September 1996

Learning First Order Quantifier Denotations An Essay in Semantic Learnability

Robin Clark
University of Pennsylvania, rclark@babel.ling.upenn.edu

University of Pennsylvania Institute for Research in Cognitive Science Technical Report No. IRCS-96-19.

Learning First Order Quantifier Denotations An Essay in Semantic Learnability

Abstract
This paper addresses the problem of how a learner would associate a denotation with a determiner on the basis of the pairing of a simple sentence with some perceptual input. Following work by van Benthem (1985), first order quantification is simulated with (a subclass of) finite state automata. Results from Angluin (1988) are used to demonstrate the learnability of this class.

Comments
University of Pennsylvania Institute for Research in Cognitive Science Technical Report No. IRCS-96-19.

Institute for Research in Cognitive Science

Learning First Order Quantifier Denotations
An Essay in Semantic Learnability

Robin Clark

University of Pennsylvania
3401 Walnut Street, Suite 400A
Philadelphia, PA 19104-6228
September 1996

Site of the NSF Science and Technology Center for Research in Cognitive Science

IRCS Report 96--19

Learning First Order Quantifier Denotations
An Essay in Semantic Learnability

Robin Clark
Department of Linguistics
University of Pennsylvania
Philadelphia, PA 19104
rclark@babel.ling.upenn.edu

1 Introduction

This paper is a first attempt at a formal theory of the development of semantic behavior. We will take as our starting point the acquisition of quantifier denotations. This might, at first blush, seem a curious point of departure. The past two decades have, however, seen as explosive growth in the study of quantifiers and the theory of generalized quantifiers is now sufficiently mature that its learnability properties can be studied. As we shall see, an algorithm exists that will learn the first-order quantifiers; higher order quantifiers, like *most*, will present a greater challenge, however.

It should come as no surprise that the learnability properties of quantified expressions have not been the subject of widespread investigation. What, after all, do quantified expressions refer to? The question has been a matter of debate since at least the scholastic philosophers of the middle ages who took it as problem of the reference of general and particular terms (Loux, 1974). Consider a recent example of the discussion:

> If a class were taken as consisting of its members, there could be no place for a null class in logic; when "nothing" or "no man" stands as a grammatical subject, it is ridiculous to ask what it refers to ... Although it might seem sensible to ask which portion of the class of men is constituted by the men referred to as "all men" or "some men", we may be led to doubt the legitimacy of this question; if we once think of comparing the adjectival uses of "all", "some", "no", and "alone"——"all men laugh, some men laugh, no men laugh, men alone laugh"——we see that none of these has the role of marking out part of a class.

> [...] Phrases like "all men" and "some men" will not on this interpretation have any reference at all; "all" and "some" will be significant not as prefixes to single terms, but as parts of logical frameworks with place for two terms, "All ——— are ———", "Some ———are ———", "Some ——— are not ———"...
>
> —P. T. Geach, *Reference and Generality*

On the above theory, quantified expressions have no reference. Their meaning is acquired from the part that they play in a theory of inference.[1] The above point is made (and lampooned) quite clearly by Lewis Carroll:

> "I see nobody on the road," said Alice.
>
> "I only wish *I* had such eyes," the King remarked in a fretful tone. "To be able to see Nobody! And at that distance too! Why, it's as much as *I* can do to see real people, by this light!"
>
> "Who did you pass on the road?" the King went on, holding out his hand to the Messenger for some more hay.
>
> "Nobody," said the Messenger.
>
> "Quite right," said the King: "this young lady saw him too. So of course Nobody walks slower than you."
>
> "I do my best," the Messenger said in a sullen tone. "I'm sure that nobody walks much faster than I do!"
>
> "He can't do that," said the King, "or else he'd have been here first...."
>
> —Lewis Carroll, *Through the Looking-Glass and What Alice Found There*

The above quote is instructive since the King treats *nobody* as though it referred to an individual, contrary to Geach's admonitions. He then applies an inferential rule to conclude incorrectly that nobody walks slower than the Messenger. Had the King only followed Geach's advice, he could have avoided his error!

Consider, though, the problem of a learner trying to associate meanings with words. According to Geach, the learner must, first, recognize that certain words or phrases do not refer to anything. That is, the learner must avoid the trap of treating *nobody* as though it referred to some object or objects in the world. Instead, the learner must discover the syllogistic patterns that they enter into. Presumably, the Geachian learner observes its caretakers engaging in various sorts of syllogistic reasoning and, then, associates each quantifier with the proper syllogism. This seems a rather unlikely scenario; it presupposes

[1] This point of view has won wide acceptance in generative grammar; see, for example, the discussion of *Quantifier Raising* in May (1985) or Chomsky (1986).

that, once the learner detects a quantifier, it must wait until it observes someone in its environment producing an inferential pattern, a behavior that has a rather low probability of overtly manifesting itself in the normal course of events.

Our goal in this report is to show that there is a reasonable procedure for associating first-order quantifiers with their denotations. Our results relate recent work on generalized quantifiers (Barwise & Cooper, 1981; Keenan & Stavi, 1986; van Benthem, 1986) with work on learning regular sets. We will turn, first, to some general discussion of the problem of learning word meanings. After that, we will consider some properties of generalized quantifiers. The formal study of generalized quantifiers in natural language has led to the interesting result that first order quantifiers can be simulated by finite state automata. We then apply some recent results on the learnability of regular sets to derive the result that first-order quantifiers in natural language are learnable. This result has a general interest, since it provides an interesting computational approach to the study of conceptual development. Whatever one believes about learning—whether learning is the application of a general inductive procedure to raw experience or whether learning is the labeling of innate concepts—quantifier denotations provide an interesting puzzle. Somehow, on the basis of the interaction between experience and inborn cognitive structure, native speakers of English learn that the phonic sequence written "no" has a particular interpretation; providing an answer to how they are able to do this is one of the fundamental challenges that confronts any thoery of the development of language.

2 Word Learning and Perceptual Regularities

One popular theory of word learning holds that the learner associates perceptual regularities with words. In particular, the learner monitors its sense data looking for correlations between its representations and speech. For example, a caretaker might repeatedly present the learner with a cup while uttering the word "cup." Eventually, the learner would come to associate its internal representation of the cup with the word "cup." Similarly, the learner might hear the word "red" while a red patch is present in its sense data; this latter example is deliberately drawn from Quine (1973), but he major proponent of this theory is surely Locke (1690).

The search for perceptual regularities must be constrained by innate mechanisms. Given the number of possible correlations that might exist in the sensory data available to the learner (Goodman, 1979), it would take an enormous number of presentations for the learner to discover just the right set of properties to associate with a word. Consider, also, such basic problems like object recognition; how does the learner come to individuate objects in its sensory data? For example, the learner must come to realize that all those different patterns of light hitting its retina is, in fact, different presentations of the family cat. The learner must also be able to detect properties of objects, even when the objects that have some property are otherwise quite divers. Consider how the learner would learn *round*. The learner might, for example, be told at various times that a pencil, a basketball,

a face, an eye, an egg and a football are all round. The problem becomes worse if we take into account properties of properties of objects like *sweet*, which might hold of various tastes and dispositions.

There is some evidence that this approach works for concrete nouns. The naive view of verb learning fails for verbs (as shown by Gleitman). Finally, as is easy to imagine, the sensory approach seems to be quite hopeless for quantifier denotations. Consider the following problems:

(1) a. What perceptual regularities correlate with *no* or *nobody*?
b. What perceptual regularities correlate with *every*?
c. What perceptual regularities correlate with *at least 10*?

The problems in (1) are reminiscent of the problems that the scholastic philosophers found in considering the reference of terms. Does *every man* have some sort of strange distributed reference or does it refer to some sort of abstract particular in an inaccessible Platonic paradise? In any event, it doesn't seem as though *every man* could refer to anything accessible in the learner's perceptual world. If this is so, then, how does the learner come to associate a meaning with quantified noun phrases?

2.1 Syntactic Bootstrapping

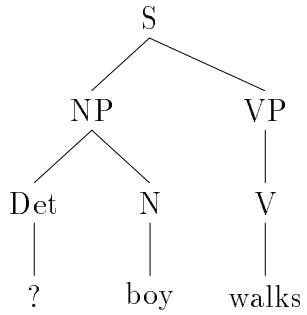
Since the idea that the learner acquires word meanings by associate perceptual regularities with phonological representations fails for quantifier meanings, we must try to take an entirely different approach to the problem. The main competitor to the semantic approach outlined above is *syntactic bootstrapping*. This is the theory that the learner uses information about syntactic distributions of words to break into their semantics. For example, semantic verb classes can be discovered exploiting innate correspondence rules between syntactic structures and semantic classes. Thus, the learner would class monotransitives into a broad semantic group, distinct from ditransitives. The theory is put quite succintly in the following:

> The child who understands the mapping rules for semantics onto syntax can use the observed syntactic structures as evidence for deducing the meanings. The learner observes the real-world situation but also observes the structures in which various words appear in the speech of the caretakers. Such an approach can succeed because, if the syntactic structures are truly correlated with the meanings, the range of structures will be informative for deducing which word goes with which concept.
>
> —Lila Gleitman, "The structural sources of verb meanings"

After identifying broad semantic classes, the learner could use other information to refine these sets into subgroupings. Syntactic bootstrapping would, at very least, provide the learner with a way into the problem of discovering word meanings.

The syntactic bootstrapping account can correctly identify determiners as a semantic class. Suppose, for example, that the learner has associated *boy* with a set of entities and *walk* as a predicate on entities. Then the following provides a syntactic frame for some determiners:

(2)



That is, the learner could use the above frame to determine that *every*, *no*, *some* and so on, are functions that map noun denotations onto elements that combine with VPs to yield sentences. This is type theoretically correct; if common nouns are of type $\langle e, t\rangle$ (functions from entities to truth values) and verb phrases are of the same type (functions that are true or false of entities), then determiners map noun denotations onto functions from VP denotations to truth values:

(3) a. $\langle\langle e, t\rangle, \langle\langle e, t\rangle, t\rangle\rangle$
b. $\lambda P \lambda Q \forall x(P(x) \rightarrow Q(x))$

(3)a shows the type theoretic analysis of determiners as functions from nouns denotations to functions from VP denotations to truth values. (3)b shows a simplified montagovian analysis of *every* (I have suppressed intentional operators); suppose that the semantic translation of the common noun *boy*, which is of type $\langle e, t\rangle$ is given to the function in (3)b the result is:

(4) $\lambda Q \forall x(\text{BOY}(x) \rightarrow Q(x))$

Feeding the denotation of *walks*, which is also of type $\langle e, t\rangle$, to the above function gives:

(5) $\forall x(\text{BOY}(x) \rightarrow \text{WALK}(x))$

which is a fair approximation of the sentence *every boy walks*. Thus, the syntactic bootstrapping theory has the virtue of taking us to the correct semantic class of functions for determiner denotations. But how could the learner distinguish between elements in this class? For example, *no* and *the* have virtually the same syntactic distribution, but they surely mean very different things. To crack this puzzle, we will need to consider the theory of generalized quantifiers.

3 Generalized Quantifiers

Let us begin our review of generalized quantifier theory with a basic case, a subject and a predicate as in (6):

(6) Every somnambulist is a philatelist.

The quantifier *every* in (6) can be taken as a relation between two 1-place prediactes, the property of being a *somnambulist* and the property of being a *philatelist*, as shown in (7):

(7) every($\{x \mid x$ is a somnambulist$\}, \{y \mid y$ is a philatelist$\}$)

The truth conditions for *every* can be given as show in (8); in particular, *every* holds of two predicates P and Q, every(P, Q), just in case P is a subset of Q:

$$(8) \qquad \text{every}(X, Y) = \begin{cases} 1 & \text{iff } X \subseteq Y \\ 0 & \text{otherwise} \end{cases}$$

Given that 1-place predicates are of the type $\langle e, t\rangle$, then every is a function that maps two 1-place predicates to a truth value. If we express *every* in terms of the lambda calculus:

(9) $\lambda P \lambda Q \text{every}(P, Q)$

then, in terms of type theory, it is of type $\langle\langle e, t\rangle, \langle\langle e, t\rangle, t\rangle\rangle$, just as in Montague's analysis of quantifiers (see (3)b, above).

More generally, we might analyze a generalized quantifier, Quant, as expressed syntactically in a simple predicational structure like:

(10) Quant P(s) is/are Q.

as a function between two 1-place predicates:

(11) Quant$(\{x \mid x \in P\}, \{y \mid y \in Q\})$

The truth conditions of the quantifier Quant will generally be of the form:

$$(12) \qquad \text{Quant}(P, Q) = \begin{cases} 1 & \text{iff } f(P, Q) \\ 0 & \text{otherwise} \end{cases}$$

where f is some relation on the sets P and Q.

From what we have said so far, f can be any function we like between the two sets P and Q. How many such functions between sets exist? P and Q are subsets on the domain of discourse, D; thus they are elements of the power set of D and there are $2^{|D|}$ such sets, where $|D|$ is the cardinality of D. There are, then, $2^{|D|}$ choices for the first argument of f and the same number for the second argument. Since f is a mapping from pairs of sets to truth values, there are are $2^{4^{|D|}}$ such functions. For even a small domain, this leads to a dazzlingly large number of possible quantifier denotations. Thus, it is of great interest to the linguistic theory, as well as to a general theory of learning, to discover

natural constraints on quantifier denotations. These constraints would serve to limit what a possible quantifier denotation could be in a natural language. We turn to this topic in next.

3.1 Constraints on Quantifier Denotations

We begin with a well-known constraint, *conservativity*, formulated most clearly in Keenan & Stavi (1986). A form of conservativity can also be found in Barwise & Cooper (1981) in the guise of the requirement that a quantifier must *live on* its first argument. A simple formulation of the conservativity constraint is given in (13):

(13) Conservativity
$\text{Quan}(P, Q)$ iff $\text{Quan}(P, P \cap Q)$

Conservativity requires that a natural language quantifier, Quan, holds between two sets P and Q if and only if Quan also holds between P and the intersection of P and Q. This requirement entails that the following two sentences are paraphrases of each other:

(14) a. Every somnambulist is a philatelist.
b. every somnambulist is both a somnambulist and a philatelist.

.
It should be noted that (15) as been put forward as a counterexample to conservativity:

(15) Only frogs croak.

Notice that the entire set of croaking things must be considered in determining the truth of (15). If I restrict myself to the intersection of the frogs and the croaking things, I will get the truth conditions of (15) wrong since, crucially, if a non-frog croaks then (15) is false; it is sufficient to find a single non-frog among the croakers. One might treat *only* as having the odd property of being conservative in its second argument. In particular:

(16) $$\text{only}(P, Q) = \begin{cases} 1 & \text{iff } Q - P = \emptyset \\ 0 & \text{otherwise} \end{cases}$$

For present purposes, I will set aside a detailed analysis of *only* as tangential to our main goal of establishing an algorithm that will acquire first order quantifier denotations.

Conservativity places a strong constraint on the class of functions in which determiners can take their denotation. Conservativity reduces the number of possible determiner denotations from $2^{4^{|D|}}$ to $2^{3^{|D|}}$ (Keenan & Stavi (1986); Keenan & Faltz (1985); van Benthem (1986)). Indeed, conservativity catures a general sense that a generalized quantifier, $\text{Quan}(P, Q)$ is somehow about P. It fails to fully characterize this intuition, as we shall see. In what follows, we will let D denote the domain of discourse and let $\text{Quan}_D(P, Q)$ denote a generalized quantifier on D. It seems sensible to say that once P and Q have been fixed then adding new elements to the universe does not change the interpretation of $\text{Quan}(P, Q)$. This principle can be stated as:

(17) Extension
If $P, Q \subseteq D \subseteq D'$, then $\text{Quan}_D(P, Q)$ iff $\text{Quan}_{D'}(P, Q)$

Conservativity and extension are closely related principles that are intended to capture the intuition that $\text{Quan}(P, Q)$ is somehow about P and Q. At first glance, one might think that extension could follow from conservativity, but this is not the case. Consider, in particular the definition of the following quantifier, G:

(18) $$G(P, Q) = \begin{cases} 1 & \text{iff } |P \cap Q| \geq |D - (P \cap Q)| \\ 0 & \text{otherwise} \end{cases}$$

By definition, G is conservative; it requires that at least have of the elements of D be both P and Q. Notice, however, that adding new elements to the domain can change the truth of $G(P, Q)$. It is possible to construct of a domain D' with the property that $D \subset D'$ and $G_D(P, Q)$ is true but $G_{D'}(P, Q)$ is false, violating extension.

A final constraint which will be important for our present purposes is that of *quantity* as defined in (19):

(19) Quantity
$\text{Quan}_D(P, Q)$ depends only on the numbers of individuals in P, $P \cap Q$, Q and D.

Formally, (19) disallows reference to hidden sets in the definition of a quantifier. In essence, we claim that the truth of $\text{Quan}_D(P, Q)$ can be determined by inspection of P, $P \cap Q$ and, possibly, D. For our purposes, the truth of many expressions containing common natural language quantifiers can be established without ever having to look outside the set P. Suppose, for example, that I had the power to assemble all the somnambulists before me in a field and I was asked to verify that every somnambulist is a philatelist. Conservativity, extension and quantity together mean that I will never need to look beyond the residents of the field to verify that no somnambulist is a non-philatelist.

3.2 The Tree of Numbers and Finite State Automata

Following van Benthem (1986), we will follow an alternative, graph-oriented approach to thinking about quantification. Conservativity, quantity and extension guarantee that any natural language quantifier, $Quan(P, Q)$, is equivalent to a set of couples of cardinalities, (m, n) as defined in (20):

(20) (m, n) where $m = |P - Q|, n = |P \cap Q|$

Imagine, now, the ordered pairs defined in (20) arrayed as shown in figure 1; that is, with the pair $(0, 0)$ at the apex and, for any pair (i, j), the pair $(i + 1, j)$ is written below it and to the left, while the pair $(i, j + 1)$ is written below (i, j) and to its right. This is, of course, the upper right hand quadrant of a cartesian coordinate system.

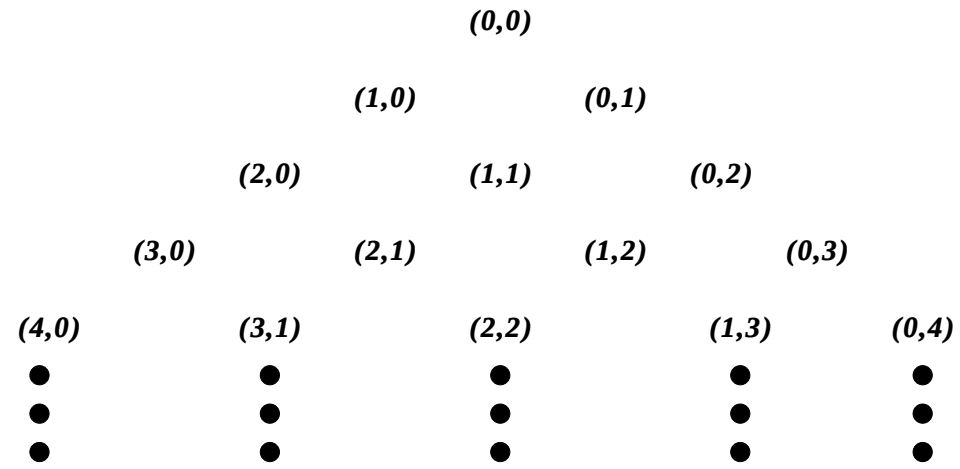
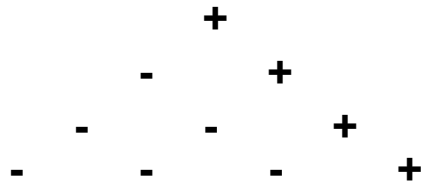


Figure 1: The Tree of Numbers

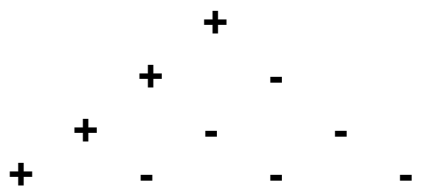
Each point in the quadrant, then, corresponds to a situation where $P - Q$ and $P \cap Q$ have particular cardinalities. For example, $(0, 2)$ corresponds to the case where there are two elements in $P \cap Q$ and nothing in $P - Q$. In this case, all the Ps are Q; loosely speaking, we can say that $(0, 2)$ is in the denotation of every(P, Q). Suppose we replace every pair that is in every(P, Q) by the symbol '+' and every pair that is not in every(P, Q) by '−'. We get the following graph for every(P, Q):

(21) *every:*



That is, every point in the right-descending spine of the tree is marked '+' and everything else is marked '−'. Equally, *no* is the mirror image of *every*:

(22) *no:*

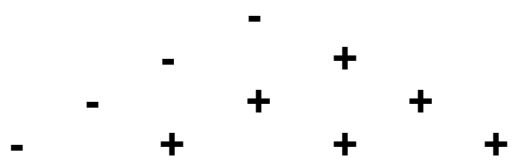


A point is contained in no(P, Q) just in case it is in $P - Q$ and not in $P \cap Q$. Thus, the left-desceding spine of the tree consists of points marked by '+' with all other points marked '−'. The mirror symmetry between *every* and *no* is unsurprising given their status

as duals.

Consider, next, some(P, Q). A point in the tree is contained in some(P, Q) just in case $n > 0$ in (m, n); this corresponds to the case where $P \cap Q$ is non-null:

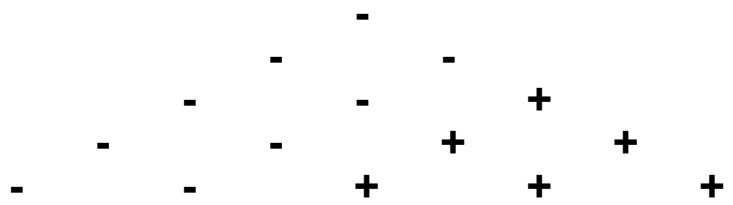
(23) *some:*



Notice that the tree in (23) can be constructed from the tree in (22) by mapping '+' in (22) to '−' in (23) and mapping '−' to '+'. This is as expected given that *some* is the complement of *no*.

Finally, let us take the case of a numeric quantifier like *at least two*. The generalized quantifier, at-least-2(P, Q) holds if $|P \cap Q| \geq 2$. In terms of the tree of numbers, this means that a point (m, n) is contained in at-least-2(P, Q) just in case $n \geq 2$. Marking points contained in at-least-2(P, Q) with '+', as usual, gives the following graph:

(24) *at least two:*



Notice that the tree in (24) is like the tree in (23) except that the triangle of '+'s has been shifted down and to the right. In general, *at least n* creates a downward triangle of +'s rooted in $(0, n)$.

The tree of numbers graphically presents a number of interesting properties of first-order generalized quantifiers. The reader is referred to van Benthem (1986) for a more thorough-going discussion of the tree of numbers and generalized quantifiers. For our purposes, it is important to note that any first order generalized quantifier can be represented by a regular pattern of +'s and −'s. In fact, the pattern associated with any first order quantifier is finite in the sense that there is a finite upper triangle in the tree of numbers that can be used to generate the entire infinite pattern. To show this, we must appeal to a result from finite model theory.

As a point of notation let us define a relation on sets X and Y, $X \sim_n Y$, as follows:

(25) $X \sim_n Y$ if either $|X| = |Y| = k < n$, or $|X|, |Y| \geq n$.

That is, X and Y are indistinguishable if the have the same cardinality or if their cardinalities both exceed some threshold k.

(26) **THEOREM.** On finite models, a quantifier Quan is first-order definable if and only if, for some fixed n, $\langle D, P, Q\rangle \sim_n \langle D', P', Q'\rangle$ implies $\text{Quan}_D(P, Q)$ iff $\text{Quan}_{D'}(P', Q')$.

The above theorem can be interpreted as saying that first-order quantifiers can only distinguish models up to a certain point; once the cardinalities of D and D' exceed some number n, for example, then first-order quantifiers can no longer distinguish between D and D'.

In terms of the tree of numbers, this means that there is a finite top triangle that completely specifies the behavior of the quantifier. In particular, for each first-order quantifier there will be a threshold line at $a + b = 2n$ in the tree of numbers, following van Benthem (1986) let us refer to it as the "Fraïssé threshold", with the following properties:

(27) The Geometric Interpretation of the Fraïssé Threshold:

(i) the truth value at (n, n) determines the value of its generated downward triangle.

(ii) the truth values at $(n + k, n - k)$ are propagated along their downward left lines.

(iii) the truth values at $(n - k, n + k)$ are propagated along their downward right lines.

I can define a first order quantifier simple by giving the pattern of $+$'s and $-$'s for the top triangle, up to the Fraïssé threshold.

This "geometric" result has an interesting an interesting interpretation with respect to automata theory, since it allows us to simulate any first-order quantifier by a finite state automaton. Recall that a finite state automaton consists of a finite set of states and a set of transitions from state to state that occur on input symbols chosen from an alphabet Σ. The formal definition of a finite state automaton is given in (28):

(28) A finite state automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where:

1. Q is a finite set of *states*.
2. Σ is a finite input alphabet.
3. $q_0 \in Q$ is the *initial state*.
4. δ is a *transition function* mapping $Q \times \Sigma$ to Q.
5. $F \subseteq Q$ is a set of *final states*.

Intuitively, finite state automata are extremely simple, memoryless devices which accept strings drawn from a language; when in a particular state, q_i, they will move to some new state, q_{i+1}, when presented with a symbol α drawn from Σ just in case $\delta(q_i, \alpha) = q_{i+1}$.

Having entered a new state, the device retains no memory of what went before. The set of languages accepted by finite set automata are referred to as the regular sets.

The fact that first order quantifiers can be simulated by a device as simple as a finite state automaton is suggested by comparing the Fraïssé threshold and the following "pumping lemma" for the regular sets:

(29) **The Pumping Lemma for Regular Sets**
THEOREM. Let L be a regular set. Then there is a constant n such that if z is a word in L and $|z| \geq n$, we may write $z = uvw$ in such a way that $|uv| \leq n$, $|v| \geq 1$ and for all $i \geq 0$, uv^iw is in L. Furthermore, n is no greater that the number of states of the smallest finite state automaton accepting L.

We can think of the relationship between regular sets and first-order quantifiers as follows: each string is a model which could either be accepted or rejected by $\text{Quant}(P, Q)$. In (29), the string v can be thought of as "witness" for $L = \text{Quan}(P, Q)$ and the number n is the Fraïssé threshold.

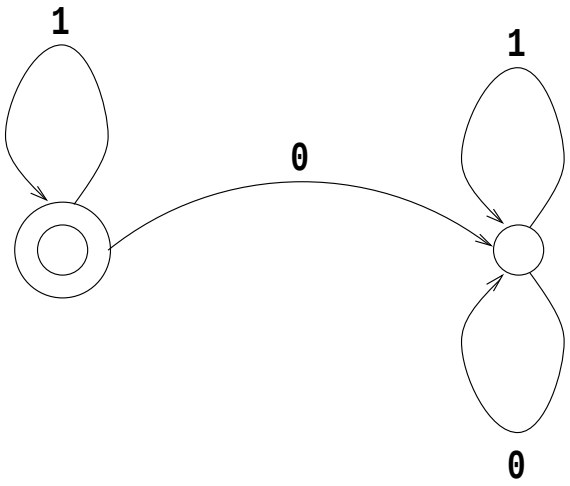
Let us be a bit more precise about how to construct a "semantic automaton". The results from finite model theory show that a first-order quantifier can be represented by a finite upper triangle of the tree of numbers. We can give a procedure for mapping from models to strings in a regular language, as suggested by our discussion of the pumping lemma for regular sets in (29). Suppose we are given a quantifier $\text{Quan}_D(P, Q)$ defined on domain D over sets P and Q. To construct a string in the regular language:

(30) a. Place the elements of P in a sequence. The sequence can be determined at random.
b. Let elements of the set $P \cap Q$ be denoted by '1'.
c. Let elements of the set $P - Q$ be denoted by '0'.

$\text{Quan}_D(P, Q)$ a set of strings of 0s and 1s where each element in P can be replaced by 1 if it is in $P \cap Q$ and 0 if it is in $P - Q$. Recall that conservativity, extension and quantity guarantee that we don't need to look outside of P to verify the truth of $\text{Quan}_D(P, Q)$.

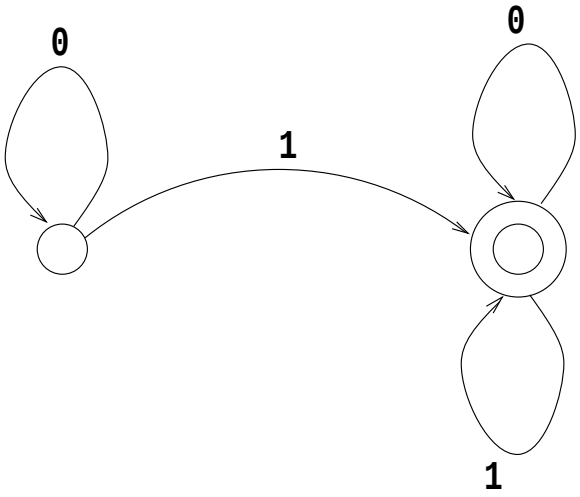
Consider, now, the following automaton:

(31) *all, every:*



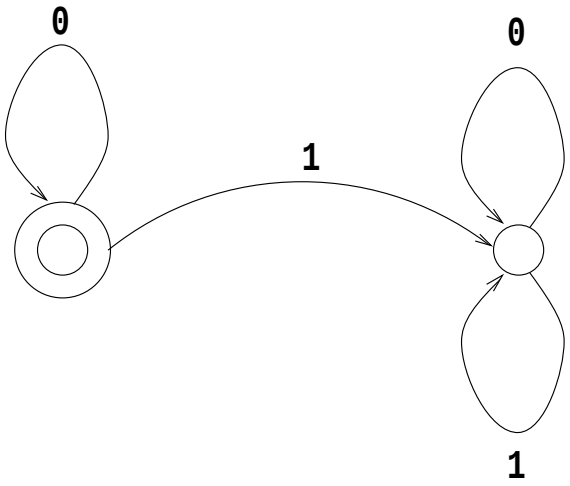
By convention, the starting state of the automaton is the leftmost state and the final state is circled. The automaton in (31) will accept a string consisting only of 1s; a single 0 sends the automaton to a non-accepting state from which it cannot escape. By our conventions in (30) this corresponds to a model where all elements of P are in $P \cap Q$. That is $P \subseteq Q$ or every$(P, Q) = 1$. In brief, the automaton in (31) will accept a string just in case it witnesses a model where every(P, Q) is true.

(32) *some:*



Similarly, consider the automaton in (33):

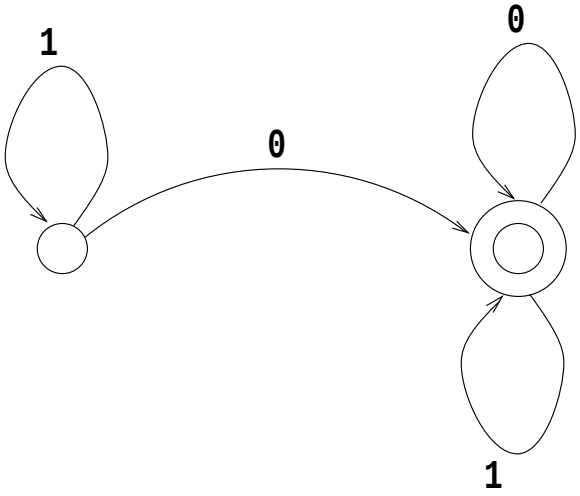
(33) *no:*



Notice that the automaton in (33) is like the automaton in (32) except that the accepting state and the non-accepting states have been interchanged. Thus, the automaton in (33) will accept a string consisting only of 0s; if it encounters a 1 in a string, it will move to a non-accepting state from which it cannot escape. Since 0 corresponds to an element in $P - Q$, the strings accepted by the above automaton witness models where no P is Q.

Finally, the following automaton simulates "not all P are Q", the complement of the regular set given in (31):

(34) *not all:*



The automaton above will accept any string that contains at least one 0. Notice, again, that the automaton in (34) is like the one given in (31) except that the accepting and non-accepting states have been interchanged.

The automata in (31), (32), (33) and (34) complement the famous Aristotelian square of opposition, which was long taken as the basis for philosophical logic. Notice that we can simulate negation in the following way:

(35) Given an automaton, Q_D, which simulates a first-order quantifier $\text{Quan}_D(P,Q)$, we can construct an automaton, Q'_D, which simulates its negation, $\text{Quan}'_D(P,Q)$, by mapping every accepting state in Q_D, to a non-accepting state in Q'_D and mapping every non-accepting state in Q_D, to an accepting state in Q'_D.

By (35), the automaton which simulates $\text{Quan}'_D(P,Q)$ will halt on an input string just in case the automaton which simulates $\text{Quan}_D(P,Q)$ does not halt and vice verse.

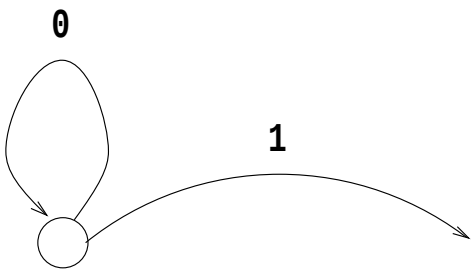
Consider the sentence in (36):

(36) Some but not all somnambulists are philatelists.

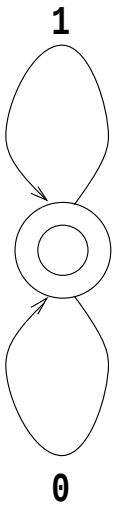
The sentence in (36) involves the conjunction of two first-order quantifiers. We can simulate conjunction of first-order quantifiers Quan_1 and Quan_2 by the following trick:

(37) Given that a finite state automaton Q_1 simulates $\text{Quan}_1(P,Q)$ and a finite state automaton Q_2 simulates $\text{Quan}_2(P,Q)$, we can simulate the conjunction of the two quantifiers, $\text{Quan}_1 \wedge \text{Quan}_2(P,Q)$, by running Q_1 and Q_2 in parallel on the witness string, ω. If both Q_1 and Q_2 accept ω, then $\text{Quan}_1 \wedge \text{Quan}_2(P,Q)$ is true in the model witnessed by ω.

Consider a quantifier like "at least n" where n is an integer. This quantifier can be modeled by copying the following fragment n times:

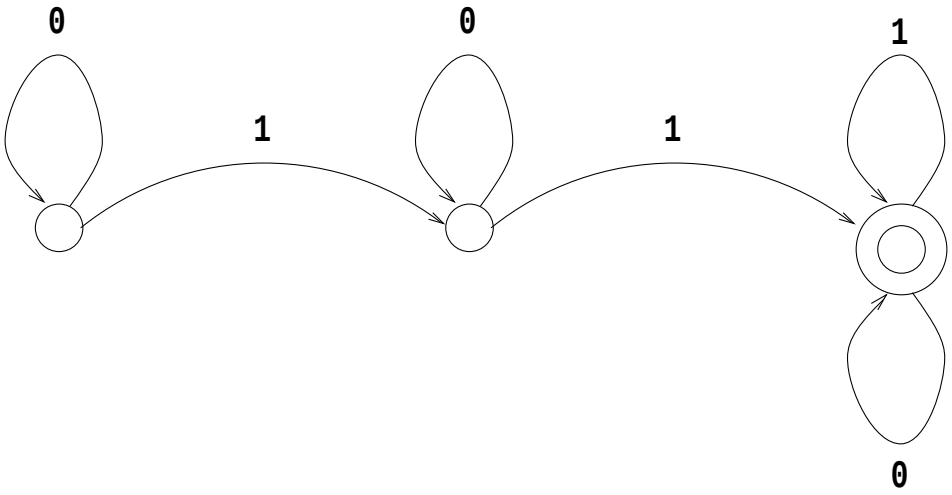


and then attaching a final state with transitions for '0' and '1' leading back into it:



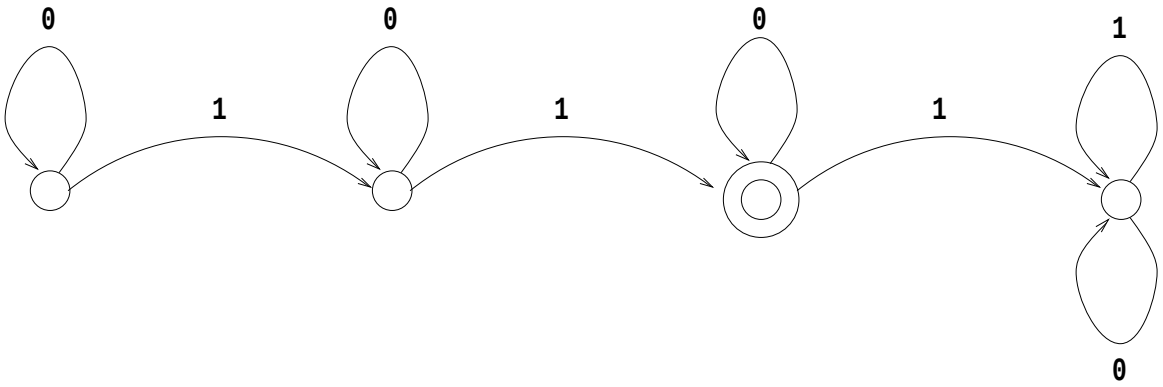
Thus, *at least two* can be mapped to the automaton shown in (38):

(38) *at least two:*



An obvious manipulation yields an automaton that will accept *exactly n*; for example, *exactly two* can be simulated by the following automaton:

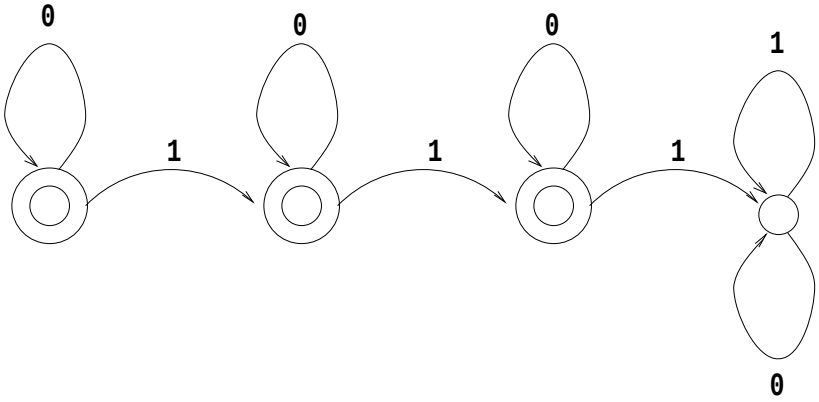
(39) *exactly two*



The automaton in (39) is like the one in (38) except that there is a transition labeled "1" leading from the accepting state to a non-accepting state. The language recognized by (39) consists of strings containing exactly two 1's. If we follow standard conventions in taking *the* to be interpreted as *the one*, we have an account for the definite article in terms of semantic automata.

We can easily construct an automaton for *at most two* from the automaton in (39) by making the first two states accepting states:

(40) *at most two:*



Next, consider a quantifier like the one in (41):

(41) *At most five or exactly ten* somnambulists smoke.

The complex determiner in (41) involves the disjunction of two first-order quantifiers. We can simulate such a quantifier by consider the union of two regular sets; in particular, we can use the following trick:

(42) Given that a finite state automaton Q_1 simulates $\text{Quan}_1(P, Q)$ and a finite state automaton Q_2 simulates $\text{Quan}_2(P, Q)$, we can simulate the conjunction of the two quantifiers, $\text{Quan}_1 \vee \text{Quan}_2(P, Q)$, by running Q_1 and Q_2 in parallel on

the witness string, ω. If either Q_1 or Q_2 accepts ω (or both), then $\text{Quan}_1 \vee \text{Quan}_2(P, Q)$ is true in the model witnessed by ω.

Combining the tricks in (35), (37) and (42), we see that we can simulate a large set of first-order quantifiers since both sets are closed under complementation, intersection and union. This is unsurprising given:

(43) a. **THEOREM.** The class of regular sets is closed under complementation. That is, if L is a regular set and $L \subseteq \Sigma*$ then $\Sigma * - L$ is a regular set.
b. **THEOREM.** The regular sets are closed under intersection.
c. **THEOREM.** The regular sets are closed under uncion, concatenation and Kleene star.

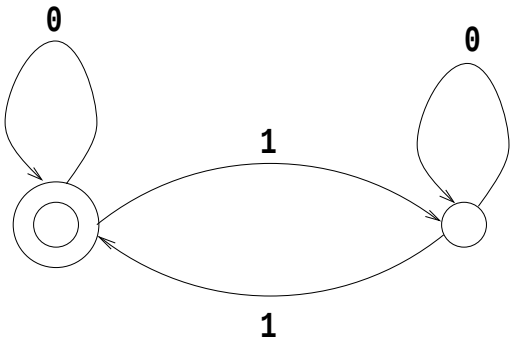
The proofs of the theorems in (43) can be found in Hopcroft & Ullman (1979). The closure properties in (43) mean that a finite set of semantic automata could be used to generate the first-order quantifiers in natural language. On analogy with the technique used in Keenan & Stavi (1986), we could define a set of basic automata, show that it is closed under boolean operations and argue that some boolean combination of the basic automata can be used to simulate any arbitrarily selected first-order quantifier. Consider in this light the following theorem from van Benthem (1986):

(44) **THEOREM.** The first-order definable quantifiers are precisely those that can be accepted by permutation-invariant acyclic finite state automata.

The term *acyclic* in (44) means that no state in the finite state automaton can be connected to an earlier state; once a state has been exited, there is no way of revisiting it, although a state may be connected to itself. *Permutation-invariance* means that the automaton is insensitive to the order of "0"s and "1"s in the string; it will accept any order so long as the quantity of "0"s and "1"s is correct.

We can extend the coverage a bit by allowing cyclic finite state automata into the set. The following is an automaton that will simulate *an even number of*:

(45) *an even number of:*



The following theorem, again from van Benthem (1986), shows how this extends our coverage:

(46) **THEOREM.** The permutation-closed languages or quantifiers recognized by finite two-state automata are:

> *all, some, no, not all, an even number of, an odd number of, all but an even number of, all but an odd number of.*

Finally, we should note that we have not accounted for all the natural language quantifiers in the above discussion:

(47) a. *Dr. Caligari's* somnambulist stalked the city.
b. *Most* somnambulists sleep deeply.
c. *At least half of the* somnambulists failed to vote.
d. *More* somnambulists *than* insomniacs abhor ludism.
e. Every integer is either even or odd.

We have said nothing about non-logical quantifiers like the possessive in (47)a; notice that possessives are not permutation invariant (Keenan & Stavi, 1986) and so deserve special treatment. Nor can we simulate *most* as in (47)b or proportional determiners as in (47)c using finite state automata. These quantifiers are not first-order since they are not compact (for example, see Landman, 1991, for a proof that *most* is not first-order); these quantifiers can, however, be simulated using push-down automata, the stack being necessary to keep track of proportions. Since the push-down automata have rather different properties with respect to the learning algorithms discussed in the next section, we will put them aside and leave them as a topic for future research. The complex determiner in (47)d is not first-order and has a rather different structure from the determiners discussed above. These latter determiners take two one-place predicates to a truth value while the determiner in (47)d takes three one-place predicates to a truth value. Again, we will not treat these determiners in the present work.

Finally, the example in (47)e involves a first order determiner that cannot be simulated by a finite state automaton for the simple reason that the automaton will never halt. By definition, a finite state automaton accepts a string just in case it halts in a final state. The proper analysis of (47)e would require some special analysis which we will not undertake here. Examples like (47)e do seem rather remote from the primary linguistic data, so that excluding such cases does not seem particularly troublesome.

4 Learning Algorithms

We have seen, so far, that first order quantification can be simulated using very simple machines, finite state automata. In particular, a first order generalized quantifier, $\mathrm{Quan}(P, Q)$, can be treated as a regular language, L. As such, there exists a finite state automaton, call it M_{Quan}, which accepts L. Furthermore, the class of finite state au-

tomata which simulate first-order quantification in natural language, call it $\mathcal{M}_{\mathrm{Det}}$, has a very simple structure:

(48) a. The class of quantifiers which can be simultated by permutation invariant finite state automata: for example, *some*, *every*, *no*, *an even number of* and so forth.
b. The class of numeric quantifiers, like *exactly* n, *at least* n, *at most* n and so forth, which can be recognized by acyclic finite state automata having either $n+1$ or $n+2$ states.

Indeed, following the strategy developed by Keenan & Stavi (1986) would could define a basic class of automata, $\mathcal{M}_{\mathrm{basic}}$, and simulate any first-order quantifier by boolean combinations of automata in $\mathcal{M}_{\mathrm{basic}}$

In this section, we will turn to a general discussion of the learnability properties of $\mathcal{M}_{\mathrm{Det}}$. As we shall see, a powerful algorithm, due to Angluin (1987), exists which will learn the set of regular languages. Since the regular languages are a superset of the set of languages accepted by the automata in $\mathcal{M}_{\mathrm{Det}}$, we will have established the learnability of the latter set; in doing so, however, we must convince ourselves that the environment for learning presupposed by Angluins algorithm is a plausible one. We will turn to this task in section 4.1. We will turn to the algorithm itself in section 4.2. Angluin's algorithm itself is more powerful than we require to learn $\mathcal{M}_{\mathrm{Det}}$; we will discuss possible simplifications in section 5

4.1 The Environment for Learning

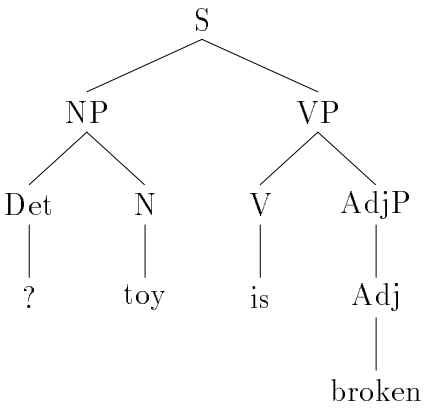
We turn now to a characterization of the environment in which learning must take place. In order for semantic learning to take place, it is insufficient to present the learner with a sentence drawn from the target language. If a learner were presented with a sentence and nothing else, there would be no obvious means of associating sentences with extra-linguistic objects, namely meanings. We assume that the learner must be presented with a sentence paired with some extra-linguistic object which will provide a set in which the subparts of the sentence can take their denotations. Normally, one assumes that the objects associated with sentences be sets of things drawn from the external world, but this need not be so. The object may be syntactic in nature; for example, it might well be a mental representation which, in turn, has semantic content.

Let us return to one of the notions that has motivated much of the thinking behind work on word learning. Recall that associationist theories required that the learner associate phonetic sequences with perceptual regularities. While this theory has an ample number of shortcomings, it has the virtue of tying language to the external world, to the degree that perceptual regularities are caused by events external to the learner. Thus, while the theory is flawed, we should at least preserve some of its elements. I will suppose, in particular, that the learner has a *sensorium*, where I take the sensorium to be an abstract, centralized level of representation of sensory input. I will assume, for example,

that objects in the visual field have been individuated at this level, and so forth. Thus, the sensorium is somewhat removed from raw sensory input.

During learning, the learner is presented with a pair consisting of a sentence and the contents of its sensorium. Notice that the meaning of the sentence is not provided to the learner. At best, if the utterance is bound to the immediate environment, subparts of the utterance will take elements of the sensorium as their denotation. There is nothing that guarantees that this is so. For example, the adult might utter "I dreamt I was a butterfly dreaming I was a man," a sentence which is obviously not tied to the external world in any obvious way. We will require, however, that adults do speak of things in the immediate environment with sufficient regularity to make learning possible. Assuming that this is so, then when the learner is presented with a sentence/sensorium pair, the learner will first attempt to parse the sentence. If the parse fails, we will assume that the learner has failed to categorize or individuate all the words properly and semantic learning is abandoned for the moment. Suppose, however, that the learner succeeds in parsing the sentence and identifies that one of the words is a determiner. Recall that this occurs in the following syntactic environment, for example:

(49)



For example, the learner might be presented with *every toy is broken*. This structure is sufficient to class *every* as a determiner.

Once the learner identifies an element that must be a determiner, it must attempt construct a string in the language $\mathrm{Det}(P, Q)$. To do so, it surveys its sensorium to determine whether anything in the sensorium satisfies the noun (P). In our example, it would test the sensorium for representations of TOYS. If it finds elements of P in the sensorium, it then scans the sensorium for elements in $P \cap Q$ and $P - Q$; intuitively, it tests the environment to see which elements of P satisfy the predicate Q and which elements do not. In our case, the learner tests to see which TOYS are BROKEN, if any. Having done so, the learner then constructs a string of 0s and 1s, which is a string in the language denoted by $\mathrm{Det}(P, Q)$. Finally, if it fails to identify any elements of P in the sensorium, it abandons the sentence as uninterpretable.

Suppose, on the other hand, that the adult utterance is of the form *det P is not Q*. The learner then surveys its sensorium to determine whether anything in the sensorium

satisfies the noun (P). If so, it scans the sensorium for elements in $P \cap Q$ and $P - Q$, as above, and having done so, it constructs a string of 0s and 1s, which is a string that is *not* in the language denoted by $\mathrm{Det}(P, Q)$. Notice that negation gives the learner information about a string that is contained in $\mathrm{Det}'(P, Q)$, the complement of $\mathrm{Det}(P, Q)$. Once again, if it fails to identify any elements of P in the sensorium, it abandons the sentence as uninterpretable.

The fact that the learner has information about the complement of $\mathrm{Det}(P, Q)$ is crucial since it supports the hypothesis that, as far as semantic learning goes, the learner has access to a *minimally adequate teacher*. Strictly speaking, a minimally adequate teacher answers two types of queries on the part of the learner. The first type is a *membership query* where the teacher responds *yes* or *no* depending on whether a particular string is a member of the target set or not. Intuitively, negation provides a membership query for the language $\mathrm{Det}(P, Q)$ allowing the learner to get access to information about strings that are not in the language denoted by the determiner. A second type of query consists of a *conjecture*, consisting of a description of a regular set. The answer is *yes* if the description is equal to the target set and a string, called a *counterexample*, drawn from difference of the target set and the description set otherwise.

It is not obvious that there is a real world correlate to a conjecture. Some further evidence comes from Brown & Hanlon (1970), a study of the role of the adult input to children, and Brown (1973). They report that adults tend to pay little attention to syntactic ill-formedness of children's utterances; they react instead to the truth value of the proposition which they suppose that the child intended to assert. This is somewhat like a conjecture to the degree that the learner produces a string contained in it hypothesis set and can then monitor parental assent to its utterances. Notice, however, that mere assent or dissent is just another form of a membership query. If the adult provides a new sentence which is intended to truthfully characterize the publicly available scene, then he or she has provided the learner with a counterexample. Furthermore, the child does not produce a full-blown description of his or her hypothesis set and present it to the parents for their approval.

Summarizing, the learner has partial access to a minimally adequate teacher. This access comes in the form of negation as well as parental assent to the truth value of the proposition encoded by the learner. We might say that they learner has access to a noisy oracle. The oracle is noisy to the degree that it cannot inspect a description of the learner's hypothesis and since it does not have reliable access to the proposition that it supposes the learner was trying to encode. This implies that the learner must approximate the target set statistically. Nevertheless, we can say that the learner has access to some negative data. Thus, semantic learning is in sharp contrast to syntactic learning where the learner is not systematicly informed about ungrammaticality. The fact that the learner has access to a minimally adequate teacher, albeit a noisy one, greatly simplifies the learning task and implies that there is a sound, tractable algorithm for learning first-order quantifier denotations.

4.2 The Learning Algorithm

The learning algorithm presented here is that of Angluin (1987). The basic idea is that the learner observes strings drawn from some regular set and uses them to construct an *observation table*. The observation table, itself, is equivalent to a finite state automaton. In this subsection we will discuss how the learner constructs the observation table and how to interpret observation tables as finite state automata.

We begin with the definition of an observation table. We first require the notion of *prefix* and *prefix closed*:

(50) a. A string ω is a *prefix* of another string π if and only if there exists a string ρ such that $\pi = \omega \cdot \rho$ (π is equal to concatenating ω before ρ).
b. A set is *prefix closed* if and only if every prefix of every member of the set is also a member of the set.

Suffix and *suffix closed* are defined analogously. We will assume that, associated with each quantifier, L, the learner constructs an observation table, (S, E, T), which records information about finite strings over a set A. The observation table consists of:

1. a nonempty prefix-closed set S of strings,
2. a nonempty suffix-closed set E of strings,
3. a finite function T mapping $((S \cup S \cdot Q) \cdot E)$ to $\{0, 1\}$, where $T(x) = 1$ if and only if x is in the unknown regular set.

This table is a two dimensional array with rows labeled by elements of $(S \cup S \cdot A)$ and columns labeled by elements of E. An entry for row s and column e is the value of $T(s \cdot e)$. Initially, the table is set to the null string; that is, $S = E = \{\lambda\}$. The table is augmented by the algorithm on the basis of observations.

For each $s \in (S \cup (S \cdot A))$, we will let $row(s)$ denote the binary sequence in the row labeled by s in the observation table. The i^{th} element of the string corresponds to $T(s \cdot e)$ where e is the label of the i^{th} column. We now define the following:

(51) a. An observation table is closed if and only if for each $t \in S \cdot A$, there is some $s \in S$ such that $row(t) = row(s)$.
b. An obersvation table is consistent if and only if whenever $s_1, s_2 \in S$ such that $row(s_1) = row(s_2)$, then for all $a \in A, row(s_1 \cdot a) = row(s_2 \cdot a)$.

If (S, E, T) is closed and consistent then we can define a finite state automaton $M(S, E, T)$ which will accept the language L. Let $M(S, E, T) = (Q, \Sigma, \delta, q_0, F)$ be defined by:

1. $Q = \{\text{row}(s) : s \in S\}$
2. $q_0 = \text{row}(\lambda)$

3. $F = \{\text{row}(s) : s \in S \text{ and } T(s) = 1\}$

4. $\delta(\text{row}(s), a) = \text{row}(s \cdot a)$

Thus, we can reinterpret the observation table as a finite state automaton. Angluin, in fact, proves the following:

(52) **THEOREM.** If (S, E, T) is a closed, consistent observation table, then the acceptor $M(S, E, T)$ is consistent with the finite function T. Any other acceptor consistent with T but inequivalent to $M(S, E, T)$ must have more states.

In other words, $M(S, E, T)$, the finite state automaton built from (S, E, T) is the minimal automaton for accepting the language defined by T.

Angluin (1987) gives a procedure for constructing an observation set given access to a minimally adequate teacher, repeated below. The learning algorithm, $L*$, constructs an observation table, (S, E, T), until it is closed and consistent. When this happens, it constructs the finite state automaton, $M(S, E, T)$ and conjectures it. The teacher the replies either with *yes* or with a counterexample:

Initialize S and E to $\{\lambda\}$.
Ask membership queires for λ and each $a \in A$.
Construct the initial observation table (S, E, T).

Repeat:

While (S, E, T) is not closed or note consistent:

If (S, E, T) is not consistent,

the find s_1 and s_2 in S, $a \in A$ and $e \in E$ such that
$row(s_1) = row(s_2)$ and $T(s_1 \cdot a \cdot e) \neq T(s_2 \cdot a \cdot e)$,
add $a \cdot e$ to E,
and extend T to $(S \cup S \cdot A) \cdot E$ using membership queries.

If (S, E, T) is not closed,

then find $s_1 \in S$ and $a \in A$ such that
$row(s_1 \cdot a)$ is different from $row(s)$ for all $s \in S$,
add $s_1 \cdot a$ to S,
and extend T to $(S \cup S \cdot A) \cdot E$ using membership queries.

Once (S, E, T) is closed and consistent, let $M = M(S, E, T)$.
Make the conjecture M. If the Teacher replies with a counterexample t, then

add t and all its prefixes to S
and extend T to $(S \cup S \cdot A) \cdot E$ using membership queries.

Until the teahcers replies *yes* to the conjecture M.
Halt and output M.

Angluin (1987) gives the following result for the learning algorithm $L*$:

(53) **THEOREM**. Given any minimally adequate teacher presenting an unkown regular set U, the learner $L*$ eventually terminates and outputs an acceptor isomorphic to the minimum deterministic finite state automaton accepting U. Moreoever, if n is the number of states of the minimum deterministic finite state automaton accepting U and m is an upper bound on the length of any counterexample provided by the teacher, then the total running time of $L*$ is bounded by a polynomial in m and n.

The theorem shows that $L*$ will learn any regular set and will do so in time bounded by n, the number of states in the minimal deterministic finite state automaton accepting the set, and m, the length of the longest counterexample available to $L*$. Note, then, that the first-order quantifiers are learnable, since they are a subset of the regular sets. Indeed, abstracting away from the learner's ability to make conjectures, we would expect the first-order quantifiers to be learned quite efficiently since (1) aside from the numeric quantifiers, they are mainly two state automata and (2) the shortest counterexamples for each quantifier is proportional to the number of states in the automaton that simulates the quantifier.

5 Prospects

In this section, we turn to future work that the above approach to semantic learnability suggests. First, and most obviously, there is the problem of the role that conjectures play in $L*$. It seems obvious that real world learners do not propose the description of a regular set for their care takers' consideration. If we can eliminate conjectures from $L*$, we would also eliminate a potential source of information to the learner. We do not yet have a characterization of how this would affect the learner.

We can speculate that the use of conjectures is not crucial for learning natural language quantifiers. The procedure described above can be used for learner regular sets quite generally. The learner makes a conjecture when its observation table is closed and consistent and uses the result to either terminate or discover that its observation table is not, in fact, closed. As we have seen, NL quantifiers have a great deal of structure. Can the procedure by optimized to exploit this structure? Indeed, we might suppose that the learner has a stock of basic quantifiers supplied a priori. Learning a quantifier denotation

would be either to associate a determiner, like *some*, with a particular automaton, or to construct an appropriate automaton via a finite boolean combination of basic automata. In the former case, the learner could let all the basic automata run on the string supplied by its sensorium, eliminating those automata that do not accept the string. Membership queries (for example, negation) would expedite the process. Complex determiners, those formed by a boolean combination of basic determiners, would not be simulated by any basic automaton and, so, would eventually be associated with the null set; at this point, a special procedure for forming combinations of basic automata could be called on.

Assuming that we are able to eliminate conjectures from the learning algorithm, we would want to know the expected sample size for convergence and how long it would take to see a sufficient number of examples in a real text. Intuitively, we would expect that simple determiners like *some*, *all*, *no* and *the* to be learned first, since they would correspond to the simple two-state automata. Numeric quantifiers, like *at least n* would take longer and would depend on the ability to construct $n + 1$ state automata. We would want to confirm this result and investigate how these complexity results square with developmental evidence.

Finally, what about higher order quantifiers, like *most* and *50%*? These can be simulated with push-down automata, but Angluin's method must be extended for these cases. Angluin (1987) discusses extending $L*$ to context-free grammars and some work has been done by Sakakibara (1990) who uses unlabeled structural descriptions as input to the learner. The model described above uses simple transduction to reduce the semantic problem (quantifier denotations) to a syntactic problem (learning regular sets). We do not yet have a model which would transduce the structures that correlate with higher order quantification to the sort of unlabeled constituent structures that Sakakibara's algorithm uses. It may be that there is sufficient structure in higher order quantifiers in natural language that we could use some other method to construct the appropriate push-down automata. We will leave this as a problem for future research.

References

1. Angluin, D. (1987). "Learning regular sets from queries and counterexamples". *Information and Computation*, 75:87-106.

2. Barwise, J. & R. Cooper (1981). "Generalized quantifiers and natural language". *Linguistics and Philosophy*, 4:159-219.

3. van Benthem, J. (1986). *Essays in Logical Semantics*. D. Reidel Publishing Co., Dordrecht.

4. Brown, R. (1973). *A First Language: The Early Stages*. Harvard University Press, Cambridge, MA.

5. Brown, R. & C. Hanlon (1970). "Derivational complexity and order of acquisition in child speech" in J. R. Hayes (ed) *Cognition and the Development of Language*. Wiley, New York. pp. 155-207.

6. Carroll, L. (1983). *Through the Looking-Glass and What Alice Found There*. University of California Press, Berkeley.

7. Chomsky, N. (1986). *Knowledge of Language: Its Nature, Origin, and Use*. Praeger, New York.

8. Geach, P. T. (1962). *Reference and Generality: An Examination of Some Medieval and Modern Theories*. Cornell University Press, Ithaca, NY.

9. Gleitman, L. (1994). "The structural sources of verb meanings", in P. Bloom (ed). *Language Acquisition: Core Readings*. The MIT Press, Cambridge, MA. pp. 174-221.

10. Goodman, N. (1979). *Fact, Fiction, and Forecast*. Harvard University Press, Cambridge, MA.

11. Hopcroft, J. & J. Ullman (1979). *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley Publishing Co., Reading, MA.

12. Keenan, E. & L. Faltz (1985). *Boolean Semantics for Natural Language*. D. Reidel Publishing Co., Dordrecht.

13. Keenan, E. & J. Stavi (1986). "A Semantic characterization of natural language determiners". *Linguistics and Philosophy*, 9:253-326.

14. Landman, F. (1991). *Structures for Semantics*. D. Reidel Publishing Co., Dordrecht.

15. Locke, J. (1690). *An Essay Concerning Human Understanding*. reprinted 1964, Meridian Books, Cleveland.

16. Loux, M. (ed) (1974). *Okham's Theory of Terms: Part 1 of the Summa Logicae*. University of Notre Dame Press, Notre Dame.

17. May, R. (1985). *Logical Form: Its Structure and Derivation*. The MIT Press, Cambridge, MA.

18. Quine, W. V. (1973). *The Roots of Reference*. Open Court, LaSalle, Ill.

19. Sakakibara, Y. (1990). "Learning context-free grammars from structural data in polynomial time". *Theoretical Computer Science*, 76:223-242.