GREGERS KOCH

Computational Man-Machine Interaction in Simple Natural Language

Abstract

For a wide variety of semantic theories we shall present a common method of calculating the semantic representation when starting from the input text and a grammar covering the syntactic description of the text. It appears that the so-called data-flow trees play a huge and central role in this kind of analysis and translation into a semantic representation. The method here seems particularly well fit for the analysis of natural language queries to database systems. The considerations here are rather tentative and reflect research in progress.

Introduction

This paper investigates methods and tools for developing a specific kind of model of human language learning capability, by presenting a performative simulation model (here termed a computational logico-semantic induction system [16, 18]).

The same methods and tools may be applied for the purpose of implementing a wide variety of computational systems including certain kinds of rule-based expert systems and certain kinds of modern grammars (in particular the so-called unification grammars) [17].

The advantage of logico-semantic induction is its applicability in the context of constructing natural language interfaces as well as a variety of other user-friendly types of interfaces to expert systems and other computer systems.

We are studying the problem of constructing language acquisition models from specific data. That is, we could be claimed to be modelling an extremely advanced type of information processing systems, viz. human beings in the role of acquiring language capabilities. However, we are modelling the performative aspects only. No claim whatsoever is made as to the possible descriptive power of the resulting models from a psychological point of view (so we might call it purely antropomorphic information technology).

The focus of this paper is on logico-semantic induction which is a method for the systematic pattern identification and extraction in linguistic data sequences,

Proceedings of NODALIDA 1989, pages 77-85

in particular at a semantic and a combined syntactic and logical level of interpretation. It provides a means for the automated analysis of verbal protocols, and it constitutes a method for the automated construction of a logico-semantic parser.

Logico-Semantic Induction and its automated variant Computational Logico-Semantic Induction designate a completely new method from the area of logic programming and natural language processing. In contrast to the majority of other inductive approaches the method here does not deal with induction in a space of possible assertions but instead with induction in a space of possible logico-semantic representations. Here is given a short introduction to the concepts. A more comprehensive discussion by this author may be found elsewhere.

The particular kind of inductive inference that we have in mind may be illustrated by means of a diagram. Along the first axis we shall map all the possible assertions or utterances (in some suitable encoding), and along the second axis we shall map all possible representations within the framework of a particular representational notation (and similarly in a suitable encoding). A semantic theory will then occur in the shape of a mapping from the axis of utterances into the axis of representations (as long as we presuppose unambiguity, otherwise it will be generalised to a relation).

For example, we might from the following facts

crow number 1 is black
crow number 2 is black
crow number 3 is black
etc.

make the attempt to induce the following more general assertion: [2]

all crows are black.

The type of induction advocated here is of a different kind: From the following conventions

text E1 has the logico-semantic representation F1
text E2 has the logico-semantic representation F2
text E3 has the logico-semantic representation F3
etc.

we should like to find a (possibly very limited) linguistic universe L and a (logical) program P such that for each text e in L its corresponding logico-semantic representation f is the result (output) of executing the program P with the given e as input. Here the example texts E1, E2, E3 etc. are all included in the linguistic universe L.

Computational Logico-Semantic Induction may be considered a generalisation of the old concept grammatical inference that may be characterised as a kind of computational syntactic induction [11].

The possibility of automation is discussed in considerable detail. The implementation of computational semantic induction has to do with the construction

of a kind of blackbox to accept a traditional syntactic description of a linguistic universe. Besides the blackbox must accept as input a finite set of pairs $<e_k,f_k>$ where e_k is a text from the linguistic universe, and f_k is the intended semantic representation corresponding to the input e_k. For instance, the f_k may be in the form of a logical formula or a logical code. Output from the blackbox should be a program that translates linguistic input e into logical output f where especially the input e_k gives the output f_k. Here is required a complete match with the given examples.

Some possible principles for such a blackbox are discussed. These principles are clarified by application to a few small sample texts. We conclude that this new concept of computational logico-semantic induction is extraordinarily promising.

This paper contains a brief discussion and sketches a solution. A more comprehensive discussion is in preparation [13, 14, 16].

Here we are concerned exclusively with parsing or textual analysis. Analogous considerations can be made concerning textual synthesis or generation.

This work on computational logico-semantic induction was performed under heavy influence by some of the leading approaches within logic grammars like those of A. Colmerauer [3, 4], V. Dahl [7, 8, 9], F. Pereira [23, 24, 25], P. Saint-Dizier [27, 28], and M. McCord [21, 22].

It may really be seen as an attempt to unify some rather diverging tendencies in the philosophy of language, namely Creswell's lambda-calculatoric theory [5, 6] and some montagovian ones [19, 10], and on the other hand, the first order logical theories from logic grammars [12, 16]. The contribution here seems to support any of these theories.

As an example we may investigate the following English sentence

(1) Mary believes that Peter loved a woman

Within the limits of a modestly extended first order predicate calculus we may assign to the sentence the following two interpretations or logico-semantic representations, respectively:

(2) ∃y[woman(y) & believe(pres,mary,love(past,peter,y))]

(3) believe(pres,mary,∃y[woman(y) & love(past,peter,y)])

An absolutely central problem of semantics (here called the logico-semantic problem) is to assign to each input text from the appropriate linguistic universe one or several formalized semantic representations. As formalizations we will here consider only logical formulae belonging to some particular logical calculus (like definite clauses or Horn clauses, first order predicate logic, some extended first order predicate logics, the lambda calculi, and Montague's intensional logic [19]).

The principles of implementation are quite clear and fairly well developed, as may be seen by studying the example below (another example may be found in [16]). But as far as an actual implementation is concerned, we are working on it albeit in a rather slow pace (due to lack of resources).

A Small Example

Now time is probably ripe to investigate the example mentioned above. This may be seen as a further development of the ideas discussed in [16]. If the syntactic description is the following little grammar

(4) Sent → Np Vp
Np → Det Noun | Prop
Vp → Tv Np | Vp-s that S

(in the last production rule we have used a categorial grammar notation) then we may look for a representative, also called an exhaustive text. Such an exhaustive sample text may be the following:

Mary believes that Peter loved a woman

Within the chosen semantic representational notation (a predicate calculus of arbitrary high order, PC_ω) we may prefer to use a kind of generalised quantifiers for representing some (two) possible interpretations of the sample text in the following way:

(5) a(y,woman(y),believe(pres,mary,love(past,peter,y)))

(6) believe(pres,mary,a(y,woman(y),love(past,peter,y)))

The two interpretations deviate by one having as a presupposition the existence of such a female and the other not having that presupposition. Montague grammars like PTQ would obtain the same distinction.

If we choose to consider the first formula (5) to be the intended representation, the method here will lead in a mechanical fashion to the logic program shown below (7), written in the form of a logic grammar.

The program constitutes just a syntactical description augmented with attributes or decorations as may be seen by ignoring the functional arguments (then quite simply the grammar of (4) occurs).

Let us see what happens more precisely in our method. The intended resulting formula (5) should be represented as a tree structure like that in figure 1. Then the following steps should be performed:

Step 1: Enumerate the boxes in the intended result structure. (In our example this means that the boxes will get the numbers from 1 to 7, as in figure 1).

Step 2: Construct the syntactic structure (by performing parsing or syntactic analysis).

Step 3: Create a match between the result structure and the syntactic structure. More precisely, make a connection from a numbered box in the result structure to the lexical category in the syntax structure to which the word (lexical or syncategorematic) belongs. This is an indication of the vertex in the syntax tree where that fraction of the result structure

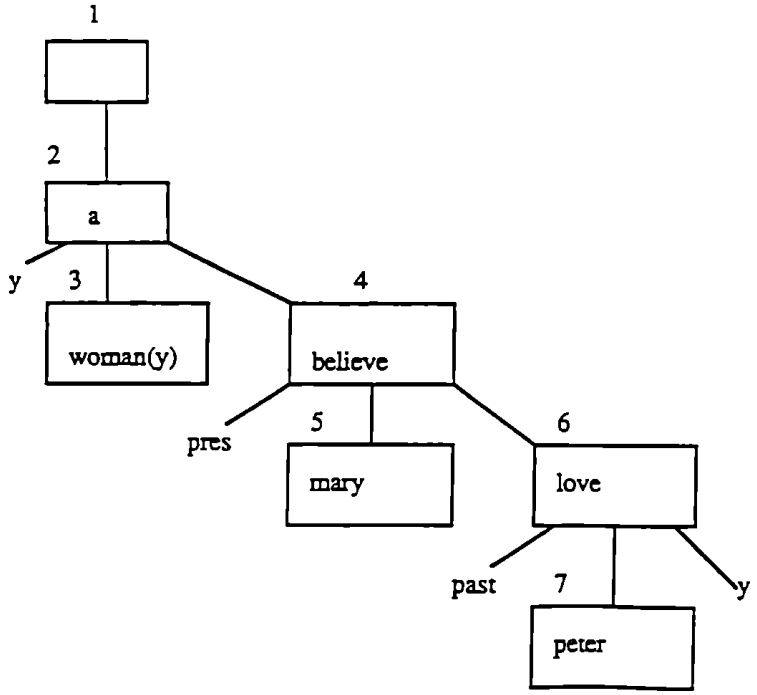

Figure 1:

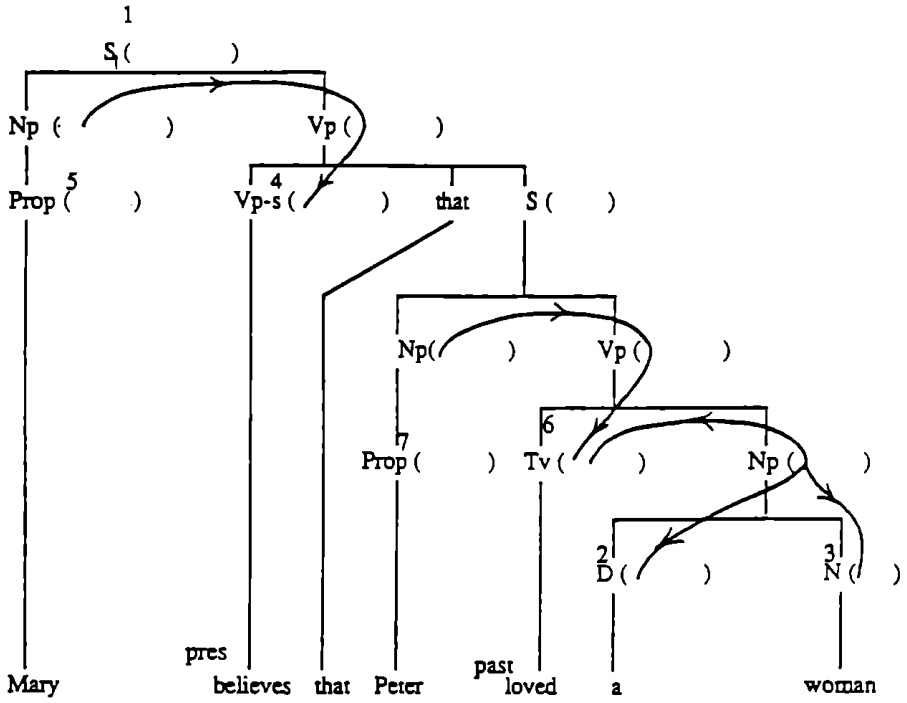

Figure 2:

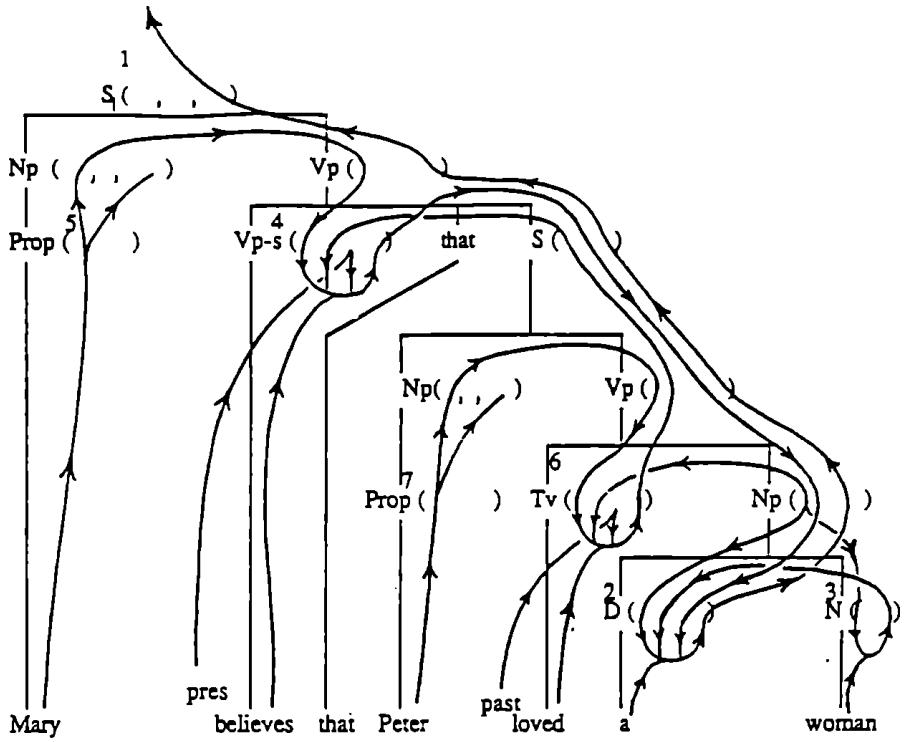

Figure 3:

having the relevant word as its top vertex, is being constructed as the result attribute of the vertex.

Step 4: Construct the flow from the so-called focus variables (form a new variable for each Np phrase, as in figure 2).

Step 5: Construct the flow in the lexical rules.

Step 6: Connect each pair of numbers corresponding to an edge in the result structure (here the tree structure should be respected, as in figure 3 where the following pairs are connected: 7–6, 5–4, 6–4, 3–2, 4–2, 2–1).

Step 7: Check the consistency concerning arity and local flow.

In our example the augmented syntactic structure will be like figure 3.

The resulting logic grammar will be the following:

(7) S(V,W,U) → Np(X,Y,Z),Vp(X,W,V,U)
Np(X,Y,Z) → Prop(X)
Np(X,Z,W) → D(X,Y,Z,W),N(X,Y)
Vp(Y,X1,Y1,V) → Vp-s(X,Y,Z,W),[that],S(W,Z,V)
Vp(Y,W,V,U) → Tv(X,Y,Z,W),Np(Z,V,U)
D(X,Y,Z,a(X,Y,Z)) → [a]
D(X,Y,Z,every(X,Y,Z)) → [every]

Concluding Remarks and Perspectives

As to which representation languages are acceptable with respect to this method, there seems to be a high degree of freedom so that we seem to be near the implementation of a general information theoretical or computer science paradigm like this:

Anyway, there exists a requirement that a kind of homomorphy property, a kind of compositionality should be available in the relationship between input and output. One or another variant of Frege's principle of compositionality should be obtained:

To the extent that our rules are of the form

$$p_0(G(y_1,...,y_n)) \rightarrow p_1(y_1),...,p_n(y_n)$$

we know about the semantic representation function Sem that

$$Sem(P_0)=G(Sem(P_1),...,Sem(P_n))$$

where $P_0=P_1 \hat{} P_2 \hat{} .. \hat{} P_n$

provided that P_k is the fragment of the input text belonging to the syntax category p_k for all $k \in \{0,1,..,n\}$.

And this property is precisely one way of expressing Fregean compositionality.

One perspective of this approach is that it allows a generalisation into what we tend to call computational logico-semantic abstraction [18]. In this context it is profitable to make use of certain results from the modern computer science disciplines of logic programming, attribute grammars, and denotational semantic theories.

Another perspective concerns automated learning. Computational logico-semantic induction has the property that the system will be able to improve its linguistic performance (i.e., handling new information of a semantic nature) by adoption from a single occurrence of a grammatical rule. That must be effective automated learning par excellence!

So, besides concluding that the method of logico-semantic induction is not only new but also promising we are able to discuss AI-problems related to inductive learning from the following perspective: inductive reasoning as a way of managing linguistic information in logical systems. Hence in this case it is not really a question of empirical information, and of course its relationship to AI is always arguable (what is the precise content of AI?), but a surprisingly high degree of automated learning is actually obtainable.

References

[1] J.W. Bresnan [ed.]. 1982. *The Mental Representation of Grammatical Relations.* MIT Press.

[2] E. Charniak & D. McDermott. 1985. *Introduction to Artificial Intelligence.* Addison-Wesley.

[3] A. Colmerauer. 1978. Metamorphosis Grammars. L. Bolc [ed.]. *Natural Language Communication with Computers.* 133–189. Lecture Notes in Computer Science No. 63.

[4] A. Colmerauer. 1982. An Interesting Subset of Natural Language. K.L. Clark & S.-Å. Tärnlund [eds.]. *Logic Programming.* Academic Press.

[5] M.J. Cresswell. 1973. *Logics and Languages.* Methuen.

[6] M.J. Cresswell. 1985. *Structured Meanings, the Semantics of Propositional Attitudes.* MIT Press.

[7] V. Dahl. 1979. *Logical Design of Deductive Natural Language Consultable Data Bases.* Proc. Fifth International Conference on Very Large Data Bases. Rio de Janeiro, Brazil.

[8] V. Dahl. 1979. *Quantification in a Three-Valued Logic for Natural Language Question-Answering Systems.* Proc. IJCAI. Tokyo, Japan.

[9] V. Dahl & M. McCord. 1983. *Treating Coordination in Logic Grammars.* Am. Journ. Comp. Ling.

[10] D.R. Dowty, R.E. Wall & S. Peters. 1981. *Introduction to Montague Semantics.* D. Reidel.

[11] J.J. Horning. 1969. *A Study of Grammatical Inference.* Technical Report No. CS 139. Computer Science Department, Stanford University.

[12] G. Koch. 1985. *Who is a Fallible Greek in Logic Grammars.* 54–77 in [15].

[13] G. Koch. 1986. *Relating Definite Clause Grammars and Montague Grammars.* Institute of Computer Science, Technical University of Denmark. 20 pages.

[14] G. Koch. 1986. *The Application of Prolog for the Translation into a Semantic Representation.* Proc. Nordic Seminar on Machine Translation. EUROTRA-DK, Copenhagen. 175–189.

[15] G. Koch [ed.]. 1985. *Fifth Generation Programming vol. 1: Logic Programming in Natural Language Analysis.* Proceedings of Workshop in Copenhagen. Dec. 1984. DIKU report 85/2.

[16] G. Koch. 1987. *Automating the Semantic Component.* Information Processing Letters 24. 299–305.

[17] G. Koch. 1987. *LFG og Prolog.* Institute for Applied and Mathematical Linguistics, Copenhagen University.

[18] G. Koch. [Forthcoming]. *A Technical Perspective on Expert Systems, Modern Grammars, Semantic Abstraction, and their Implementations.* Proc. Fifth Symposyum on Empirical Foundations of Information and Software Science, Risø, Denmark, Nov. 1987.

[19] R. Montague. 1974. *The Proper Treatment of Quantification in Ordinary English.* In [20].

[20] R. Montague. 1974. *Formal Philosophy.* Yale University Press.

[21] M. McCord. 1982. Using Slots and Modifiers in Logic Grammars for Natural Language. *Artificial Intelligence.* 18,3:327–367.

[22] M. McCord. 1987. Natural Language Processing in Prolog. A. Walker [ed.]. *Knowledge Systems and Prolog.* Addison-Wesley.

[23] F.C.N. Pereira. 1983. *Logic for Natural Language Analysis.* SRI International. Technical Note 275.

[24] F.C.N. Pereira & D.H.D. Warren. 1983. *Parsing as Deduction.* SRI Technical Report 293. Stanford Research Institute.

[25] F. Pereira & D. Warren. 1980. Definite Clause Grammars for Language Analysis — A Survey of the Formalism and a Comparison with Augmented Transition Networks. *Artificial Intelligence.* 13,3:231–278.

[26] U. Reyle & W. Frey. 1983. *A Prolog Implementation of Lexical-Functional Grammar.* Proc. IJCAI, International Joint Conference on Artificial Intelligence. 693–695.

[27] P. Saint-Dizier. 1985. *On Syntax and Semantics of Modifiers in Natural Language Sentences.* In [15].

[28] P. Saint-Dizier. 1986. An Approach to Natural Language Semantics in Logic Programming. *The Journal of Logic Programming.* 3(4):329–356.

Institute of Datalogy
University of Copenhagen
DK 2100 Ø
Copenhagen, Denmark