlations among the data bits may be independent of the temporal behavior. For example, the past tense of "choose" [ch.u.z] is "chose" [ch.o.z] and the past tense of "break" [b.r.e.k] is "broke" [b.r.o.k]. For irregular verbs of this sort, a front vowel in the present tense becomes a back vowel in the past, or a high vowel in the present becomes a low vowel in the past. This change of one or two bits in the distinctive feature representation of the vowel can be captured in a past-tense constraint for the common irregular cases.

Sometimes a constraint needs to be able to warp time, deleting or inserting a phoneme by means of a shift. For example, given a few examples of irregular verb stem and its past tense form (such as "bite" [b.aI.t] and "bit" [b.I.t], "hide" [h.aI.d] and "hid" [h.I.d]), the learner observes that the past tense grammar bit is correlated with the change of a diphthong [aI][6] to a short vowel [I]. This kind of generalization can be captured by the same mechanism.

5.2 Acquisition Procedure

We decompose the learning problem into two subproblems: (1) the detection and grouping of correlations, and (2) the summarization of accumulated correlations. Summarization becomes much easier when it only needs to account for relevant classes of correlations instead of all possible correlations which might include exceptions. The flowchart in Figure 10 depicts the top-level actions of the learning procedure.

Initially the learner has no classifiers. A sequence of words is presented. If the learner can fill in details for a new word without error, then it proceeds to the next word. On the other hand, the learner might fail either because there are not any applicable classifiers or because the applicable classifiers cause conflicts during the constraint propagation. In the first failure situation, either new classifiers are created from accumulated examples, or existing classifiers are generalized to cover the new word. In the second failure situation, existing classifiers are incrementally refined.

In more detail, the learning procedure cycles through the following steps:

Input a word

1. If no classifier is excited, create a rote-classifier for the word, add it to the classifier pool, and go to step 3. Otherwise, the excited classifiers attempt to fill in details.

2. If the excited classifiers successfully fill in details without running into conflicts,

[6] In our system, a diphthong is represented as a sequence of two phonemes. For example [aI] is represented as [a.I].

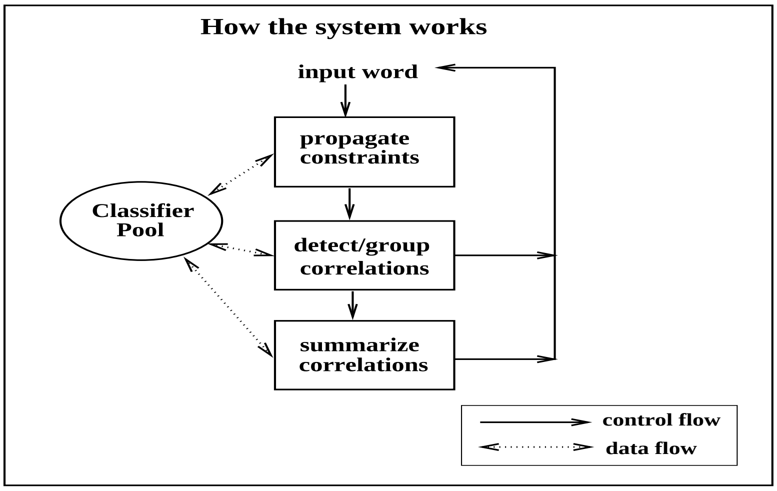

Figure 10: Top-level flowchart for learning classifiers

record in a new rote-classifier any new first-order correlation observed after the constraint propagation. Otherwise, create a new rote-classifier to record the conflicting correlation, and mark this classifier as a non-word (near miss).

3. (a) Find second-order correlations for the newly created rote-classifier by comparing it with rote-classifiers that have the same meaning bits.
   (b) Classify the second-order correlations according to the grammar bits, the shift and unlock actions, and similarity of the rote-classifiers.

4. For each rule-classifier applicable to a second-order correlation[7], do the following:[8]

   (a) If the correlation is consistent with the rule-classifier, then record the correlation as an example covered by the rule-classifier.
   (b) If the correlation is a *false negative* (i.e., the rule-classifier is applicable but the phoneme bits do not match), then incrementally *generalize* the rule-classifier to cover the correlation. Add the refined rule-classifier to the classifier pool.
   (c) If the correlation is a *false positive* (i.e., the rule-classifier generates a near miss), then incrementally *specialize* the rule-classifier to avoid covering the correlation. Add the refined rule-classifier to the classifier pool.

5. If the correlation is not covered by any rule-classifier, accumulate it into a dataset. If the number of correlations of the same type (i.e. same grammar

[7]A rule-classifier is *applicable* to a second-order correlation if its grammar and control components match those of the correlation.

[8]Unlike the version space algorithm [8], our algorithm does not maintain all the most general and most specific generalizations consistent with the current set of examples. Our algorithm can also handle disjunctive generalizations and noise.

bits, and same shift and unlock actions) exceeds a threshold, invoke the summarization procedure to extract the regularities of these correlations. Otherwise, input the next word and go to step 1.

The details of steps 4 and 5 are expanded in the flowchart in Figure 11.

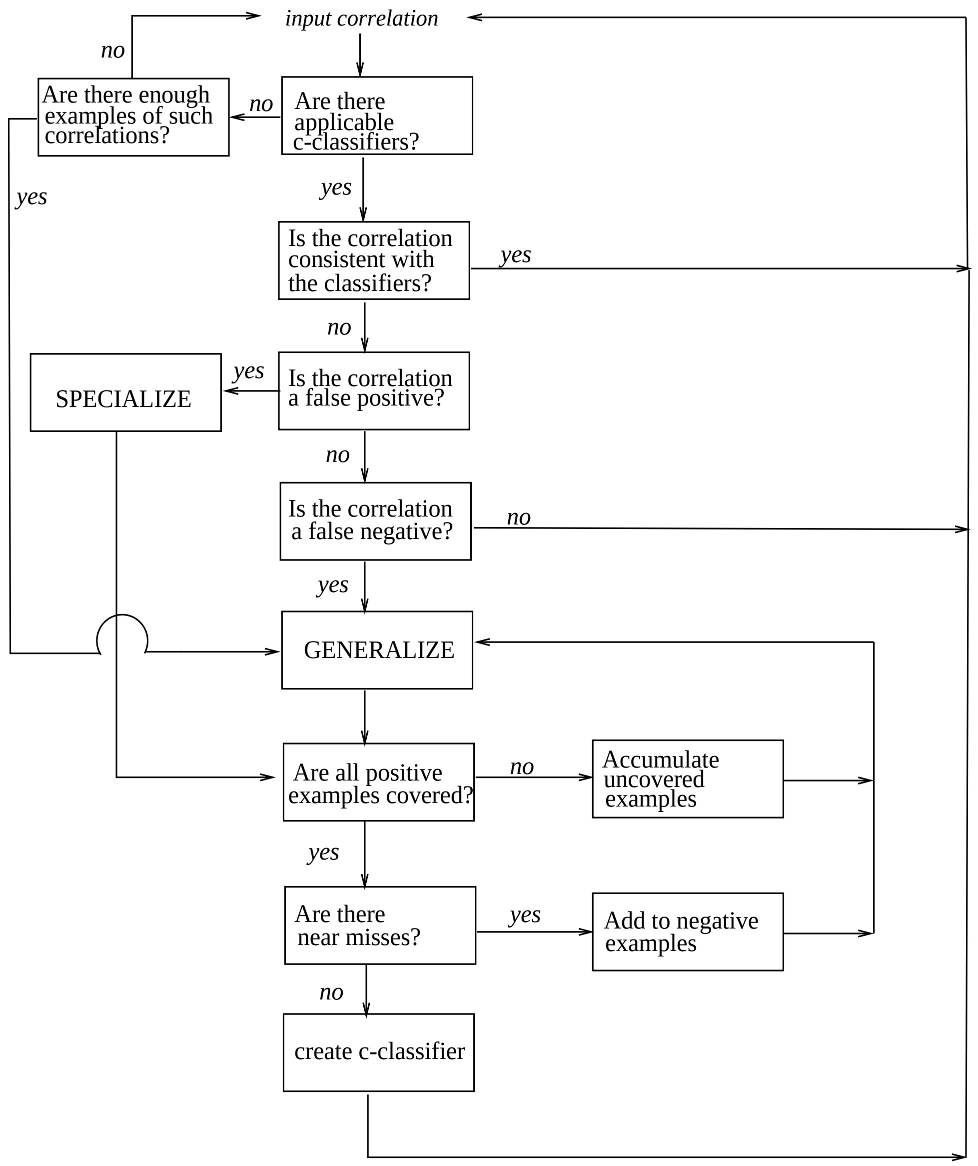

Figure 11: A flowchart explaining the details of steps 4 and 5 of the learning procedure. When there are no more correlations, control is transferred back to the top-level loop (previous figure) to wait for the next input word.

Let us follow the learning procedure step by step for a specific scenario. (The example will illustrate most but not all of the steps in the acquisition procedure.) Suppose that to begin with the learner has no classifiers and is presented four noun pairs in succession: cat/cats `[k.ae.t.s]`, dog/dogs `[d.).g.z]`, duck/ducks `[d.^.k.s]`, and gun/guns `[g.^.n.z]`. We assume the input data contains the phonetic, grammatical, and meaning features associated with a given word.

The learner first encounters the word "cat." Since there are no classifiers to consider, the learner constructs a rote-classifier to record the correlations among the bits of the data registers:

```
rote-classifier-cat
  Phonemes: k.ae.t
  Grammar:  [+noun -verb -plural ...]
  Meaning:  [+animal +tail +4-legged -harmful ... ]
```

The learner does not find any second-order correlations because there are no other known classifiers that share the same meaning features.

The learner proceeds to process the next word, "cats." The rote-classifier-cat is excited because it matches the initial sequence of phonemes of "cats." It attempts to enforce its meaning and grammar bits. There is an initial conflict in the plural bit, but the conflict is resolved when the cat classifier withdraws its control after the entire phoneme sequence is heard. A new rote-classifier is constructed to record the first-order correlations:

```
rote-classifier-cats
  Phonemes: k.ae.t.s
  Grammar:  [+noun -verb +plural ...]
  Meaning:  [+animal +tail +4-legged -harmful ... ]
```

The learner notices that rote-classifier-cats and rote-classifier-cat have the same meaning bits, and constructs a second-order correlation to relate the two rote-classifiers. The correlation process finds by convolution the shifting action that produces the maximal match between the phoneme bits of the two classifiers. The process also determines the unlock mask by comparing the shifted phoneme bits of "cat" with the phoneme bits of "cats." In this case, the phoneme bits (except of course the bit positions that are filled by unknowns) are not allowed to be modified. The second-order correlation is classified by its target-grammar bits and control bits, and is accumulated in a dataset indexed by the same classification bits. The second-order correlation is represented by a bit vector, which can be symbolically abbreviated as:

```
correlation-cat/cats:
 Source-classifier: rote-classifier-cat
 Target-classifier: rote-classifier-cats
 Source-Grammar:  [+noun -verb -plural ...]
 Target-Grammar:  [+noun -verb +plural ...]
 Control:  [shift
                 [direction  : left
                  start loc  : 0
                  unit       : 1
                  fill symbol: ?]]
            [unlock
              phoneme slot  0: no
              phoneme slot -1: no
              phoneme slot -2: no]
```

The learner does not have any rule-classifiers yet, so nothing triggers on the second-order correlation. There are also not enough examples of this type of correlation to trigger the summarization process. So the learner proceeds to the next input.

Similar constructions of rote-classifiers and second-order correlations occur for dog/dogs, duck/ducks, and gun/guns. After the second-order correlation is constructed for gun/guns, the learner has accumulated four examples of this type of correlation. This number of examples is sufficient to trigger the summarization process.

The summarization process attempts to extract a common pattern from the examples. In particular, the process looks for a general description of the phoneme bits common to the target classifiers (cats/dogs/ducks/guns). It might also be useful to obtain a description for the source classifiers, but we have not found this necessary for learning English plural and past tense rules.

The core of the summarization process is a generalization algorithm, which aims to find a description that covers the phoneme bits of the target classifiers (the positive examples) and fails to cover those of the source classifiers (the negative examples). A description is said to cover an example if the example matches all the conditions in the description; the example is called a positive example. Otherwise, it is a negative example.

The description language consists of disjunctive normal forms. A boolean formula is in disjunctive normal form if it consists of disjunctions of clauses each of which is a conjunction of literals. We need this expressive power to handle disjunctive rules and exceptions, which are common in phonological rules.

Starting with the phoneme bits of a target classifier as the initial description, the generalization algorithm does a specific-to-general search in the space of possible descriptions. For example, an initial seed might be the phoneme bits for "cats." The seed is a bit vector of 56 bits (14 bits for each of the 4 phonemes [k.ae.t.s]), which can be thought of as a logical conjunction of boolean features:

```
01011001000000101001000011000100000111000001000001110101
<------ k ---><----- ae  --><----- t ----><----- s ---->
```

```
Seed              : [k.ae.t.s]
Positive examples: [k.ae.t.s] [d.).g.z] [d.^.k.s] [g.^.n.z]
Negative examples: [k.ae.t] [d.).g] [d.^.k] [g.^.n]
```

The initial seed covers one positive example and no negative examples. At each step of the search, the algorithm considers all generalizations that involve the dropping of one or two features (i.e., changing a 0 or 1 to a don't care). To limit the number of generalizations to be considered, the algorithm generalizes one phoneme at a time from left to right (thus preferring generalizations that keep the most recently heard phonemes). The generalizations are ranked according to a goodness function. The search retains only the best k generalizations without any backtracking. The goodness of a generalization increases with the number of positive examples it covers and with the number of negative examples it does not cover. When each of the best k generalization either covers all positive examples or covers a negative example, the search terminates.

The search eventually produces a description G that covers all four positive examples and avoids all four negative examples. The description says that all positive examples end with either the [s] or [z] phoneme.

```
G: [dc.dc.dc.{s,z}]
```

The next step in the summarization process is to verify covering and over-generalizations.

The generalization G is overly general because applying it to the source classifiers gives not only the correct plural forms (such as [k.ae.t.s]) but also incorrect ones (such as *[k.ae.t.z]). The incorrect ones are treated as near misses (i.e., negative examples that are slightly different from the positive ones). Basically the learner assumes there is only one plural form for each noun. Since it already knows [k.ae.t.s] is the correct one, [k.ae.t.z] must be wrong. Near misses, as we shall see, greatly speed up the discovery of correct generalizations.[9]

[9] Winston [14] emphasized the usefulness of near misses in his ARCH learning program. In our program, the near misses are not supplied by a teacher or given in the input. They are generated internally.

The generalization algorithm is re-invoked with the addition of these new negative examples:

```
Seed             : [k.ae.t.s]
Positive examples: [k.ae.t.s] [d.).g.z] [d.^.k.s] [g.^.n.z]
Negative examples: *[k.ae.t.z] *[d.).g.s] *[d.^.k.z] *[g.^.n.s]
                   [k.ae.t] [d.).g] [d.^.k] [g.^.n]
```

This time the search results in a disjunction of two generalizations G1 and G2:

```
G1:  [dc.dc.[-voice,-strident].s]
G2:  [dc.dc.[+voice,-strident].z]
```

G1 covers two positive examples: "cats" and "ducks." G2 covers the remaining two: "dogs" and "guns." The two generalizations are verified as before. However, this time the generalizations are consistent: there are not any new exceptions or near misses. Note that we can already read off symbolic descriptions from G1 and G2, which resemble those found in linguistics books.

The summarization process then creates rule-classifiers to represent the consistent generalizations. These rule-classifiers are now available for constraint propagation, and are subject to further refinement when new examples appear.

That the two rule-classifiers can be learned in a few examples has interesting psychological implications. Berko in her well-known study of children's learning of English morphology made the following observation. While the first-graders can apply the add [z] and add [s] pluralization rules productively to new words, they fail to apply the add [I.z] rule to nonce words like "tass" or "gutch." When asked to produce the plural of a nonce word ending in [s] or [ch], they either repeat the word in its singular form or fail to respond. In no cases do they give wrong answers like *tass[z], *tass[s], or *gutch[z], and only in few cases do they respond with *gutch[s]. The children fail to use the [I.z] rule productively despite the fact that they can recognize and use real words like "glass" and "glasses" correctly.

Our acquisition theory gives a plausible explanation of this behavior. Even with a small number of examples, the children can acquire general rules like G1 and G2 that prevent the appending of the [s] phoneme and [z] phoneme to nouns ending in a strident. When asked to pluralize a nonce word like "tass," our performance program leaves the word unchanged because neither of the rules will be excited. Our theory also predicts that prior exposure to examples of the adding [I.z] rule is not necessary to produce the behavior as observed by Berko.

This example illustrates the basic learning mechanisms. One reason why the generalization is so effective is that exceptions like "foot/feet" [f.U.t]/[f.i.t] (which involves vowel change) and "leaf/leaves" [l.i.f]/[l.i.v.z] (which involves voicing the last consonant [f] to [v] in the plural form) are separated into different correlation types by the correlation process. The division of labor between the correlation and summarization processes allows the generalization algorithm to work on rather clean examples.

We now back up and cover two topics in more detail: classifying correlations and generalizing classifiers.

Classifying correlations: Shift and Similarity

The second-order correlations are divided into two types according to whether the target classifiers have unlock privileges to change the slots of the phoneme register. The correlations whose target classifiers do not have any unlock privileges (e.g., the correlation between a regular noun and its plural) are called *non-mutable.* The non-mutable correlations are further subdivided by the type of shifting action and the grammar bits. The mutable correlations (e.g., the correlation between an irregular verb and its past tense) is further subdivided according to the similarity of the target classifiers, where the similarity of two classifiers is measured by the Hamming distance between their phoneme and grammar bits.

```
                                | membership
             |- mutable       --| determined
correlation -|                  | by similarity
             |
             |                  | membership determined
             |- non-mutable --| by shift action
                                | and grammar bits
```

Classifier generalization: Cube Growing Algorithm

The generalization algorithm is best explained geometrically. One can associate boolean variables with spatial dimensions. A relation with n boolean variables defines an instance space of 3^n possible instances (because each bit can be 0, 1, or don't-care). An n-bit vector corresponds to a point in the instance space. For example, a 3-bit vector "111" can be interpreted as a vertex (0-cube) in the boolean 3-dimensional cube. The bit vector "-11" is a line (1-cube). The bit vector "- - -" with all don't cares is the universal 3-cube; it covers the entire space.

Classifier generalization can be visualized as the growing of n-cubes to cover positive instances without overlapping negative ones. Cube growing is done by raising bits of a vector, i.e., turning 0's or 1's to don't cares. Conversely, specialization is the shrinking of n-cubes. Shrinking is accomplished by lowering don't cares to 0's or 1's.

The generalization space of possible phoneme bits for a classifier is $O(3^n)$, where n is the number of phoneme bits. For example the generalization space for a classifier with five phonemes contains $O(3^{70})$ instances. To explore this huge space, the generalization process relies on three search biases:

1. Whenever possible it revises the current best classifiers instead of starting from scratch,
2. It prefers classifiers that contain the most recently heard phonemes, and
3. It is non-aggressive: it does not deliberately look for the minimal classifiers (i.e., the largest n-cubes) to cover a given set of correlations.

The generalization procedure is a beam search with a simple goodness function. A beam search is just like a best-first search except that it does not backtrack. It keeps a fixed number of current-best classifiers and discards the remaining ones.

The goodness of a cube is given by:

$$\text{goodness (cube)} = \text{Pc (cube)} + \text{Nc (cube)}$$

where Pc is the number of positive examples it covers, and Nc is the number of negative examples it does not cover. To break ties in the goodness score, generalization prefers larger cubes with higher Pc. The phonemes are ordered by recency. The bits of the least recently heard phoneme are raised first, one or two bits at a time. The best k such cubes are selected for further expansion. The search terminates when either all positive examples are covered, or a negative example is covered.

The specialization algorithm is an incremental general-to-specific search. It aims to avoid negative instances while retaining most of the positive instances. The algorithm uses the same beam search and goodness function. To break ties in the goodness score, specialization prefers smaller cubes with higher Nc. It repeatedly shrinks cubes by lowering don't cares, at most two bits at a time, starting from the most recently heard phoneme. It only explores candidate cubes that contain some possible phonemes. This is done by matching a candidate cube with all the phonemes known to the learner. For instance, a cube with the nasal bit on and the voicing bit off is discarded because in English there is no [−voice,+nasal] phoneme. The specialization process terminates when none of the negative examples is covered.

6 Experimental Results

The corpus consists of 250 words. The words are common nouns (about 50) and verbs (about 200) that first-graders might know. The nouns are the singular and plural forms of common animals and everyday objects (e.g., cat, cats, dog, dogs, cup, cups, man, men). The corpus includes most of the regular and irregular verbs used in the psycholinguistic experiments of Marcus et. al. [7] on English tenses (e.g., go, went, play, played, kick, kicked).

Consistent with the observation that a human learner receives little explicit correction, the corpus contains only positive examples. However, the lack of external negative evidence does not rule out the possibility that the learner can generate internal negative examples when testing out hypotheses. These internal negative examples, as we have seen, play a significant role in the rapid learning of classifiers.

The data record for each word in the corpus has five pieces of information: (1) word identifier, (2) word spelling, (3) a unique meaning identifier (e.g., "cat" and "cats" have the same meaning id, but "cat" and "dog" do not), (4) its pronunciation as a sequence of phonemes, (5) its grammatical status (e.g., whether it is a noun or verb, singular or plural, present or past). The data records for "cat(s)" and "dog(s)" are shown below:

```
word-id    spelling meaning-id pronunciation  grammar
-------------------------------------------------------------
12789          cat   6601       k.ae.t.        Noun  Sing
12956         cats   6601       k.ae.t.s.      Noun  Plu
25815          dog  13185       d.).g.         Noun  Sing
25869         dogs  13185       d.).g.z.       Noun  Plu
```

The data records are pre-processed to produce bit vector inputs for the performance model and learner. The output of the performance model and learner is bit vectors that typically have a straightforward symbolic interpretation.

In all the experiments below, we use the same parameter settings for the beam search width[10] (in the generalization algorithm), excitation threshold (in constraint excitation), and similarity threshold (in classifying correlations). The results are not sensitive to the particular parameter settings.

Experiment 1: How regular pluralization rules are learned

The objective of this experiment is to determine what pluralization rules are acquired by our learner given a sample of common nouns and their plurals. The formation of English plurals is unusually regular. There are very few irregular plural nouns. This property of English might lead one to propose learning mechanisms that exploit the statistics of regular plurals by training on a large number of examples so that any new test noun is sufficiently similar to a known one to produce the closest matched

[10]The beam search width is set to 2.

plural ending.

But there is evidence that the statistical property may not be essential to the acquisition of regular rules. For example, Marcus et. al. [7] and Clahsen [3] showed that the German -s plural behaves like a regular rule despite the fact that the rule applies to fewer than 30 common nouns. This observation raises the question of how a child can acquire regular rules from very few examples. The experiment will show that our learner can acquire generalizations that closely resemble those described in linguistics books after seeing on the order of 10 examples.

The input of this experiment consists of 22 noun-plural pairs. The particular choices of words are not very important as long as there are some examples of singular nouns ending in different phonemes. We pick a few examples for each of five types of plural formation:

[s]	[z]	[I.z]	semi-regular	irregular
cake(s)	bottle(s)	box(es)	house(s)	man/men
cat(s)	boy(s)	bush(es)	leaf/leaves	foot/feet
chief(s)	dog(s)	church(es)		
cup(s)	girl(s)	dish(es)		
fruit(s)	gun(s)	glass(es)		
month(s)		horse(s)		
		nose(s)		

The first three columns are the regular plural forms. The semi-regular column contains words whose singular forms end in voiceless fricatives ([s] and [f]), which become voiced in their plural forms ([z] and [v] respectively). The irregular nouns involve internal vowel changes instead of adding affixes.

The 22 pairs are fed to the learner sequentially in a random order once. The results presented here are typical because the final rules acquired are found not to be sensitive to the order of presentation.

The entire learning session takes about 15 minutes on a Sun Sparc 10. After the presentation of all 22 pairs, the learner has acquired five rule-classifiers and four exceptions. The phoneme bits of the classifiers are as follows:

```
1. [dc.dc.[+voice,-strident].z]
2. [dc.dc.{y,e,I,v}.z]
3. [dc.dc.[-voice,-strident].s]
4. [dc.dc.[-voice,-coronal].s]
5. [dc.[+coronal,+strident].I.z]
```

Rule 3 is acquired after the presentation of 7 pairs. Rule 1 is acquired after 9 pairs. Although the list of 22 pairs is sufficient for the acquisition of the English plural rules, the list is far from minimal. For example, rule 5 is acquired after the learner encounters only 4 examples of nouns ending in [I.z]. The remaining three [I.z] examples are redundant. The irregulars also do not affect the acquisition of the regular rules. They are represented as specific rote-classifiers. The correlations among the irregulars are grouped into three exceptional classes: (1) foot/feet and man/men, (2) leaf/leaves, and (3) house/houses.

Notice that we can almost read off the standard English pluralization rules from these classifiers. There are, however, two differences. First, the standard English pluralization rules are typically ordered:

a. If the noun ends a phoneme containing the features [+strident, +coronal] (i.e., one of the sounds [s], [z], [sh], [zh], [ch], [j]), the plural affix is [I z].
Otherwise,
b. If the noun ends in a [+voice] phoneme, the affix is [z].
c. If the noun ends in a [−voice] phoneme, the affix is [s].

In our system, the classifiers are activated in parallel, with the most excited ones gaining control over the data registers.

The second difference is that the unvoiced-plural rule c is represented by a disjunction of two rules 3 and 4 in our system. Rule 3 covers nouns ending in consonants [t], [k], [p], [h], or [th]. Rule 4 covers nouns ending in the strident [f] or the non-coronal stops [k] and [p]. Similarly, the voiced-plural rule b is split into rules 1 and 2.

The learner also exhibits intermediate behaviors similar to those of young children. After rule 1 and rule 3 are acquired, the performance program produces plurals like *foot[s] and *man[z]. Upon presentation of the nonce word "wug," it gives wug[z]. For nonce words ending in a strident like "tass" or "gutch," it gives the unaltered singular forms as plurals.

Experiment 2: Learning plurals in the presence of noise

In this experiment, we examine the behavior of the learner when the input contains error. The learner is given the same 22 noun-pairs from experiment 1 and an incorrect plural form *cat[z].

The result is interesting: The incorrect form does not affect the acquisition of the correct constraints. The learner acquires the same 5 constraints as in experiment 1. An additional constraint is created to account for the incorrect *cat[z]:

```
6. [dc.[-tense,-strident],t,z]
```

The [tense] feature indicates whether the articulatory gestures associated with a phoneme are produced with considerable muscular effort and a long duration. Non-tense phonemes (such as [ae]) are produced rapidly and with less effort than the tense ones (such as [i]).

Experiment 3: How regular past tense rules are learned

This experiment shows what regular past tense rules are acquired from common verbs and their past-tense forms. Like the plurals, English past tense has regular affixes [t], [d], or [I.d] depending on the properties of the last phoneme of the verb stem. Verb stems that end in a voiced phoneme other than [d] receive [d], while those that end in an unvoiced phoneme other than [t] receive [t]. For verb stems ending in [d] or [t], the syllable [I.d] is added to the stem.

Unlike those in the plural rules, the exceptions in the regular past tense rules (for stems ending in [d] or [t]), do not form a natural class in terms of distinctive features. It is therefore interesting to see how the learner constructs disjunctive rules to capture the regularities in the input data.

The input consists of 21 verbs and their past-tense forms:

```
    [t]       [d]       [I.d]       irregular
------------------------------------------------------------
    danced    answered  added       draw/drew
    dropped   called    needed      sing/sang
    fixed     cried     painted     feed/fed
    kissed    hugged    waited
    laughed   turned
    liked
    looked
    touched
    walked
```

The stem-past pairs are presented sequentially in a random order once. After the presentation of all the verb pairs, the learner has acquired six rule-classifiers and three exceptions (the irregulars). The phoneme bits of the classifiers are as follows:

```
1. [dc.dc.[+voice,+sonorant].d]
2. [dc.dc.[+voice,-coronal].d]
3. [dc.dc.[-low,-round,-tense,+continuant].d]
4. [dc.dc.[-voice,+strident].t]
```

```
5. [dc.dc.[-voice,-coronal,-continuant].t]
6. [dc.{d,t}.I.d]
```

Rules 1, 2, and 3 together cover all the verb stems that end in a voiced phoneme other than [d]. These rules overlap in the examples they cover. Rule 1 covers the majority of these cases (all the vowels, nasals, liquids, and glides). Rule 2 covers the voiced non-coronal stops ([b] and [g]) as well as some of the cases covered by rule 1, while rule 3 covers the voiced stridents. An over-general rule [dc.dc.[+voice].d] is acquired after the presentation of 11 stem-past pairs. The learner refines the rule to rule 1 on examining another 6 stem-past pairs (not all of which are relevant to rule 1).

Similarly, rules 4 and 5 cover verb stems that end in an unvoiced phoneme other than [t]. Rule 4 covers stems ending in [k] or [p], while rule 5 covers the unvoiced stridents. Rule 4 is acquired after 8 stem-past pairs. Rule 5 is acquired after 11 pairs.

Rule 6 directly corresponds to the add [I.d] rule. The rule is acquired after 18 stem-past pairs.

The experiment shows that the learner can indeed acquire past tense rules that lead to the correct behavior.

Experiment 4: Learning plural and past tense rules together

The objective of experiment 4 is to test whether higher-order correlations can be extracted from the past tense rules and the plural rules, which have very similar structures.

The generalization algorithm when presented with the past tense rules (from experiment 3) and the plural rules (from experiment 1) produces a third-order correlation that says the voicing bits of the ending phoneme of the stem and the affix have to match:

```
[dc.dc.[-voice].[-voice]]
[dc.dc.[+voice].[+voice]]
```

Our learning theory gives a plausible mechanism to produce the kind of compact, elegant phonological rules that linguists develop to explain complicated phonological processes in terms of the interactions of nearly independent and widely applicable rules. The theory also exhibits the "Waltz" effect [13] that learning becomes more effective when the learner is exposed to more varieties of constraints.

Experiment 5: How irregular past tense patterns are learned

This experiment aims to show what our learner can learn from irregular verbs. The input consists of 55 common irregular verbs (such as eat, blow, buy, go) and their past forms.

The learner acquires six rule-classifiers that cover 19 of the 55 input verbs:

```
[dc.dc.ae.ng]       rang, sang
[dc.dc.a.t]         forgot, got, shot
[dc.E.n.t]          bent, lent, meant, spent
[dc.dc.{r,l}.u]     blew, drew, grew
[dc.dc.).t]         bought, brought, caught, taught
[dc.dc.o.z]         chose, froze, rose
```

Since irregular verb forms are in general not productive and idiosyncratic (such as go/went), we expect they fall into many sub-classes. The results confirm our expectation. The learner is able to find the more common patterns (such as blew/drew/grew and bought/caught/taught). The results also suggest that most irregulars are just rote-learned and the learner makes few generalizations about these forms.

7 Discussion/Conclusion

We have demonstrated that a rather simple mechanism, which can be implemented in surprisingly small amounts of physical hardware (or meatware?), exhibits behavior comparable to the behavior of small children in the task of learning and using phonological knowledge. In our theory phonological knowledge is encapsulated as a set of boolean constraints. These constraints operate on the classical linguistic representation of a pattern of sound in terms of phonemes and binary distinctive features. The knowledge is applied in phonological performance by classical constraint propagation. The constraints are learned by an incremental process with two phases: detecting correlations and summarizing the accumulated regularities. These summaries are specifications of the particular boolean constraints to be imposed. The summaries may be incrementally generalized or specialized as new data appears. As a bonus, the summaries that are compiled may be read out to obtain recognizable rules of classical linguistics.

Our mechanism has been quite successful for learning a portion of English phonology. Our mechanism yields almost one-shot learning, similar to that observed in children: It takes only a few carelessly chosen examples to learn the important rules; there is no unreasonable repetition of the data; and there is no requirement to zealously correct erroneous behavior. The mechanism tolerates noise and exceptions. It learns higher-order constraints as it knows more. Furthermore, the intermediate states of learning produce errors that are just like the errors produced by children as they are learning phonology.

While this mechanism has been tested for a chunk of English phonology, it has not yet been extensively tested in all corners of English and it has not yet been tried for other languages (though we have started testing our theory on learning Hebrew verb patterns). This is important, because we intend this to be a strong theory: It had better work in all cases—it is either right or wrong—there are very few parameters that we can wiggle to extend the coverage of the theory if it is wrong.

Over the past few years there has been a rather heated debate between advocates of "Connectionism" and advocates of more traditional "Symbolic Artificial Intelligence." We believe that contemplation of our mechanism for acquiring and using phonological knowledge can shed considerable light on this question.[11] The essence here is in understanding the relationship between the signals in the neural circuits of the brain and the symbols that they are said to represent.

Consider first an ordinary computer. Are there symbols in the computer? No, there are transistors in the computer, and capacitors, and wires interconnecting them, etc. It is a connectionist system. There are voltages on the nodes and currents in the wires. We as programmers interpret the patterns of voltages as representations of our symbols and symbolic expressions. We impose patterns we call programs that cause the patterns of data voltages to evolve in a way that we interpret as the manipulation of symbolic expressions that we intend. Thus the symbols and symbolic expressions are a compact and useful way of describing the behavior of the connectionist system. We as engineers arrange for our connectionist system to exhibit behavior that we can usefully describe as the manipulation of our symbols.

In much the same way, auditory signals are analog trajectories through a rather low-dimensional space—pressure on the eardrum. By signal processing these are transformed into trajectories in a rather high-dimensional space that linguists abstract, approximate, and describe in terms of phonemes and their distinctive features. This high-dimensional space is very sparsely populated by linguistic utterances. Because of the sparsity of this space, we can easily interpret configurations in this space as discrete symbolic expressions and interpret behaviors in this space as symbolic manipulations.

It may be the case that the linguistic representation is necessarily sparse because that is the key to making a simple, efficient, one-shot learning algorithm. Thus

[11] Debate in the context of a specific problem—learning phonological knowledge—is documented in [12, 10, 9, 11].

sparseness of the representation, and the attendant possibility of symbolic description is just a consequence of the fact that human language is learnable and understandable by mechanisms that are evolvable and implementable in realistic biological systems.

So in the case of phonology at least, the Connectionist/Symbolic distinction is a matter of level of detail. Everything is implemented in terms of neurons or transistors, depending on whether we are building meatware or hardware. However, because the representation of linguistic information is sparse, we can think of the data as bits and the mechanisms as shift registers and boolean constraints. If we were dealing with the details of muscle control we would probably have a much denser representation and then we would want to think in terms of approximations of multivariate functions. But when it is possible to abstract symbols we obtain a tremendous advantage. We get the power to express descriptions of mechanisms in a compact form that is convenient for communication to other scientists, or as part of an engineering design.

So what of signals and symbols? There are signals in the brain, and when possible, there are symbols in the mind.

Acknowledgments

We thank Morris Halle for teaching us elementary phonology and helping us to get started in this research. We thank Tom Knight for showing us that an electrical implementation of constraints is feasible. We thank Julie Sussman for numerous criticisms and suggestions to improve the presentation of the material. We thank Patrick Winston for his thoughtful critique and ideas for extending this work to elucidate questions on the general principles of language and learning. We thank Rodney Brooks, Elisha Sacks, Peter Szolovits, and Feng Zhao for comments on the manuscript.

References

[1] J. Berko. The child's learning of English morphology. *Word*, 14, 1958.

[2] Noam Chomsky and Morris Halle. *The Sound Pattern of English*. Harper and Row, 1968.

[3] Harold Clahsen, Monika Rothweiler, and Andreas Woest. Regular and irregular inflection of German noun plurals. *Cognition*, 45(3), 1992.

[4] Susan Ervin. Imitation and structural change in childrren's language. In *New Directions in the Study of Language*. MIT Press, 1964.

[5] Michael Kenstowicz. *Phonology in Generative Grammar*. Blackwell Publishers, 1994.

[6] Brian MacWhinney and Jared Leinbach. Implementations are not conceptualizations: Revising the verb learning model. *Cognition*, 40, 1991.

[7] Gary Marcus, Steven Pinker, Michael Ullman, Michelle Hollander, T. John Rosen, and Fei Xu. *Overregularization in Language Acquisition*, volume 57. Monographs of the Society for research in child development, 1992.

[8] Tom Mitchell. Generalization as search. *Artificial Intelligence Journal*, 18, 1982.

[9] Steven Pinker. Rules of language. *Science*, 253, 1991.

[10] Steven Pinker and Alan Prince. On language and connectionism: Analysis of a parallel distributed processing model of language acquisition. *Cognition*, 28, 1988.

[11] Sandeep Prasada and Steven Pinker. Generalization of regular and irregular morphological patterns. *Cognition*, 45(3), 1992.

[12] D Rumelhart and J.L. McClelland. On learning the past tenses of English verbs. In *Parallel Distributed Processing: Exploration in the microstructure of cognition.* MIT Press, 1986.

[13] David Waltz. Understanding line drawings of scenes with shadows. In *The Psychology of Computer Vision.* McGraw-Hill, 1975.

[14] Patrick Winston. Learning structural descriptions from examples. In *The Psychology of Computer Vision.* McGraw-Hill, 1975.