Machine Learning 2: 9–38, 1987
© 1987 Kluwer Academic Publishers, Boston – Manufactured in The Netherlands

Learning Syntax by Automata Induction

ROBERT C. BERWICK (BERWICK%MIT-OZ@MIT-MC.ARPA)
MIT Artificial Intelligence Laboratory, 545 Technology Square, Cambridge, Massachusetts 02139, U.S.A.

SAM PILATO
Brattle Research Corporation, 55 Wheeler Street, Cambridge, Massachusetts 02138, U.S.A.

(Received: October 27, 1986)
(Revised: January 15, 1987)

Keywords: Formal inductive inference, language acquisition, automata theory

Abstract. In this paper we propose an explicit computer model for learning natural language syntax based on Angluin's (1982) efficient induction algorithms, using a complete corpus of grammatical example sentences. We use these results to show how inductive inference methods may be applied to learn substantial, coherent subparts of at least one natural language – English – that are not susceptible to the kinds of learning envisioned in linguistic theory. As two concrete case studies, we show how to learn English auxiliary verb sequences (such as *could be taking*, *will have been taking*) and the sequences of articles and adjectives that appear before noun phrases (such as *the very old big deer*). Both systems can be acquired in a computationally feasible amount of time using either positive examples, or, in an incremental mode, with implicit negative examples (examples outside a finite corpus are considered to be negative examples). As far as we know, this is the first computer procedure that learns a full-scale range of noun subclasses and noun phrase structure. The generalizations and the time required for acquisition match our knowledge of child language acquisition for these two cases. More importantly, these results show that just where linguistic theories admit to highly irregular subportions, we can apply efficient automata-theoretic learning algorithms. Since the algorithm works only for fragments of language syntax, we do not believe that it suffices for all of language acquisition. Rather, we would claim that language acquisition is nonuniform and susceptible to a variety of acquisition strategies; this algorithm may be one these.

1. Introduction: The role of inductive inference in language acquisition

As a sophisticated cognitive faculty, language acquisition poses an acid test for any learning theory. How are children able to learn language so fluently and effortlessly, without evident explicit instruction? Broadly speaking, researchers have proposed two ways to attack this problem: first, assume that the child has a rather sophisticated inductive inference engine that can infer the required rules and representations (whatever those may be); second, assume that the grammar hypothesis space is small, so that no sophisticated inference is required. Unfortunately, these differing perspectives have rarely been combined; too often, the results and techniques of one group have been downplayed by others.

For example, many linguists, starting from a position first outlined by Chomsky in the early 1960s, see their job as delimiting the class of natural grammars so narrowly that there really need not be any learning theory for natural language at all – something as simple as hypothesis-and-test would do the job, at least in principle.[1] Such "no learning theory needed" views have often been criticized as ignoring the actual time course of acquisition, computational demands, and the need for some kind of induction (if only the induction of word classes).

On the other hand, researchers concerned with the detailed developmental course of acquisition (MacWhinney, 1982; Langley, 1982) have assumed a larger role for mechanical inductive inference in language acquisition.

Finally, though a considerable body of mathematical inductive inference techniques have been accumulated, these are not usually applied to natural languages in any detailed way. Either the results supply general "boundary conditions" that apply to all kinds of learning (see Osherson, Stob, & Weinstein, 1986), or else the systems described are applied only to artificial examples (Fu & Booth, 1975) and not to natural languages.[2]

This paper bridges the gap between these three traditions by presenting a polynomial-time computer model that uses recent advances both in inductive inference techniques *and* explicit constraints on natural grammars to learn certain regular (finite-state) syntactic subsystems of English syntax. Using Angluin's (1982) algorithm for the inference of *reversible* automata, we show how substantial syntax subsystems may be learned by examining finite, positive-only example corpuses.[3] We also compare the al-

[1]See Wexler and Culicover (1982) for one explicit formulation of this kind of "simple" learning procedure.

[2]This is not universally so. Olivier (1968) used a statistical data compression algorithm to pinpoint word boundaries by hunting for commonly occurring token clusters, such as 't h e' (which are much more common than 't t h'). The same statistical regularity approach was exploited in Wolff's SNPR system (1978, 1982). SNPR also contained a general substitution class algorithm that is less constrained than the one that we present below.

[3]Since examples outside the finite corpus are assumed to be negative examples, the algorithm in fact uses implicit negative evidence. The inference algorithm can operate incrementally *only* after we have restricted the space of target grammars it can consider as valid hypotheses. This restriction can be built in only after we have determined the right hypothesis space restriction. Exact details of the acquisition procedure are given in section 2.4. Each corpus contains on the order of 100 to 500 distinct sentence types,

gorithm's performance to what is known about child acquisition, and find a rough correlation.[4] (As always, it is sometimes difficult to assess the child acquisition data here, so the usual caveats apply.)

More importantly, whatever the particular outcome of these case studies, our results suggest that finite-state natural language subsystems can be learned by general induction procedures, provided those procedures are coupled with restrictive computational constraints. On this view, there *is* a role for machine learning theory in natural language acquisition, but it must be woven together carefully with what is known about constraints on natural grammars. In fact, grammatical constraints and inductive inference appear to work hand in hand: in our case studies, just where the grammar hypothesis space becomes enormously large (the number of possible automata with n states for $n > 20$ is huge), inductive inference techniques may be applied because the associated grammatical subsystem is susceptible to efficient induction techniques.[5] This suggests that there may be general constraints on the design of certain linguistic subsystems to make them easy to learn by general inductive inference methods.

The remainder of this paper is organized as follows. The next two subsections discuss the limits on grammatical regularities as a source of hypothesis space restrictions and computational limits on inductive inference algorithms. Section 2 continues with an informal description of Angluin's inductive inference algorithm, as applied to natural languages. It also formally describes the Angluin inference algorithm. Section 3 applies that algorithm to two case studies, the English auxiliary verb system and noun phrase specifiers (material that precedes the head of a noun, e.g., *the big blue* in *the big blue ball*). Section 4 evaluates our case study results and probes more deeply into how linguistic and inductive inference constraints may be combined.

1.1 The limits of grammatical regularities

Many linguists adopt the view that natural language syntax is regular enough to be learned by simple positive examples, without explicit instruction. But this is really an empirical issue. There are many subportions of a language's syntax that do not demand a powerful inference engine. As an example, the *complements* of most phrases are highly systematic, as one can see from the English examples given below. A complement is simply

and we are currently conducting experiments with even larger corpuses.

[4]We have drawn primarily on the discussion in Pinker (1984) here, though recently we have examined a computer data base from the Brown (1973) corpus.

[5]In general, it is possible to show that *each* of these hypotheses must be explicitly considered, as Gold's 1967 and 1978 results imply. Therefore, "summarization" methods, like Mitchell's (1978) version space algorithm, would become unwieldy.

the phrasal sequence that follows the *Head* of a phrase, e.g., the Noun in a noun phrase (NP), the verb in a verb phrase (VP), and so forth. As a simple example, in *John hit the ball against the fence*, *hit the ball against the fence* is the verb phrase, and *the ball against the fence* is the complement of the verb *hit*. It is made up of two separate phrases: the object noun phrase *the ball* and the prepositional phrase *against the fence*. More generally, the following array of possibilities is permitted in English:

Verb Phrase →	Verb	Noun Phrase (Prep Phrase)* (Sentence)
Prep Phrase →	Preposition	Noun Phrase (Prep Phrase)* (Sentence)
Adjective Phrase →	Adjective	Phrase (Prep Phrase)* (Sentence)
Noun Phrase →	Noun	(Prep Phrase)* (Sentence)

Evidently, within a language like English, all complement phrases can be expressed via the schema XP→X NP PP* (S). That is, they obey the format *Head-Complement*, where *Head* is a metavariable replaceable by verb, preposition, adjective, noun, and where *Complement* is a metavariable replaceable by NP PP*(S). Note that the Complement roughly denotes the "arguments" of the Head.[6] If all natural grammars have this structure, then what the child must learn is quite trivial: since the order of complement phrases is fixed, the only decision to be made is whether the head comes first (as it does in English, French, or Italian) or last (as it does in Japanese). If word classes are known, this evidence is readily available from simple sentences.[7]

While this is not the whole story of phrasal acquisition, many linguistic authors (Lightfoot, 1982) have noted that it goes a long way to explaining the relative rapidity and error-free acquisition of basic phrase structure among children. This highly constrained structure can be exploited in computer models of language acquisition, as Berwick (1982, 1985) has shown.

We may contrast the extreme regularity of phrasal complements with so-called *specifiers*, the material that precedes a phrasal Head. For instance, in the following examples the specifiers are italicized: *a very big* deer; *a*

[6]Of course, there are additional factors that intervene to complicate this simple picture: *NP* may be replaced by 2 or more NPs, depending on a verb's type; and NP must not be present if the head is an adjective or a noun. These constraints arise essentially from the *case marking* properties of the language; this is not a part of basic phrasal information, but additional constraints that must be learned. For example, in other languages the noun may mark case, and in these languages NP may appear after a head noun.

[7]Continuing the point made in the previous footnote, we note that in any individual language there may occasionally be variation in this strict Head-first/Head-final division, but presumably the bulk of sentences, and in particular the sentences used for learning, exhibit a clear-cut choice. See Lightfoot (1982) for additional discussion.

few dozen deer; *a great number of* deer; and so forth.

Allowable specifier sequences are highly idiosyncratic within and across languages. Jackendoff (1977) puts it this way:

> There are problems in studying specifier systems that do not arise in studying complements. First, specifier systems involve very small numbers of lexical items and are riddled with idiosyncrasies. Thus general phrase structure rules must be supported on the basis of impoverished and skewed surface distributions ...
>
> A second problem with specifier systems, at least in English, is that it appears much less possible to correlate semantic regularities with syntactic positions.

It does not appear, then, that learning specifier systems should be as easy as learning phrasal complements – because there are more than one or two binary decisions to make. This would seem to require a powerful kind of inductive inference engine. Indeed, Pinker (1984) has noted that fixing noun subclasses seems to be quite difficult:

> How might the identity of Noun subclasses be established? ... There are several ways this might be done. One could combine two word classes that overlap to some minimum extent ... but this step is treacherous.

On Pinker's view, combining word classes is "treacherous" because of the tremendous number of possibilities involved and because one does not know how to define overlap properly: should two words be substituted if they occur only in exactly the same contexts, or only if their one or two word surrounding contexts are identical?

In the remainder of this paper, we probe exactly this point by examining two specifier systems: first, English auxiliary verbs (such as *could have been won*, which may be loosely regarded as specifiers of the main verb); and second, noun phrase specifiers (such as *three dozen deer*). Despite the "small number" of lexical items involved (see the appendix for the actual sentence sequences), we discovered substantial variation in how difficult these systems are to induce (as measured by the computational complexity of the required inference program), and corresponding variation in the difficulty children have in acquiring these two systems. See section 4.2 for further discussion.

1.2 The limits of formal inductive inference

Having pointed out a natural language domain where simple, linguistically-motivated learning procedures fail, and hence a suitable domain for formal inductive inference, we turn next to the limits of mechanical inductive inference itself. We maintain that unless inductive inference is formally restricted and applied to narrow linguistic domains, it rapidly becomes computationally intractable. This may account in part for the reason that formal mechanical inductive inference techniques have been so rarely applied to full-scale language acquisition studies. Thus, such procedures are bound to limit themselves to simple artificial examples, with at most a few distinct word categories.

To begin, we first note that finite-state inference is in general computationally intractable. Finding an automaton of n states or less agreeing with a given sample of positive and negative data is NP-complete (Gold, 1978).[8] If the number of states is not known at all, then as Gold (1967) showed much earlier, positive examples alone will not suffice.

The reason for this computational difficulty is intuitively clear. If all we know is that a target automaton is a finite-state automaton with n states, then it may take a very long string to distinguish that automaton from all other n-state machines. Indeed, Angluin (1977) shows that in some cases one must look at all $O(2^n)$ strings of length n, where n is the number of states in the target machine.

In order for inference to be computationally feasible, we must restrict the class of target automata to be acquired, just as the linguists have argued; not all finite-state automata can be in the hypothesis space. We propose that the target automata are all deterministic, finite-state, and *k-reversible.* This constraint guarantees an $O(n^3)$ time inference algorithm from positive-only examples (Angluin, 1982), where n is the number of example sentences examined. The associated induction procedure is also incremental; that is, it can process one example sentence at a time, rather than the entire corpus all at once. This constraint is again designed to be consistent with knowledge about human language acquisition. In the remainder of this paper we use an explicit computational model to show that the English auxiliary verb and NP specifier systems meet these constraints, allowing them to be easily inferred from a positive-example corpus.

[8]The problem is NP-complete in the number of states of the target automaton. This problem can be solved in polynomial time if we exhaustively list all strings of length n or less over the assumed alphabet, but only by a coding trick. It is important to remember that the statement of the problem itself *includes* in its encoding the set of all sentences (strings) of length n or less, which in this case will be an exponential amount of data. To get this result, then, we in effect do not "charge" the learning algorithm for the time to read the positive and negative data. If one changed the representation used for characterizing finite-state automata, then it might also be possible to develop more efficient inference algorithms.

2. Learning k-reversible languages from examples

We now introduce the notion of k-reversibility, and follow with a formal definition. The next section presents Angluin's inference algorithm itself. Informally, a *zero-reversible* language supports the simplest kind of word class induction: If you were told that *Mary bakes cakes*, *John bakes cakes*, and *Mary eats pies* are legal strings in some language, and if one then guessed that *John eats pies* is also in that language, then you have assumed that the target language was zero-reversible. The strings mentioned above might been generated by the language expressed by the following regular expression: (Mary|John) (bakes|eats) (cakes|pies).

2.1 The formal definition of 0-reversibility

To formally define when a regular language is reversible, let us first define a *prefix* as any substring (possibly zero-length) that can be found at the very beginning of some legal string in a language, and a *suffix* as any substring (again, possibly zero-length) that can be found at the very end of some legal string in a language. In our case the strings are sequences of words, and the language is the set of all legal sentences in our simplified subset of English.

Also, say that in any legal string the suffix that immediately follows a prefix is a *tail* for that prefix. Then a regular (finite-state) language is *zero-reversible* if, whenever two prefixes in the language have a *single* tail in common, then the two prefixes have *all* tails in common. Put another way, a language is 0-reversible if the automaton recognizing it remains deterministic when one swaps initial and final states and reverses all arcs.[9]

Figure 1 gives some simple examples. The top half (a) shows a non-0-reversible automaton, generating strings such as *walks very very fast*, *walks very very very very fast* and *talks very fast*. The automaton in the bottom half of the figure (b) generates an infinite 0-reversible language.[10] One can see how automaton (a) fails to meet 0-reversibility: the prefixes *talk* and *walk* share the tails *very very fast* in common, but they do not share *all* tails in common, since automaton (a) does not generate *walks very fast*. In contrast, the second automaton in the bottom half of the figure is 0-reversible, as can be easily seen by reversing all the arcs and swapping initial and final states.

[9]Of course, the reversed automaton does not accept the same language as the original; reversing the automaton just tests for a property of the original language.

[10]In fact, automaton (a) is not even k-reversible; see the next subsection for a definition.

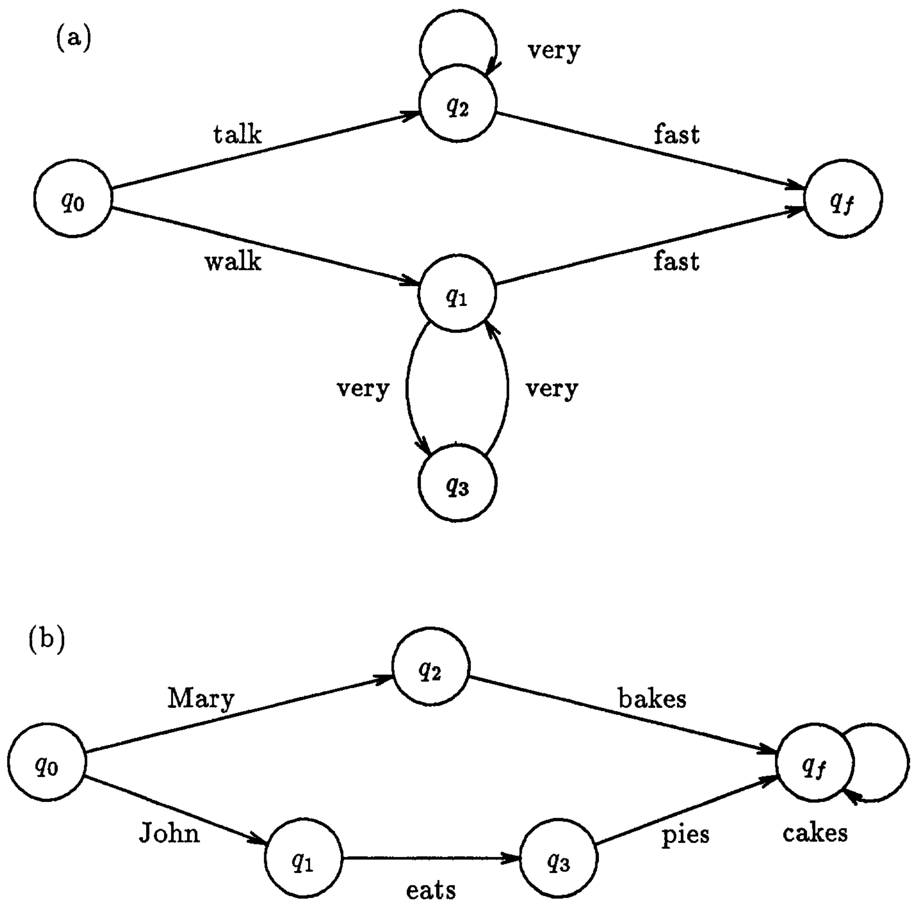

Figure 1. (a) A non-0-reversible automaton; this automaton also happens to be non-k-reversible, for any value of k. (b) A 0-reversible automaton.

2.2 The formal definition of k-reversibility

Intuitively, the extension from 0- to k-reversibility expands the backward context that must be deterministic. A regular language is *k-reversible*, where k is a nonnegative integer, if whenever two prefixes *whose last k words match* have a tail in common, then the two prefixes have all tails in common. In other words, a deterministic finite-state automaton (DFA) is k-reversible if it is still deterministic with lookahead k when its sets of initial and final states are swapped and all of its arcs are reversed. A higher value of k gives more conservative inference, in the sense that it will not overgenerate as readily (because it looks at more possible sentences).[11]

2.3 A simple language example

Before presenting the induction algorithm proper, we will give a simple example showing how the notions of 0- and k-reversibility may be used for inference.

Consider again just the sentences *Mary bakes cakes*; *Mary eats pies*; and *John bakes cakes.* Suppose we assume that the target automata are 0-reversible. (This would be an *a priori* restriction on the class of possible learnable languages, like that made by linguists.) But then, since the language is assumed to be 0-reversible, all prefix tails must be held in common. In particular, the prefix *Mary* has the tail *eats pies*, but the prefix *John* does not. In order to maintain 0-reversibility, the string *John eats pies* must be in the target language. Thus we have inferred a new string, just enough to make the language 0-reversible.

The same idea holds for other values of k. For example, if we assume the target language is 1-reversible, then we must tack on an additional word and see whether *May bakes* and *John bakes* have all tails in common (and hence that the language is 1-reversible). In this case these two-word sequences do have all tails in common (*cakes*), so the three-sentence corpus does not force any additional inference. However, if we now added the sentence *Mary bakes pies*, then we would have to add the sentence *John bakes pies* to the language in order to maintain 1-reversibility. Adding one more sentence, *Mary bakes*, would force us to add *John bakes*, resulting in the seven-string 1-reversible language expressed by (*Mary*|*John*) *bakes* [*cakes*|*pies*] | *Mary eats pies.*

With these same examples, assuming the target is 0-reversible would have produced the regular expression (*Mary*|*John*) (*bakes*|*eats*) (*cakes*|*pies*)*

[11]In the worst case, if we make k as long as all possible sentences, then the procedure cannot overgenerate. Of course, the time required for inference also increases modestly: Angluin's algorithm runs in time $O(kn^3)$.

Table 1. Example of incremental k-reversible inference for several values of k.

Sequence of new strings presented	New strings inferred: $k = 0$	$k = 1$	$k = 2$
Mary bakes cakes	NONE	NONE	NONE
John bakes cakes	NONE	NONE	NONE
Mary eats pies	John eats pies	NONE	NONE
Mary bakes pies	John bakes pies Mary eats cakes John eats cakes	John bakes pies	NONE
Mary bakes	John bakes Mary eats John eats Mary bakes cakes cakes John bakes cakes cakes Mary bakes pies cakes ⋮ (Mary|John)(bakes|eats)(cakes|pies)*	John bakes	NONE

This generates an infinite language, as indicated in the second column of Table 1.[12] On the other hand, assuming that the target language is 2-reversible would force us to add no new sentences. For a particular language we hope to find a k that is small enough to yield some inference. However, k should not be so small that we overgeneralize. Table 1 summarizes our examples of 0-, 1- and 2-reversible inference.[13]

2.4 An inference algorithm

With the definitions of k-reversibility and a simple natural language example behind us, let us consider the inference algorithm itself. In addition to formally characterizing k-reversible languages, Angluin (1982) also developed an algorithm for inferring a k-reversible language from a finite set of positive examples, as well as a method for discovering an appropriate k when negative examples (strings known not to be in the language) are pre-

[12]Because of loops like these, the 0-reversible algorithm may in general be able to infer infinite languages.

[13]Note that since we minimize the resulting automaton, we always obtain the smallest reversible automaton that just covers a corpus.

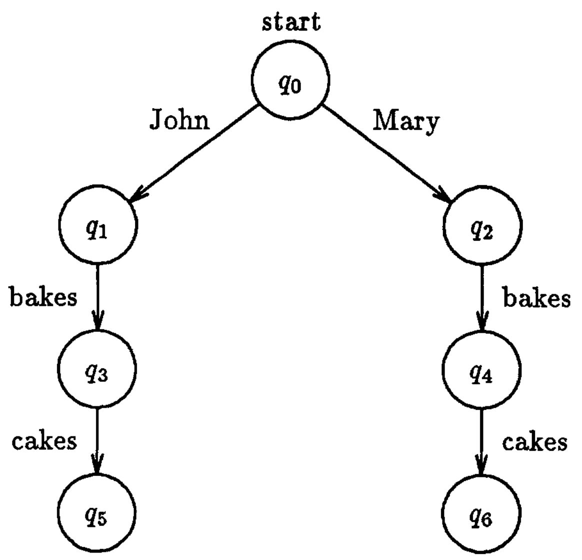

Figure 2. A prefix-tree for some simple sentences.

sented. She also gave an algorithm for determining, given some k-reversible regular language, a minimal set of examples from which the entire language can be induced. We have developed a LISP program that implements this procedure, as well as some refinements on Angluin's incremental acquisition algorithm.

Given a sample of strings taken from the full corpus, we first generate a *prefix-tree* automaton that accepts or generates exactly those strings and no others. As its name implies, a prefix-tree is simply a directed acyclic graph with a single root, where every sentence is "spelled out" by tracing a unique path from the root to the terminal nodes, which are all final automaton states. Figure 2 gives a prefix tree for the sentences *Mary bakes cakes* and *John bakes cakes*.

We now want to add additional strings to maintain a k-reversible language, for some chosen k. The key idea is to collapse equivalent states in the prefix tree, starting from the final states and working backwards, according to the following definition of equivalence:

Let us say that when accepting a string, the last k symbols encountered before arriving at a state is a *k-leader* of that state. Then to generalize the language, we recursively merge any two states for which either of the following two conditions are true:

1. Another state arcs to both states on the same word (this enforces determinism); OR
2. Both states have a common k-leader AND either
   (a) both states are accepting states, OR
   (b) both states arc to a common state via the same word.

When none of these conditions obtains any longer, the resulting DFA accepts or generates the smallest k-reversible language that includes the original sample of strings.[14] This procedure works incrementally. Each new string may be added to the DFA in prefix-tree fashion and the state-merging algorithm repeated. The resulting language induced is independent of the presentation order of sample strings.

Returning to our example in Figure 2, suppose we assume a 0-reversible target automaton. We now work backwards from the bottom of the tree. We first note that both states q_5 and q_6 are final states and that the last 0 symbols (that is, no symbols) before arriving at these states are the same; therefore, we merge these two states. Call this new state q_{56}. Continuing upwards from this newly merged state, we merge states q_3 and q_4 under condition (2) because both arc to the same (new) state q_{56} on the same word and both have the same 0-leader. Finally, we do not merge states q_1 or q_2 because neither of the two conditions is met. This gives us the regular expression [(John|Mary) bakes cakes].

One can determine for what value of k a (finite-state) language is reversible (assuming it is reversible at all) if some negative as well as positive examples are known. One simply tries increasing values of k until the induced language contains no negative examples. In our case studies, we assumed that the learning procedure in effect knows in advance what the appropriate value of k is for a given corpus; this is like the linguists' assumption that the learner knows something about the class of target languages to be acquired. In practice, we carried out this approach by the following procedure that is executed external to the learning algorithm itself: We assume that every sentence outside a particular finite corpus is a negative example. We set $k = 0$ and see whether the resulting DFA covers the corpus and does not generate any negative examples. If so, we are done; if not, we increase k by 1 and try again.[15] Once we know that a particular corpus

[14]This usually is not the smallest DFA for the language; we can minimize the corresponding DFA using standard techniques.

[15]In fact, the learning procedure could use this same method to discover the proper value of k for itself, but only if it has access to some negative examples or a complete set of positive examples and hence, implicitly, negative examples. We discuss this proposal explicitly in section 4.3, since it seems to correspond to what happens in children's acquisition of these syntactic subdomains.

is k-reversible for a specific value of k, we can then impose this as an *a priori* constraint on the class of target automata acquired by the learning system via positive-only examples. As we shall see, for interesting natural language syntax fragments, setting k to 1 or 2 seems most appropriate.

Though the inference algorithm takes a sample and induces a k-reversible language, it is quite helpful to use Angluin's algorithm for going in the reverse direction: given a k-reversible language we can determine a minimal set of shortest possible examples (a "characteristic" or "covering" sample) sufficient for inducing the language. This is helpful in determining a minimal corpus that suffices for acquiring a particular language fragment, or in calculating the "inferential power" of the algorithm; the fewer sentences required to infer the full corpus, the greater the inference power. Though the minimal *number* of examples is of course unique, the set of particular strings in the covering sample is not necessarily unique.

3. Applying formal inductive inference to natural language

As our example corpuses, we aimed to select subportions of English syntax known to be partly regular, yet with some exceptions and variation. There are two well-known examples: auxiliary verbs and noun phrase specifiers. Linguists point out that auxiliary verbs are more regular than noun phrase specifiers (Akmajian, Steele, & Wasow, 1979).

3.1 Learning the English auxiliary system

We represent the English auxiliary system as a corpus of 92 variants of a declarative statement in third person singular. The variants cover all standard legal permutations of tense, aspect, and voice, including *do* support and nine modals. We simply use the surface forms, which are strings of words with no additional information such as syntactic category or root-by-inflection breakdown. For instance, one present, simple, active example is *Judy gives bread*. One modal, perfective, passive variant is *Judy would have been given bread*. It is clear that this corpus does not cover all modals: for example, negatives are not represented, and examples such as *Judy need not bake bread* or *Judy got taken to the bakery* are also omitted. Nonetheless, we feel that this corpus is reasonably representative; in later experiments we plan to expand the range of sentences covered.

We first determined for what values of k the corpus is in fact k-reversible. We found that the English auxiliary system can be faithfully modeled as a 1-reversible regular language. Thus, if a learning system assumes that the target corpus is 1-reversible, it can use the 92 positive examples to learn auxiliary system in time $O(2 \cdot n^3)$ in the number of corpus examples

(Angluin, 1982).[16] If the learning system assumed that the auxiliary system were 0-reversible, it would overgeneralize; the inferred DFA contains loops and so generates infinite numbers of illegal variants.

Figure 3 compares a correct DFA for the English auxiliary system with the 0-reversible, overgeneralized DFA. Both are shown in a minimized, canonical form. The top (correct) automaton can be generated in two ways. First, one can minimize the prefix tree for the full corpus; second, one can minimize the result of k-reversible inference applied to any sufficiently characteristic set of sample sentences, for any $k \geq 1$. One can read off all 92 variants in the language by taking different paths from initial state to final state. The bottom (overgeneralized) automaton is generated by subjecting the first to 0-reversible inference.

Does treating the English auxiliary system as a 1-reversible language yield any inferential power? Making this assumption, the system can in fact infer the entire auxiliary system from a cover of only 48 examples out of the 92 variants in the corpus. Further, if the learning system "knows" that the corpus divides into subportions that can be separately acquired (this might be known on other syntactic or semantic grounds, for example) then additional savings may be won. For instance, if we split up active and passive forms and acquire them separately, the active corpus requires only 38 examples out of 46 and the passive corpus, 28 examples out of 46. Treating the full corpus as a 2-reversible language requires more examples (76), and a 3^+-reversible model cannot infer the corpus from any proper subset whatsoever.

3.2 Learning noun phrase specifiers

Our second example corpus, noun phrase specifiers, is more challenging. No simple, exhaustive covering corpus is directly available. We first used 84 example sentences culled from Jackendoff (1977) as a test case. We then expanded the Jackendoff corpus to 735 example specifier sequences. As far as we know, this is one of the largest and richest subfragments of a natural language ever analyzed by an automatic induction technique. For example, it includes such long sequences as *these very oldest two hundred very big deer*. Still, as we shall note below, even this sample is known to be incomplete and omits some NP specifier sequences; we believe that several thousand sequences are possible. Second, the noun phrase specifier corpus divides into many more "natural" semantic subcomponents than just active/passive; for instance, it might divide into sequences with mass and count nouns, or into sequences with partitives (*of*, as in *some of those deer*)

[16]It will also successfully acquire the auxiliary system if it assumes the target language to be $k+1$ reversible, for $k > 0$.

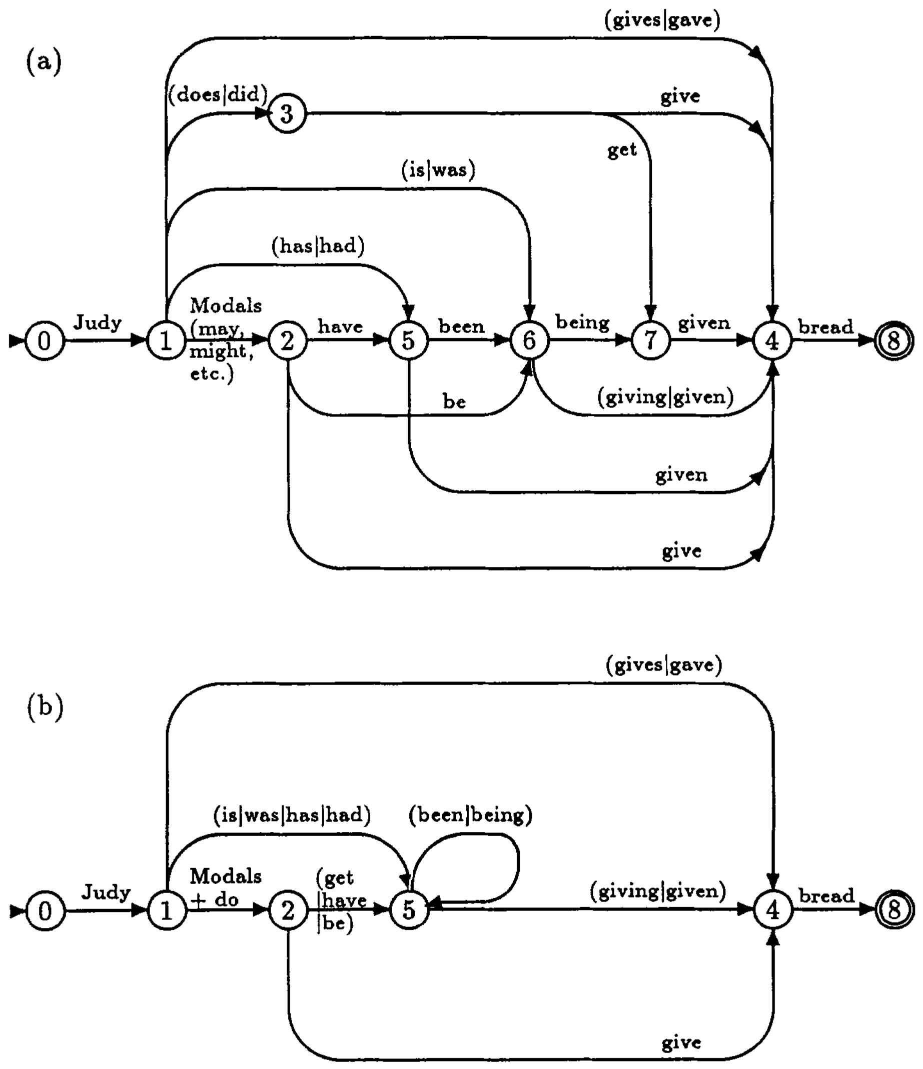

Figure 3. The top automaton (a) generates the English auxiliary system. Zero-reversible inference merges state 3 with state 2 and merges states 7 and 6 with state 5, resulting in the bottom overgeneralized version (b).

and sequences without partitives. Whether these divisions can be exploited remains largely an open question, though it appears that some divisions (see below) can make the system 1-reversible, like the active/passive forms for the auxiliary verbs.

The noun phrase specifier sentences fall into the following rough groups: demonstrative-quantifier-adjectives, such as *those several deer*; article-adjective-cardinal numbers, such as *no big deer* or *any two deer*; quantifier-adjectives, such as *many deer*, *many old deer*; article-adjectives, such as *this deer*, *this big deer*; seminumerals, such as *a score deer*, *only a score deer*; cardinal numerals, such as *these two hundred deer*; intensifiers, such as *very many deer*; superlatives, such as *the oldest deer*; pseudopartitives, such as *a group of some deer*; and a residual collection of eclectic examples.

One can see that the range of these constructions is quite varied, and the restrictions on them quite subtle. For example, one cannot say, *which many several deer*, *enough a deer*, or *many of some deer*. As far as we are aware, this is one of the most complex target languages ever attempted for mechanical inductive inference. Indeed, we did not know beforehand whether the sample would be 1 or 2-reversible at all.

Before using the learning procedure proper, to determine whether the corpus was reversible we again applied our incremental procedure first: we assumed a 0-reversible target language, then checked whether this assumption resulted in a DFA that produced any negative (ungrammatical) example sentences; if so, we proceeded to assume a 1-reversible target language, and so on.

The corpus was not 0-reversible or 1-reversible: both of these automata vastly overgenerated. The inferred 0-reversible automaton had only 3 states, including a loop from state 0 to itself that collapsed together words such as *a*, *all*, *any*, *no*, *these*, and *this*. The 1-reversible automaton had 33 states, and did not contain these loops; however, it still overgenerated, accepting such sequences such as *a score of all deer*. This is because the "window" for 1-reversible inference cannot detect the cooccurrence restriction between *a bunch* or *a score* and *all* when they are separated by *of*.

We found that the NP specifier corpus is 2-reversible, and that it can be inferred from a subset of the full corpus, 359 examples out of 735. This represents considerable inferential power. The resulting full DFA has 81 states (see Table 2), which is considerably larger than the auxiliary system's automaton. Many more states are required to make the requisite fine distinctions between cardinals, seminumerals, quantifiers, partitives (with *of*) and so forth. For instance, in our corpus one can say *a hundred big deer* but not *a number big deer*, so *hundred* and *number* must arc to distinct states, even though they seem otherwise very much alike. Table 2(a) further shows that the inferred automaton does exactly that: *hundred* maps to state 26, while *number* maps to state 27. Similarly, *enough* and *either* are different: one cannot say *enough two deer*, but one can say *either two deer*, so these two words must arc to distinct states (5 and 6), as the table shows. On the other hand, *these* and *those* can be substituted for each other, and the automaton shows them mapping to the same state, 16. Additionally, one cannot have two number-like quantifiers in the same specifier sequence, such as *all several deer*, or *the all deer*, though one can have *all the deer*.

However, a note of caution must be added about the results. It appears that in part the larger number of states has to do with the restricted corpus used. We believe that many of the states from 60 on would be collapsed if we used a full corpus of several thousand distinct examples. In other words, plainly interchangeable states in the table probably result from gaps in our sample data set. Thus, these results are best regarded as tentative, subject to future revision, and serve mainly as a demonstration that a large automaton may be mechanically inferred from a very large corpus. We plan to carry out these even larger experiments shortly; the existing corpus size is unwieldy enough as it stands to demand the full resources of a lisp machine.

The inference system takes twenty times longer to process the larger NP corpus before arriving at a result as compared to the smaller auxiliary corpus – about 30 minutes of execution time compared to a minute and a half. While the procedure is still cubic time in the number of input sentences, this increased processing load is not so unrealistic, given the increased difficulty children have with this richer system (see the next section).

Just as with the auxiliary corpus, the specifier corpus may be split up in certain semantically relevant ways to make learning easier. For instance, the partitives – constructions with *of*, such as *a group of some deer* – may be removed. Like the active/passive split of the auxiliary system, this makes some semantic sense, since the partitives are meaningfully distinct from ordinary specifier sequences and one could argue that a learner can separate partitive from nonpartitive sequences. The resulting smaller data set is nearly 1-reversible; unfortunately, being almost 1-reversible is not good enough. We plan to continue experimenting with semantically defensible partitions of the NP specifier sequences.

A second kind of corpus division – including only article-noun, possessor-noun, or adjective-noun pairs – may be more psychologically relevant. There is considerable evidence that children pass through a two-word stage (Brown, 1973). On the other hand, this restriction may in part be due to memory limitations on the speech production side rather than an accurate reflection of what children actually know; Gleitman and Wanner (1982) summarize this evidence. Therefore, we could legitimately limit the spec-

Table 2 (a). Part of the 2-reversible, minimized DFA for the NP specifier corpus. State 23 is the state reached just before reading the end-of-sentence marker /. State 10 is the state reached just before reading the head noun, *deer*; state 68 is the final state.

STATE	TOKEN	NEXT STATE	STATE	TOKEN	NEXT STATE
q_0	A	q_1	q_5	BIG	q_{10}
	ALL	q_2		DEER	q_{23}
	ANY	q_3		TWO	q_{35}
	EACH	q_4	q_6	BIG	q_{10}
	EITHER	q_5		DEER	q_{23}
	ENOUGH	q_6		VERY	q_{31}
	EVERY	q_7	q_7	BIG	q_{10}
	FRED'S	q_8		DEER	q_{23}
	MANY	q_9		TWO	q_{37}
	MUCH	q_{10}		VERY	q_{31}
	NO	q_{11}	q_8	BIG	q_{10}
	ONE	q_6		DEER	q_{23}
	ONLY	q_{12}		FEW	q_{25}
	SEVERAL	q_{13}		GROUP	q_{22}
	SOME	q_{14}		MANY	q_{32}
	THAT	q_6		ONE	q_{38}
	THE	q_{15}		SCORE	q_{28}
	THESE	q_{16}		SEVERAL	q_{39}
	THIS	q_6		TWO	q_{32}
	THOSE	q_{16}		VERY	q_{40}
	TOO	q_{17}	q_9	BIG	q_{10}
	TWO	q_{18}		DEER	q_{23}
	WE	q_{19}		HUNDRED	q_{26}
	WHICH	q_{20}	q_{10}	DEER	q_{23}
q_1	BIG	q_{21}	q_{11}	BIG	q_{10}
	BUNCH	q_{22}		DEER	q_{23}
	DEER	q_{23}		FEW	q_{10}
	FEW	q_{24}		OLDEST	q_{41}
	GALLON	q_{25}		TWO	q_{42}
	GROUP	q_{22}		VERY	q_{43}
	HUNDRED	q_{26}	q_{12}	A	q_{44}
	NUMBER	q_{27}		THE	q_{45}
	SCORE	q_{28}		TWO	q_{46}
	VERY	q_{29}	q_{13}	BIG	q_{10}
q_2	BIG	q_{10}		DEER	q_{23}
	DEER	q_{23}		HUNDRED	q_{47}
	OLDEST	q_{10}	q_{14}	BIG	q_{10}
	TWO	q_{30}		DEER	q_{23}
	VERY	q_{31}		TWO	q_{46}
q_3	BIG	q_{10}		VERY	q_{31}
	DEER	q_{24}	q_{15}	BIG	q_{21}
	OLDEST	q_{32}		BUNCH	q_{22}
	TWO	q_{30}		DEER	q_{23}
	VERY	q_{31}		FEW	q_{48}
q_4	BIG	q_{10}		GALLON	q_{49}
	DEER	q_{24}		GROUP	q_{22}
	FEW	q_{33}		HUNDRED	q_{26}
	HUNDRED	q_{26}		MANY	q_{50}
	OLDEST	q_{32}		OLDEST	q_{51}
	SEVERAL	q_{34}		ONE	q_{52}
	TWO	q_{35}		SCORE	q_{28}
	VERY	q_{37}		SEVERAL	q_{53}
				TWO	q_{35}
				VERY	q_{36}

Table 2 (b). Second part of the 2-reversible, minimized DFA for the specifier sequences.

STATE	TOKEN	NEXT STATE	STATE	TOKEN	NEXT STATE
q_{16}	DEER	q_{23}	q_{33}	BIG	q_{10}
	FEW	q_{33}		HUNDRED	q_{69}
	MANY	q_{54}		OLDEST	q_{71}
	OLDEST	q_{55}		VERY	q_{36}
	ONE	q_{38}	q_{34}	DEER	q_{23}
	SEVERAL	q_{56}		HUNDRED	q_{47}
	TWO	q_{42}		OLDEST	q_{72}
	VERY	q_{57}		VERY	q_{36}
q_{17}	FEW	q_{26}	q_{35}	BIG	q_{10}
	MANY	q_{32}		DEER	q_{23}
q_{18}	BIG	q_{10}		HUNDRED	q_{59}
	DEER	q_{23}		VERY	q_{29}
	GALLONS	q_{58}	q_{36}	BIG	q_{10}
	VERY	q_{29}		FEW	q_{33}
q_{19}	BIG	q_{10}		OLDEST	q_{74}
	DEER	q_{23}	q_{37}	DEER	q_{23}
	FEW	q_{24}		HUNDRED	q_{59}
	HUNDRED	q_{26}		VERY	q_{29}
	OLDEST	q_{60}	q_{38}	HUNDRED	q_{77}
	ONE	q_{38}	q_{39}	DEER	q_{23}
	SEVERAL	q_{61}		HUNDRED	q_{47}
	TWO	q_{62}	q_{40}	FEW	q_{33}
	VERY	q_{36}		MANY	q_{76}
q_{20}	FEW	q_{63}	q_{41}	BIG	q_{10}
	HUNDRED	q_{64}		DEER	q_{23}
	MANY	q_{65}		TWO	q_{35}
	ONE	q_{38}		VERY	q_{29}
	SEVERAL	q_{39}	q_{42}	BIG	q_{10}
	TWO	q_{46}		DEER	q_{23}
	VERY	q_{66}		HUNDRED	q_{59}
q_{21}	DEER	q_{23}	q_{43}	BIG	q_{10}
	ONE	q_{10}		OLDEST	q_{74}
	TWO	q_{10}			
q_{22}	OF	q_{67}	q_{44}	FEW	q_{25}
q_{23}	/.	q_{68}		HUNDRED	q_{26}
q_{24}	BIG	q_{10}	q_{45}	FEW	q_{49}
	DEER	q_{23}		HUNDRED	q_{26}
	HUNDRED	q_{69}	q_{46}	HUNDRED	q_{59}
	VERY	q_{36}	q_{47}	BIG	q_{10}
q_{25}	OF	q_{58}		DEER	q_{23}
	WATER	q_{23}		FEW	q_{26}
q_{26}	BIG	q_{10}		OLDEST	q_{72}
	DEER	q_{23}		SEVERAL	q_{32}
	VERY	q_{36}		VERY	q_{36}
q_{27}	OF	q_{70}	q_{48}	DEER	q_{24}
q_{28}	OF	q_{10}		HUNDRED	q_{69}
q_{29}	BIG	q_{10}	q_{49}	OF	q_{59}
	FEW	q_{33}	q_{50}	BIG	q_{10}
q_{30}	BIG	q_{10}		DEER	q_{23}
	HUNDRED	q_{59}		FEW	q_{77}
	VERY	q_{29}		HUNDRED	q_{26}
q_{31}	BIG	q_{10}		VERY	q_{29}
q_{32}	BIG	q_{10}			
	DEER	q_{23}			
	VERY	q_{29}			

Table 2 (c). Third part of the 2-reversible, minimized DFA for the specifier sequences.

STATE	TOKEN	NEXT STATE	STATE	TOKEN	NEXT STATE
q_{51}	BIG	q_{10}	q_{64}	BIG	q_{10}
	DEER	q_{23}		DEER	q_{24}
	OF	q_{78}		FEW	q_{26}
	ONE	q_{38}		VERY	q_{36}
	SEVERAL	q_{79}	q_{65}	BIG	q_{10}
	TWO	q_{35}		DEER	q_{23}
	VERY	q_{29}		FEW	q_{78}
q_{52}	BIG	q_{10}		VERY	q_{29}
	DEER	q_{23}	q_{66}	FEW	q_{33}
	HUNDRED	q_{75}	q_{67}	ALL	q_{10}
	VERY	q_{31}		DEER	q_{23}
q_{53}	BIG	q_{10}		SOME	q_{10}
	DEER	q_{23}	q_{68}	final state	
	FEW	q_{77}	q_{69}	BIG	q_{10}
	HUNDRED	q_{47}		DEER	q_{23}
	VERY	q_{29}		FEW	q_{26}
q_{54}	BIG	q_{10}		OLDEST	q_{72}
	DEER	q_{23}		VERY	q_{36}
	HUNDRED	q_{27}	q_{70}	DEER	q_{23}
	VERY	q_{29}		SOME	q_{10}
q_{55}	BIG	q_{10}	q_{71}	BIG	q_{10}
	DEER	q_{23}		DEER	q_{23}
	HUNDRED	q_{26}		FEW	q_{26}
	ONE	q_{38}		HUNDRED	q_{26}
	TWO	q_{35}		ONE	q_{38}
	VERY	q_{29}		TWO	q_{35}
	DEER	q_{23}		VERY	q_{29}
	HUNDRED	q_{47}	q_{72}	BIG	q_{10}
	OLDEST	q_{80}		DEER	q_{23}
	VERY	q_{29}		FEW	q_{26}
q_{57}	FEW	q_{33}		SEVERAL	q_{79}
	MANY	q_{76}		VERY	q_{29}
	OLDEST	q_{74}	q_{73}	OLDEST	q_{74}
q_{58}	WATER	q_{23}	q_{74}	BIG	q_{10}
q_{59}	BIG	q_{10}		DEER	q_{24}
	DEER	q_{23}		HUNDRED	q_{26}
	FEW	q_{26}		ONE	q_{38}
	VERY	q_{36}		SEVERAL	q_{79}
q_{60}	BIG	q_{10}		TWO	q_{35}
	VERY	q_{29}		VERY	q_{29}
q_{61}	BIG	q_{10}	q_{75}	BIG	q_{10}
	DEER	q_{24}		DEER	q_{23}
	FEW	q_{81}		FEW	q_{26}
	HUNDRED	q_{47}		VERY	q_{36}
	VERY	q_{29}	q_{76}	BIG	q_{10}
q_{62}	BIG	q_{10}		DEER	q_{23}
	DEER	q_{23}		HUNDRED	q_{26}
	FEW	q_{26}		OLDEST	q_{80}
	HUNDRED	q_{59}		VERY	q_{29}
	VERY	q_{29}	q_{77}	BIG	q_{10}
q_{63}	BIG	q_{10}		VERY	q_{36}
	DEER	q_{23}	q_{78}	THE	q_{10}
	FEW	q_{77}	q_{79}	BIG	q_{10}
	HUNDRED	q_{69}		DEER	q_{23}
	VERY	q_{36}		HUNDRED	q_{47}
				VERY	q_{29}
			q_{80}	HUNDRED	q_{26}
			q_{81}	HUNDRED	q_{69}

ifier corpus to just two-word sequences, eliminating the partitives and the quantifier sequences altogether, by assuming some kind of processing load that initially filters out multiword sequences from consideration.

This yields a 0-reversible automaton that collapses two-token sequences together: *the*, *big*, *red*, *Fred's* are all put into the same word class. While this overgeneralizes the adult grammar, it matches the child grammar quite closely. A possible developmental sequence would then be to add in the partitive and quantifier sequences, incrementally. The end result would be the 2-reversible machine described above, but one that is built in two distinct stages. As we discuss in the next section, an initial pass over thousands of sample sentences in the Brown corpus data base at least casually confirms this hypothetical developmental sequence. Pinker (1984) also presents evidence in favor of this view.

4. Formal inductive inference in language acquisition

Our two test cases show clearly that formal inductive inference *can* play a role in syntax acquisition. In this section we consider the implications of our tests for models of human syntax acquisition and development.

Our first general observation is that, contrary to some expectations, formal inductive inference need not "run wild" or take extraordinary computational time with full-scale natural language examples. Second, the learning procedure can work incrementally with just the positive examples it is given. Some have objected to applying formal inductive inference to natural language because certain sentences, like the auxiliary verb sequences *could have been being given*, are too rare. But we note that the inference method correctly infers the very longest (and rarest) auxiliary sequences from the shorter ones. It also learns the two-word NP specifier sequences quite easily, while having to work harder at the multiple-word sequences. This apparently accords with human performance (see below).

4.1 Auxiliary system inference: Discussion

The auxiliary system has often been regarded as an acid test for a theory of language acquisition. Given this, we are encouraged that it is in fact learnable via an n^3 method. This success derives from the systematic sequential structure of the English auxiliary system. In an idealized form (ignoring tense and inflections) the regular expression

[DO | [<modal>] [HAVE] [BE]] [BEpassive] GIVE

generates all English verb sequence patterns in our corpus.[17]

[17]The double *be* form – now including tense – shows up in sentences such as *I could be*

Basically, zero-reversible inference attempts to simplify any partial, disjunctive permutation like $(a|b)x \mid ay$ into an exhaustive, combinatorial permutation like $(a|b)(x|y)$. Since except for *do* the active auxiliary verb forms in fact pattern this way, zero-reversible inference almost works for active auxiliary sentences. However, one must move to 1-reversible inference to acquire a correct automaton.

Rather than raising k, one could instead chop the corpus into finer pieces, as briefly mentioned earlier. For example, a more realistic model of processing English verb sequences might have an external, more linguistically motivated mechanism that forces the separate treatment of active and passive forms. If *do* exceptions were recognized as separate forms and the infrequent ... *be being* ... cases were similarly excluded from the immature learner, one could apply simpler and faster zero-reversible inference to the remaining active and passive forms without overgeneralizing.[18] In such a case the active system can be induced from 18 examples out of 44 variants and the passive system from 14 out of 22. The entire active system is learnable once examples of each form of each verb and each modal have been seen, plus one example to fix the relative order of *have* vs. *be*, and one example each to fix the order of modal vs. *have* or *be*.

The ... *be being* ... cases are systematically related to the rest, but also have a natural boundary: as mentioned above, the very rarest sequences like *could have been being given* may be successfully acquired from just a few shorter examples such as *could have been given* and *been being given*, even if the rare sequences are not actually seen. This seems consistent with human judgments that such phrasing is awkward but apparently legal.

4.2 Developmental evidence for reversible inference

How do our results compare with what is known about child language development? Children evidently never make mistakes on the relative order of auxiliaries, which is consistent with the reversibility model, but they do mistakenly combine *do* with tensed verb forms (Pinker, 1984). In contrast, children never make mistakes with auxiliary sequences and modals in straight declarative form; Pinker (1984, p. 272) notes that "no errors with auxiliary ordering have been observed in children's spontaneous speech," but that auxiliary repetition errors are observed in question formation, e.g., *Will it will rain.* No such repetition errors have been reported in declarative sentences (Pinker, 1984, p. 395). We have confirmed these results by

being given bread. These are quite rare, but acceptable.

[18]Given that the appearance of *do* in declarative sentences is also fairly rare, one might prefer a 0-reversible system that handles *do* support as an exception, rather than opt for a 1-reversible inference that is flawless but a slower learner.

examining thousands of examples from one computer data base of child utterances, the Brown corpus.

The NP specifier inference is also supported by developmental evidence. Children rarely or never make mistakes with article-noun, possessor-noun, or adjective-noun combinations, a success supported by the 0-reversibility of the corresponding inferred automaton. For instance, Brown (1973) reports that at what he calls Stage I and Stage II speech, over 30 sentences with adjective/or determiner+Noun combinations were recorded for one child, including 15 examples of article+D Noun and many examples of possessor+Noun (adjective+Noun being less frequent). Examples like *more milk*, *another book*, or *big book* are frequent (Pinker, 1984, p. 149).

Multiword specifier sequences are also produced by age 3 or so, such as *read dat cowboy book* (Pinker, 1984, p. 133); errors occur when children miscategorize words, as in *another one pencil* or *more some milk* (Pinker, 1984, p. 113). This greater difficulty in categorizing multiple word sequences is directly reflected in the non-0-reversibility of the corresponding automaton for such examples: since they are 2-reversible, inference time is correspondingly greater and the window required for correct inference and induction of the word classes is greater. We could argue that computational burden is simply greater in such cases, since 2-reversible inference is required to determine the correct categorization.

As mentioned, we have also examined thousands of examples from the Brown corpus data base ourselves. This initial survey shows fewer than one or two errors in NP specifier order when there are only two words in the specifier sequence (out of about a thousand examples with those specifiers). In contrast, there are many dozens of errors with partitives and quantifiers (out of a few hundred examples). This pattern suggests that human learners might in fact proceed in the tentative way that we did, by boosting k in stages: if the learner first assumed a 0 or 1 reversible target language, they would then get the two-word NP specifier sequences correct, but not the partitives and complex quantifier sequences; they would overgeneralize these.[19] If they were then able to apply a principle of indirect negative evidence, and assume that if some example were not encountered after a certain length of time it must not be a positive example, they could boost k and try again, this time succeeding if k were set to 2. Additional evidence for the validity of this incremental, developmental model of NP specifier acquisition comes from the relative ease with which the auxiliary system is acquired relative to the NP specifier system: since the auxiliary system

[19]Recall that we used this incremental method to first fix the reversibility of the language in question before applying the actual positive-example inference algorithm itself. Here we are suggesting that this incremental procedure might be relevant to human acquisition.

is 1-reversible, it takes less time before an incremental learner would successfully acquire the auxiliary automaton, as compared to the NP specifier automaton.

To summarize, an incremental k-reversible inference model mirrors the ease with which children learn auxiliaries and 2-word NP specifier sequences, as well as the greater problems and miscategorizations they make with multiple-word NP specifier sequences.

4.3 Reversible inference and other language acquisition models

How does our model of k-reversible inference fit into the larger picture of language acquisition? We see it as one way to combine a domain independent inference mechanism with domain dependent constraints, such as a division into auxiliary verbs, active/passive forms, or two-word specifier sequences. In this view, both domain-dependent and domain-independent constraints have their own role to play, and contribute jointly to the success of the acquisition procedure. k-Reversibility guarantees cubic-time learnability, while the language domain itself guarantees that the corpus fragments are k-reversible for small values of k.

We do not believe, however, that k-reversible inference suffices for language acquisition. Rather, we would prefer to claim that language acquisition is nonuniform and susceptible to a variety of acquisition strategies, and that k-reversibility is one of these. We can compare it, in fact, to Pinker's procedure P6 (1984, p. 68), that states in part:

> If 2 [np] expansions are identical except for one position that contains one annotated category in one rule, and another annotated category in the other, collapse expressions by placing the noncommon ones in braces ... if 2 expansions are identical, except for one position that contains one annotated category in one rule and an additional annotated category in the other, collapse expressions by placing the additional annotated category in parentheses.

To unpack this a bit, Pinker's P6 procedure would lead to the following examples of rule collapsing:

$$\begin{array}{llll} \text{Given:} & \text{NP} \rightarrow XY & \text{NP} & \rightarrow UY \\ \text{Yields:} & \text{NP} \rightarrow \left\{ \begin{array}{c} X \\ U \end{array} \right\} Y & & \end{array}$$

$$\begin{array}{ll} \text{Given:} & \text{VP} \rightarrow \text{V NP PP} \mid \text{V NP} \\ \text{Yields:} & \text{VP} \rightarrow \text{V NP (PP)} \end{array}$$

P6 aims at the same kind of similarity collapsing as 1-reversibility, because it collapses identical contexts, but it is not so systematic or formalized. However, P6 does *not* say how word classes would be determined to be equivalent in the first place. Pinker's quote cited earlier on how to fix noun subclasses hints at something very like 1-reversibility. In the end, though, Pinker uses semantic equivalence to establish categories for different verbs or nouns. In contrast, our results show that such regularities, be they syntactic or semantic, *can* be inferred from distributional evidence in highly restricted subdomains, by assuming a k-reversible target class.

Wolff's (1978, 1982) SNPR system also aims at something similar to reversible induction. Given the word sequences *Mary eats pies* and *Mary bakes pies*, Wolff would define a new class X={eats, bakes} corresponding to substitution in identical contexts. However, Wolff's procedure is probably less constrained, since it does not require the target language to be reversible at all.

5. Conclusion

To conclude, k-reversibility is essentially a model of simplicity, not complexity. It basically induces the substitution classes that are the building blocks of larger sentence structures. In the linguistic subdomain for which k-reversibility is defined – regular grammars – it functions to induce the classes that fill "slots" in a regular expression, based on the similarity of tail sets. Increasing the value of k is a way of requiring a higher degree of similarity before calling a match.[20]

When applied to idiosyncratic fragments of English syntax, k-reversible induction is a psychologically plausible and computationally feasible learning procedure. Further, the greater difficulty of NP specifier acquisition implied by its 2-reversibility is mirrored by a corresponding difficulty in child language development. More generally, in at least two areas of English syntax, just where linguistic constraints are so weak that the trivial parameter-learning procedures discussed by linguists fail, the formal constraint of k-reversibility succeeds. Thus, by combining two different kinds of language acquisition mechanisms – the domain-independent model of k-reversibility and the domain-dependent model of syntactic phrase structure – we can arrive at a more powerful overall language acquisition procedure.

[20]See Gonzalez and Thomason (1978) for other approaches to k-tail inference that are not as efficient.

Acknowledgements

Support for this research was provided in part by the Advanced Projects Research Agency of the Department of Defense under Office of Naval Research contract ONR-14-80-C-0505. Marc N. Light greatly assisted us in preparing and programming the NP specifier cases and in examining the Brown corpus. We would also like to thank the reviewers of this paper, including Kurt VanLehn and Pat Langley, whose insightful comments greatly improved this paper's content and presentation.

References

Akmajian, A., Steele, S., & Wasow, T. (1979). The category AUX in universal grammar. *Linguistic Inquiry*, *10*, 1–64.

Angluin, D. (1977). Inductive inference of formal languages from positive data. *Information and Control*, *45*, 117–135.

Angluin, D. (1982). Inference of reversible languages. *Journal of the Association for Computing Machinery*, *29*, 741–765.

Berwick, R. (1982). *Locality principles and the acquisition of syntactic knowledge.* Doctoral dissertation, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA.

Berwick, R. (1985). *The acquisition of syntactic knowledge.* Cambridge, MA: MIT Press.

Brown, R. (1973). *A first language.* Cambridge, MA: Harvard University Press.

Fu, K., & Booth, T. (1975). Grammatical inference: Introduction and survey. *IEEE Transactions on Systems, Man, and Cybernetics*, *5*, 95–111.

Gleitman, L., & Wanner, E. (1982). Language acquisition: The state of the state of the art. In E. Wanner & L. Gleitman (Eds.), *Language acquisition: The state of the art.* New York: Cambridge University Press.

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

Gold, E. M. (1978). Complexity of automaton identification from given data. *Information and Control*, *37*, 302–320.

Gonzalez, R. C., & Thomason, M. G. (1978). *Syntactic pattern recognition.* Reading, MA: Addison-Wesley.

Jackendoff, R. (1977). $\overline{\mathrm{X}}$ *syntax: A study in phrase structure.* Cambridge, MA: MIT Press.

Langley, P. (1982). Language acquisition through error recovery. *Cognition and Brain Theory*, *3*, 211–255.

Lightfoot, D. (1982). *The language lottery.* Cambridge, MA: MIT Press.

MacWhinney, B. (1982). Basic processes in syntactic acquisition. In S.A. Kuczaj, II (Ed.), *Language development: Vol. 1. Syntax and semantics.* Hillsdale, NJ: Lawrence Erlbaum.

Mitchell, T. M. (1978). *Version spaces: An approach to concept learning.* Doctoral dissertation, Department of Electrical Engineering, Stanford University, Stanford, CA.

Olivier, D. (1968). *Stochastic grammars and language acquisition mechanisms.* Doctoral dissertation, Department of Psychology and Social Relations, Harvard University, Cambridge, MA.

Osherson, D., Stob, M., & Weinstein, S. (1985). *Systems that learn.* Cambridge, MA: MIT Press.

Pinker, S. (1984). *Language learnability and language development.* Cambridge, MA: Harvard University Press.

Wexler, K., & Culicover, P. (1982). *Formal principles of language acquisition.* Cambridge, MA: MIT Press.

Wolff, J. G. (1978). Grammar discovery as data compression. In *Proceedings of the AISB/GI Conference on Artificial Intelligence* (pp. 375–379). Hamburg, West Germany.

Wolff, J. G. (1982). Language acquisition, data compression, and generalization. *Language and Communication*, *2*, 57–89.

Appendix: The Auxiliary Data

In this section we list the 92 auxiliary verb specifier sequences used for the automaton induction experiments. The full set of 735 noun phrase specifier sequences is available on request from the first author.

Judy				gives	bread
Judy			is	giving	bread
Judy		has		given	bread
Judy		has	been	giving	bread
Judy				gave	bread
Judy			was	giving	bread
Judy		had		given	bread
Judy		had	been	giving	bread
Judy	does			give	bread
Judy	did			give	bread
Judy	can			give	bread
Judy	can		be	giving	bread
Judy	can	have		given	bread
Judy	can	have	been	giving	bread
Judy	could			give	bread
Judy	could		be	giving	bread
Judy	could	have		given	bread
Judy	could	have	been	giving	bread
Judy	may			give	bread
Judy	may		be	giving	bread
Judy	may	have		given	bread
Judy	may	have	been	giving	bread
Judy	might			give	bread
Judy	might		be	giving	bread
Judy	might	have		given	bread
Judy	might	have	been	giving	bread
Judy	must			give	bread
Judy	must		be	giving	bread
Judy	must	have		given	bread
Judy	must	have	been	giving	bread
Judy	shall			give	bread
Judy	shall		be	giving	bread
Judy	shall	have		given	bread
Judy	shall	have	been	giving	bread

```
Judy   should            give      bread
Judy   should      be    giving    bread
Judy   should have       given     bread
Judy   should have been  giving    bread

Judy   will              give      bread
Judy   will        be    giving    bread
Judy   will   have       given     bread
Judy   will   have been  giving    bread

Judy   would             give      bread
Judy   would       be    giving    bread
Judy   would  have       given     bread
Judy   would  have been  giving    bread

Judy                     is        given     bread
                   is    being     given     bread
Judy          has  been            given     bread
Judy          has  been  being     given     bread

Judy                     was       given     bread
Judy               was   being     given     bread
Judy          had        been      given     bread
Judy          had  been  being     given     bread

Judy   does              get       given     bread
Judy   did               get       given     bread

Judy   can               be        given     bread
Judy   can   be          being     given     bread
Judy   can   have        been      given     bread
Judy   can   have   been being     given     bread

Judy   could             be        given     bread
Judy   could        be   being     given     bread
Judy   could  have       been      given     bread
Judy   could  have  been being     given     bread

Judy   may               be        given     bread
Judy   may          be   being     given     bread
Judy   may    have       been      given     bread
Judy   may    have  been being     given     bread

Judy   might         be  being     given     bread
Judy   might             be        given     bread
Judy   might  have       been      given     bread
Judy   might  have   been being    given     bread
```

Judy	must			be	given	bread
Judy	must		be	being	given	bread
Judy	must	have		been	given	bread
Judy	must	have	been	being	given	bread

Judy	shall			be	given	bread
Judy	shall		be	being	given	bread
Judy	shall	have		been	given	bread
Judy	shall	have	been	being	given	bread

Judy	should			be	given	bread
Judy	should		be	being	given	bread
Judy	should	have		been	given	bread
Judy	should	have	been	being	given	bread

Judy	will			be	given	bread
Judy	will		be	being	given	bread
Judy	will	have		been	given	bread
Judy	will	have	been	being	given	bread

Judy	would			be	given	bread
Judy	would		be	being	given	bread
Judy	would	have		been	given	bread
Judy	would	have	been	being	given	bread