The Problem of Learning the Semantics of Quantifiers

Nina Gierasimczuk

October 13, 2006

Abstract

This paper is concerned with a possible mechanism for learning the meanings of quantifiers in natural language. The meaning of a natural language construction is identified with a procedure for recognizing its extension. Therefore, acquisition of natural language quantifiers is supposed to consist in collecting procedures for computing their denotations. A method for encoding classes of finite models corresponding to given quantifiers is shown. The class of finite models is represented by appropriate languages. Some facts describing dependencies between classes of quantifiers and classes of devices are presented. In the second part of the paper examples of syntax-learning models are shown. According to these models new results in quantifier learning are presented. Finally, the question of the adequacy of syntax-learning tools for describing the process of semantic learning is stated.

1 Introduction

According to an old philosophical idea, the meaning of a natural language construction can be identified with a representation of its denotation [Frege 1892]. This thought has been developed in the direction of identifying the meaning of an expression with a procedure for finding its extension [Tichy 1969, Moschovakis 1990, van Lambalgen, Hamm 2004, Szymanik 2004]. In the case of words: the meaning of "Poland" is the procedure of checking if the object in question satisfies conditions of being Poland. In the case of sentences: the meaning is a procedure for finding a sentence's logical value. The meaning of "Alice has a cat." is a procedure for checking if Alicia really has a cat. Therefore, we can say that someone understands a sentence (knows its meaning), if he knows a procedure for checking whether it is true or not.

In this paper we assume that for modelling ordinary linguistic behaviour finite models are sufficient. We state this assumption for both theoretical and practical reasons. First of all we claim that most natural language sentences have natural interpretations in finite universes. The practical reason is that if we restrict ourselves to finite models, then the objects computational semantics is concerned with, namely procedures for finding denotations, become effective (algorithmizable).

2 Quantifiers

Many authors have already considered semantics of quantifiers from a computational point of view (see e.g. [van Benthem 1986, M. Mostowski 1998]); there have also been a few such attempts in linguistics (see e.g. [Suppes 1982, Cooper 1994, Bunt 2003]).

The presence and importance of quantifiers in natural language and consequently in linguistic research is beyond discussion. We use quantifiers very often in various contexts: "all", "some", "every other", "half of"... examples can be multiplied. Computational semantics gives us an idea of the meaning of quantifier constructions. We know the meaning of the sentence:

1. Every other European is depressed

if we know how to check its logical value. The first solution for this sentence would be: we just count to two on our Europeans. One – normal, two – depressed, one – normal, two – depressed.... If this procedure is satisfied on the whole set of Europeans, then we can say that our sentence is true. If we made a wrong prediction somewhere and some "one" appeared to be depressed or some "two" was normal, we can conclude that the sentence is false in our universe of Europeans. Of course the meaning of "every other" explained above is not very obvious and common. In most usages of this quantifier we would say that "every other" is a more pictorial version of "exactly half" and means the same thing. This is the case especially when we have no natural ordering on our universe. Therefore let us now consider the second meaning of "every other". In order to check whether every other European is depressed, we can execute one of following procedures:

1. Count Europeans; count depressed Europeans; if *Number of Europeans* $=$ $2 \times$ *Number of depressed Europeans*, then the sentence is true, otherwise it is false.

2. Make Europeans stand in pairs: every normal European with a depressed European; if our pair-ordering does not leave any European alone, then the sentence is true, otherwise it is false.

Our example shows not only the trivial fact that some words or phrases are ambiguous, but also that we can use many non-equivalent procedures to operate "inside" one established meaning. Here arises the problem of identifying algorithms, which partially justifies our failures in explaining the phenomenon of synonymy (see [Moschovakis 2001, Szymanik 2004]).

In order to use these ideas and to apply them to finite models, we should think about quantifiers as classes of finite models satisfying some special conditions.[1] We make here the not very controversial restriction to monadic quantifiers. This subclass we consider sufficient for linguistic considerations. Let us define quantifier as follows:

Definition 1 *Let K be a class, closed under isomorphism, of finite models of the form $(U, R_1, \ldots, R_n)$, where $U \neq \emptyset$ and $R_i \subseteq U$, for $i = 1, \ldots, n$. K is an interpretation of monadic quantifier Q_K. For every model M and valuation $\bar{a}$ on M:*

$$M \models Q_K x(\varphi_1(x), \ldots, \varphi_n(x))[\bar{a}] \iff (|M|, \varphi_1^{M,x,\bar{a}}, \ldots, \varphi_n^{M,x,\bar{a}}) \in K,$$

where $|M|$ is universe of model M, and $\varphi^{M,x,\bar{a}}$ is a set indicated by φ in M with respect to variable x by the valuation $\bar{a}$. Quantifier Q_K of type

$$\underbrace{(1, 1, \ldots, 1)}_{n}$$

binds one first-order variable in n formulae.

Let us give an example of a quantifier defined in the way described above:

Existential quantifier ($\exists$) For all M:

$$M \models \exists x \varphi(x)[\bar{a}] \iff \mathrm{card}(\varphi^{M,x,\bar{a}}) \geq 1 \iff (|M|, \varphi^{M,x,\bar{a}}) \in K_E$$

The class of models K_E which determines the interpretation of the existential quantifier we define as follows:

$$K_E = \{(|M|, R) : R \subseteq |M| \wedge R \neq \emptyset\}$$

We have identified quantifiers with classes of appropriate finite models. This step allows us to encode models as words with certain features. Classes of models will be encoded as sets of words (languages). This encoding can be done by means of the concept of constituents (see [M. Mostowski 1998]).

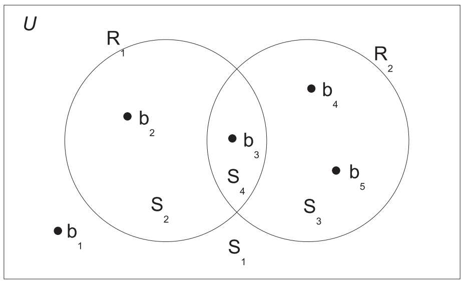
Definition 2 *The class K_Q of finite models of the form $(M, R_1, \ldots, R_n)$ can be represented by the set of nonempty words L_Q over the alphabet $A = \{a_1, \ldots, a_{2^n}\}$ such that: $\alpha \in L_Q$ if and only if there is $(U, R_1, \ldots, R_n) \in K_Q$ and a linear ordering $U = \{b_1, \ldots, b_k\}$ such that $\mathrm{lh}(\alpha) = k$ and the i-th character of α is a_j exactly when $b_i \in S_1 \cap \cdots \cap S_n$, where:*

$$S_l = \begin{cases} R_l & \text{if the integer part of } \frac{j}{2^l} \text{ is odd,} \\ U - R_l & \text{otherwise.} \end{cases}$$

Such defined intersections $S_1 \cap \cdots \cap S_n$ are called constituents of the proper model. Characters $a_1, \ldots, a_{2^n}$ are names for these constituents. Our definition says that the i-th character of α is a_j exactly when element b_i belongs to the j-th

[1] For a detailed review of results on monadic quantifiers in computational semantics, see [M. Mostowski 1998].

constituent. In other words, α uniquely encodes the model by giving information on the constituent to which every element belongs. We illustrate the idea for $n = 2$. We consider $M = (U, R_1, R_2)$, where $U = \{b_1, b_2, b_3, b_4, b_5\}$. Our model is represented by the word $\alpha_M = a_1 a_2 a_4 a_3 a_3$ over the alphabet $A = \{a_1, a_2, a_3, a_4\}$ which says that element $b_1 \in S_1 = U - (R_1 \cup R_2)$, $b_2 \in S_2 = R_1 - R_2$, $b_3 \in S_4 = R_1 \cap R_2$, and $b_4, b_5 \in S_3 = R_2 - R_1$.



Let us think of classes of monadic quantifiers as languages obtained by means of this encoding. We can now define what it means that some class of monadic quantifiers is recognized by some class of devices.

Definition 3 *Let $\mathcal{D}$ be a class of recognizing devices, and Ω be a class of monadic quantifiers. We say that $\mathcal{D}$* accepts Ω *if and only if for every monadic quantifier Q:*

$$(Q \in \Omega \iff \textit{there is some device } A \in \mathcal{D} \textit{ such that } A \textit{ accepts } L_Q).$$

Using this definition, we can now recall the following results describing dependences between classes of quantifiers and classes of devices:

Theorem 1 [van Benthem 1984] *Quantifier Q is first-order definable $\iff$ L_Q is accepted by some acyclic finite automaton.*

Theorem 2 [M. Mostowski 1998] *Monadic quantifier Q is definable in the divisibility logic $FO(D_\omega)$ $\iff$ L_Q is accepted by some finite automaton.*

Theorem 3 [van Benthem 1986] *Quantifier Q of type (1) is semilinear (elementary definable in the structure $(\omega, +)$) $\iff$ L_Q is accepted by a push-down automaton.*

Additionally we can state that there are many natural language quantifiers which lie outside the contextfree languages [Clark 1996].

3 Learning

3.1 Identification in the limit

After stating some assumptions and preliminary definitions in computational linguistics and quantifiers, we would like to present basic ideas of formal learning theory.[2] This part of the paper is needed to establish final results.

The *identification in the limit* model [Gold 1967] shows what the process of learning can look like in general and what results we can obtain using it. This model describes the learnability of a given class of languages. One language from the class is chosen. The learner gains some information about it. The information can be presented in several possible ways (the *data presentation method*). The learner's task is to guess the name of the language in question. Names of languages are simply grammars. The aim of the learner is to find a correct grammar for the presented sequence of linguistic data.

In each step of the procedure the learner is given a unit of data about the unknown language. Therefore, the learner always has only a finite set of information. In each step the learner chooses a name of a language.

The procedure is infinite. The language is identified in the limit if, after some time (finite but not specified in advance), the guesses remain the same and are correct. Identifiability concerns classes of languages. The whole class of languages is identifiable in the limit, if there is a guessing algorithm (learner) such that it identifies in the limit every language from this class. It is worth mentioning that the learner does not know when his guesses are correct. The learner proceeds infinitely, because it is not able to check if the next step won't force it to change its decision.

Identification in the limit depends on three factors: data presentation method, naming relation and chosen class of languages.

The learner has the information presented in one of two ways:

Definition 4 *A* ***TEXT for language*** L *is an* ω*-sequence,* I*, of words* $\alpha_1, \alpha_2, \ldots \in L$*, such that every word* $\alpha \in L$ *occurs at least once in* I*.*

Definition 5 *An* ***INFORMANT for language*** L *is an* ω*-sequence,* I*, of elements of* $(A^* \times \{0,1\})$*, such that for each* $\alpha \in A^*$*:*

$$\begin{array}{ll} (\alpha, 1) \text{ is in } I & \text{if } \alpha \in L \\ (\alpha, 0) \text{ is in } I & \text{if } \alpha \notin L. \end{array}$$

The learner can use one of two naming relations:

Definition 6 *A* ***GENERATOR for language*** L *is Turing Machine,* e*, such that:*

$$L = \{\alpha \in A^* : \{e\}\{\alpha\} \downarrow\}.$$

Definition 7 *A* ***TESTER for language*** L *is Turing Machine that computes the function* χ_L *such that for each* $\alpha \in A^*$*:*

$$\chi_L(\alpha) = \begin{cases} 0 & \text{if } \alpha \notin L \\ 1 & \text{if } \alpha \in L. \end{cases}$$

Let us now present Gold's table of results for learnability and non-learnability of languages from the Chomsky hierarchy.[3]

[2] For a detailed analysis of various learning algorithms see [Gierasimczuk 2005].

[3] For details and proofs see [Gold 1967].

Anomalous text	Recursively enumerable Recursive
Informant	Primitive recursive Context-sensitive Context-free Regular Superfinite
Text	Finite cardinality languages

Table 1: Identifiability results. (Anomalous text is primitive recursive text with a generator naming relation. A superfinite class of languages is a class containing all finite languages and at least one infinite language.)

3.2 Syntax Learning Algorithms

We give here examples of learning algorithms effective for a certain class of languages. Firstly let us describe the L^*-algorithm proposed by Dana Angluin [Angluin 1987]. In her paper she analyses the possibility of finite and effective identification of regular languages from an informant. The algorithm identifies the language by finding a deterministic finite automaton adequate for the unknown language. This procedure is controlled by the so-called *Minimally Adequate Teacher*. He answers two types of questions:

1. Membership queries: Is sequence α in the unknown language?
2. Extensional equivalence queries: Is the deterministic finite automaton currently being guessed by the L^*-algorithm extensionally equivalent[4] to the deterministic finite automaton which corresponds to the unknown language being learned? If it is not, then the teacher gives a counterexample.

The algorithm L^* is able to identify every regular language. L^* works in time polynomial in the number of states of the minimal DFA for the language being learned and in the maximum length of the counterexample given by the teacher.

The idea of learning with queries has been further explored, particularly in the direction of wider classes of languages. An example of such an attempt is the algorithm of Yasubumi Sakakibara [Sakakibara 1990]. It is a quite straightforward translation of the L^*-algorithm described in the previous paragraph for context-free grammars. The LA-algorithm learns a given context-free grammar on the basis of so-called structural data: skeletons of derivation trees of the given grammar. The following facts allow identification in the limit:

1. The set of derivation trees of given context-free grammar is regular.
2. A regular set of trees is recognized by some tree automaton.

[4] We say that two finite automata are extensionally equivalent if they accept the same set of strings.

3. The procedure of changing derivation trees into their structural descriptions preserves the regularity of the set.
4. The problem of learning a context-free grammar from structural descriptions is therefore reducible to the problem of learning a certain tree automaton.

It should be stressed that the aim of LA learning is not context-free language but some particular context-free grammar. It is known that for each context-free language there are infinitely many adequate grammars, therefore such a restriction is indispensable here.

4 Quantifier Learning

The problem of quantifier learning was raised and explored in [van Benthem 1986, Clark 1996, Florêncio 2002, Tiede 1999]. These were similar to our attempts to use syntax-learning models to describe learning of the semantic aspect of language.

Persisting in the declared paradigm of computational semantics leads us to the assumption that acquisition of natural language quantifiers consists essentially in collecting procedures for computing their denotations. Additionally assumptions about the adequacy of finite models and restriction to monadic quantifiers gives us an opportunity to analyse many natural language quantifiers from the point of view of syntax-learning models. We can encode a quantifier and check its learnability according to results known from the field of inference theory. For instance, one such known fact is:

Theorem 4 [Tiede 1999] *There are subclasses of FO quantifiers which are identifiable in the limit using text, e.g. the set of first-order left upward monotone quantifiers.*

We now present similar results which can be inferred from previous considerations and theorems:

Proposition 1 *The classes of FO, $FO(D_\omega)$ and semilinear quantifiers are not identifiable in the limit using text but are identifiable using informant.*

Proof This result follows directly form Theorems 1, 2, 3, and the fact that regular and context-free languages are not identifiable using text but are identifiable using informant. □

Proposition 2 *The monadic $FO(D_\omega)$-definable quantifiers are learnable using the L^*-algorithm.*

Proof By Theorem 2 every monadic $FO(D_\omega)$-definable quantifier can be represented by the set L_Q, which is accepted by some deterministic finite automaton. We know that deterministic finite automata are learnable by Angluin's L^*-algorithm. Therefore, the monadic $FO(D_\omega)$-definable quantifiers are learnable using the L^*-algorithm. □

Proposition 3 *Semilinear quantifiers of type* (1) *are learnable using the LA-algorithm.*

Proof From Theorem 3 we know that a quantifier of type (1) is semilinear iff it can be represented by a set L_Q which is accepted by some pushdown automaton. Pushdown automata are a class of devices equivalent to context-free grammars, which are effectively learnable using Sakakibara's LA-algorithm. Therefore, the semilinear quantifiers of type (1) are learnable using the LA-algorithm. $\square$

5 Conclusions

The approach presented in this paper can be treated as a strictly theoretical proposal. Nevertheless, let us now discuss some problems connected with modelling the natural process of semantic learning. First of all we can pose the question about the adequacy of tools of syntactic learning theory for describing the process of semantic learning. Let us present some intuitions about the construction of semantic competence.

	ability to check the logical value input: M and φ; $M \models \varphi$?
Semantic competence	ability to recognize inferential relations input: $\bar{\varphi}, \bar{\psi}$; e.g. $\bar{\varphi} \vdash \bar{\psi}$?
	ability to generate adequate descriptions input: M; find $\bar{\varphi}$ s.t. $M \models \bar{\varphi}$

The learnability models presented so far do not distinguish between the ability to check the logical value of a sentence and the ability to describe a given situation. There is of course a mutual translation between automata (testing devices) and grammars (generating devices). But are we allowed to treat these abilities as equivalent? Is it the case that if we can recognize the logical value of some sentence φ in a given model M, then we can also generate the sentence φ as the description of M? Some research concerning the relation between comprehension and production (equivalents of testing and generating) has already been done. It shows that the respective acquisitions of testing and generating competence are not parallel. First we can understand semantic constructions and only then are we able to use them in descriptions (see e.g. [Bates et al. 1995, Benedict 1979, Clark 1993, Fraser et al. 1979, Goldin-Meadow et al. 1976, Layton, Stick 1979]). Generating is more complicated than testing and the assumption of mutual reducibility of these two competences seems unrealistic.

To state another problem with our approach we should focus on the distinction between referential and inferential meaning.

The referential meaning of a sentence φ is given by determining a method of establishing the truth-value of φ in all possible situations. This kind of meaning is what we mainly refer to in this paper.

However, having a sentence φ we can establish its truth-value by means of inferences (recognized by our logical competence) between φ and other sentences. For example, knowing that a sentence ψ is true and $\psi \Rightarrow \varphi$ we know that φ is true; knowing that φ is false and $\psi \Rightarrow \varphi$ we know that ψ is false. In this way we determine the inferential meaning of φ.

Semantic learning models which are based on the syntax-learning approach seem to have no application in the case of learning inferential meaning. The nontrivial enterprise would be to describe possible learning mechanisms responsible for the acquisition of various semantic devices. Therefore we conclude that the learnability model presented so far is not compatible with the proposed description of semantic competence. If one wants to propose a psychologically and linguistically plausible model of semantic learning, one must fight the aforementioned subtle difficulties.

References

[Angluin 1987] D. ANGLUIN *Learning Regular Sets from Queries and Counterexamples*, **Information and Computation** 75(1987), pp. 87 – 106.

[Bates et al. 1995] E. BATES, P.S. DALE and D. THAL *Individual Differences and their Implications*, in P. Fletcher and B. MacWhinney (eds.) **The Handbook of Child Language**, Oxford 1995.

[Benedict 1979] H. BENEDICT *Early Lexical Development: Comprehension and Production*, **Journal of Child Language** 6(1979), pp. 183 – 200.

[van Benthem 1984] J. VAN BENTHEM *Semantic Automata*, **Raport of the Center for the Study of Language and Information**, Stanford 1984.

[van Benthem 1986] J. VAN BENTHEM **Essays in Logical Semantics**, Reidel Publishing Company, Amsterdam 1986.

[Bunt 2003] H. BUNT *Underspecification in Semantic Representations: Which Technique for What Purpose?*, **Proceedings of the Fifth International Workshop on Computational Semantics** 2003, pp. 37-54.

[Clark 1993] E.V. CLARK **The Lexicon in Acquisition**, Cambridge 1993.

[Clark 1996] R. CLARK *Learning First–Order Quantifiers Denotations. An Essay in Semantic Learnability*, **IRCS Technical Report** 1996, University of Pennsylvania, pp. 19 – 96.

[Cooper 1994] R. COOPER **A Framework for Computational Semantics**, Edinburgh 1994.

[Florêncio 2002] C.C. FLORÊNCIO *Learning Generalized Quantifiers*, **Proceedings of Seventh ESSLLI Student Session** 2002.

[Fraser et al. 1979] C. FRASER, U. BELLUGI, R. BROWN *Control of Grammar in Imitation, Production, and Comprehension*, **Journal of Verbal Learning and Verbal Behaviour** 2(1979), pp. 121 – 135.

[Frege 1892] G. FREGE *Über Sinn und Bedeutung*, **Zeitschrift für Philosophie und philosophische Kritik** 100(1892), pp. 25 – 50.

[Gierasimczuk 2005] N. GIERASIMCZUK *Algorithmic Approach to the Problem of Language Learning* (in polish), MA Thesis, Warsaw University 2005, also to appear in: **Studia Semiotyczne**.

[Gold 1967] E.M GOLD *Language Identification in the Limit*, **Information and Control** 10(1967), pp. 447 – 474.

[Goldin-Meadow et al. 1976] S. GOLDIN-MEADOW, M.E.P. SELIGMAN and R. GELMAN *Language in the Two Year Old*, **Cognition** 4(1976), pp. 189 – 202.

[van Lambalgen, Hamm 2004] M. VAN LAMBALGEN, F. HAMM *Moschovakis' notion of meaning as applied to linguistics*, M. Baaz, S. Friedman, J. Krajicek (eds.), **Logic Colloquium '01** ASL Lecture Notes in Logic 2004.

[Layton, Stick 1979] T.L. LAYTON, S.L. STICK *Comprehension and Production of Comparatives and Superlatives*, **Journal of Child Language** 6(1979), pp. 511 – 527.

[Moschovakis 1990] Y. MOSCHOVAKIS *Sense and Denotation as Algorithm and Value*, in: J. Oikkonen and J. Väänänen (eds.) **Lecture Notes in Logic 2** 1994, pp. 210 – 249

[Moschovakis 2001] Y. MOSCHOVAKIS *What is an algorithm?*, **Mathematic Unlimited — 2001 and beyond**, Springer 2001, pp. 919 – 936.

[M. Mostowski 1998] M. MOSTOWSKI *Computational Semantics for Monadic Quantifiers*, **Journal of Applied Non-Classical Logics** 8(1998) no 1-2, pp. 107 – 121.

[Sakakibara 1990] Y. SAKAKIBARA *Learning context-free grammars from structural data in polynomial time*, **Theoretical Computer Science** 75(1990), pp. 223 – 242.

[Suppes 1982] P. SUPPES *Variable-Free Semantics with Remarks on Procedural Extensions*, in: H. Bunt, I. Sluis, R. Morante (eds.) **Language, Mind, and Brain** 1982, pp. 21 – 31.

[Szymanik 2004] J. SZYMANIK *Computational Semantics for Monadic Quantifiers in Natural Language* (in polish), MA Thesis, Warsaw University 2004, also to appear in **Studia Semiotyczne**.

[Tichy 1969] P. TICHY *Intension In Terms Of Turing Machines*, **Studia Logica** XXIV(1969), pp. 7 – 23.

[Tiede 1999] H.J. TIEDE *Identifiability in the Limit of Context–Free Generalized Quantifiers*, **Journal of Language and Computation** 1(1999), pp. 93–102.