Toward a Method for the Automated Design of Semantic Representations

Gregers Koch
Departement of Computer Science (DIKU)
University of Copenhagen
E-post: gregers@diku.dk

1. Automated Semantic Induction

Information is a concept of crucial importance in any conceivable scientific endeavour. Also the modeling of information or information modeling is becoming more and more important, not least in information systems. Databases, knowledge bases as well as knowledge management systems are continually growing. Modeling helps to understand, explain, predict, and reason on information manipulated in the systems, and to understand the role and functions of components of the systems. Modeling can be made with many different purposes in mind and at different levels. It can be made by emphasising the users' conceptual understanding. It can be made on a domain level on which the application domain is described, on an algorithmic level, or on a representational level. Here the interest is focused on modeling of information on a representational level.

As a logic programming activity, we are studying certain computational linguistic problems, in particular concerning the automation of data flow analysis synthesizing a sort of data flow structures. This automated method may be utilized for inductive purposes (in the sense of generalization from examples). The method here is called logico-semantic induction and it constitutes an efficient kind of automated program synthesis.

More precisely we are studying data flow structures and the construction and testing of software for automatic implementation of the principles of logico-semantic induction [Koch 1988, 1991, 1993]. The software seems to apply conveniently for the automated construction of 1) natural language interfaces to

knowledge based systems, 2) simple translation between human languages like English, Danish, and Japanese, 3) compilers of traditional programming languages, and 4) translators between fragments of human languages and logical formalisms.

1.1. An Introduction to the System

When constructing natural language interfaces, one necessarily has to select a linguistic fragment or a sublanguage to be used by the user in the communication or interaction with the computational system [Abramson and Dahl 1989]. Precisely which sublanguage or fragment should be selected, is an open question. Furthermore, which internal representation should be preferred from the point of view of efficiency or practical convenience, is also an open question [Pereira 1987]. Hence it may be a good idea to construct a frame system allowing flexible experimentation with language constructs and their possible representations. What is needed, is a facility to help in the automated translation of the selected constructs into some flexible and useful representations allowing further processing (e.g. for translation into a database query language or into an implemented programming language).

Here we discuss such a flexible home-made frame system. It is functioning in an inductive manner, since the system requires the user to specify a text or query combined with the suggestion for an internal representation. In this case the system is capable of working out the details of a translator system translating texts of a syntactic form similar to the given text into the internal representation in use. In other words, this is a method for inductive and automated program synthesis, sometimes called logico-semantic induction. Such a system may also be used for obtaining completely automatic implementations of parsers implementing semantic theories like Discourse Representaion Theory and Situation Semantics.

For a beginning let us give a closer description of the mentioned method for inductive and partly automated construction of programs for logical analysis of natural language texts [Koch 1992]. In an earlier paper the same method was applied to a scientific abstract [Koch 1997], and in a recent paper the same method was applied to Discourse Representation Theory [Koch 1999].

We shall illustrate the method by dealing with an utterly simple example. Here we shall analyse a tiny little sentence of four words

Peter seeks a mermaid

This sentence can trivially be described by means of the following syntax (a simple context-free grammar):

S → NP VP.
NP → Prop | D N.
VP → IV | TV NP.

The semantics in the form of a semantic parser, that is a program translating into some semantic representation, may be given in the form of a definite clause grammar (short DCG) like this

S(Z) → NP(X,Y,Z), VP(X,Y).
NP(X,Y,Y) → Prop(X).
NP(X,Z,W) → D(X,Y,Z,W), N(X,Y).
VP(.....) → TV(.....), NP(.....). (*)

with some lexical data

D(X,Y,Z,a(X,Y,Z)) → [a].
TV(X,Y,seeks(X,Y)) → [seeks].
N(X,mermaid(X)) → [mermaid].
Prop(Peter) → [Peter].

The last line (*) of the program was missing, it still needs to be filled out.

If we write it this way

VP(X,W) → TV(X,Y,Z), NP(Y,Z,W).

we obtain the extensional reading.

On the other hand, if we fill it out this way

VP(X,Z) → TV(X,Y,Z), NP(_,_,Y).

we obtain an intensional reading, where the existence of a mermaid is not presupposed.
A central point of this analysis is the observation, that the construction of this kind of parser programs can relatively easily be done automatically by another program, sometimes called the meta parser. The central part of such a meta parser may conveniently be a kind of data flow analysis, but other possibilities are also conceivable.
As an application, let us return to and continue with a closer analysis of the extensional reading of the small example sentence. In short, by interpreting a(X,Y,Z) in different ways, we obtain a number of different semantic representations.

For example, interpreting it this way

a(X; Y; Z) = ∃X[Y &Z]

will give a predicate logic expression as the obtained semantic representation. On the other hand, interpreting it this way

a(X; Y; Z) = [[X]; Y &Z]

will lead to a representation in the style of Discourse Representation Theory [Kamp and Reyle 1993].

2. Automated Design of Semantic Representation

We shall discuss a couple of new methods for extracting the informational content of (relatively brief) texts formulated in natural language (NL). It makes sense to consider information extraction from NL texts to be essentially the same task as building simple kinds of information models when parsing the texts. Here we present two new methods that are distinguished by extreme simplicity and robustness. The simplicity makes programming of the methods feasible, and so a kind of automatic program synthesis is obtained. The robustness causes wide applicability, and so the methods have a high degree of generality. We have to distinguish between two cases (or two situations).

Case 1: Let us here assume that we are in possession of a suggestion for the form of the information models. This means, we have a formalism for mapping into. In this case we can generate the translating parsers automatically from carefully selected examples, by means of so-called logico-semantic induction, as described for instance in [Koch 1988, 1991, 1993].

Case 2: As a contrast, let us here assume that we do not know of any suggestion for the form of the information models. In this case we may follow the following recipe and hereby obtain a kind of semi-automatic design of consistent information structures.

In the simplest version of this analytic method, our text should be exposed to the following selectional or constructive steps.

Step 1: Select an example text.

Step 2: Construct a syntactic description, sufficient to describe the example text.

Step 3: Perform a complete data flow analysis on the example text to obtain an augmented syntactic tree structure (sometimes called a data flow structure).

Step 4: Extract a definite clause grammar (DCG) from the data flow structure.

Step 5: Supplement the DCG grammar obtained in step 4 with relevant lexical information.

Step 6: Perform a traditional kind of symbolic execution on the logic program obtained from step 4 and step 5.

Step 7: Extract all possible symbolic equations from the symbolic execution in step 6.

Step 8: Solve the symbolic equations to obtain a feasible solution. This solution constitutes a feasible representational structure.

Step 9: Perform logico-semantic induction, in similarity with case 1 above, to the text of step 1 combined with the solution of step 8.

Step 10: Perform a forward test run on the example from step 1, to compare the result with the solution of step 8.

Step 11: Perform a backward test run on the solution from step 8, to compare the result with the original text from step 1.

Step 12: Perform an accumulation of the acceptable fractional linguistic descriptions in the shape of definite clause grammars obtained through this process from a variety of carefully selected example texts.

In a tentative comparison with some important alternative approaches, it makes sense to distinguish between the design of representation, manual programming, and automated program synthesis.

3. Discussion

Hans Kamp and followers seem to handle the design of representation very well. I do not know if they can produce the hand-coded programs. Roger Schank and followers seem to be able to handle both the design and the hand-coded programming. But we should notice that their software solutions seem to suffer from mediocre reproducibility. As to other approaches, including that of professor Kawaguchi [Yoshihara 2000] and others, they can handle the design of representation, but they seem unable to produce the hand-coded translation software. Some other alternatives were discussed in earlier papers. None of the mentioned alternative approaches seems to be able to produce the automated program synthesis. In contrast, my project reported here seems to handle the de-

sign of representation rather well and to give partial solutions to the manual programming (steps 3-8) and the automated program synthesis (steps 9-11 and case 1). In short, the method of case 1 and in particular the method of case 2 seem to be unique compared to today's scientific literature.

4. References

Abramson, H. and V. Dahl 1989. *Logic Grammar*, Springer.

Brown, C. G. and G. Koch, eds. 1991. *Natural Language Understanding and Logic Programming*, III, Amsterdam:North-Holland.

Kamp, H. and Reyle U. 1993. *From Discourse to Logic*. Amsterdam:Kluwer.

Koch, G. 1993. Montague's PTQ as a Case of Advanced Text Comprehension, In Kangassalo, H. et al. (eds.) *Information Modelling and Knowledge Bases IV*, Amsterdam:IOS:377-387.

Koch, G. 1992. Logics and informatics in an integrated approach to natural language database interfaces. In Ohsuga, S. et al. (eds.) *Information Modelling and Knowledge Bases III*. Amsterdam:IOS:602-616.

Koch, G. 1991. Linguistic data flow structures. In Brown and Koch, op.cit.:293-308.

Koch, G. 1988. Computational logico-semantic induction. In Dahl, V. and P. Saint-Dizier (eds.). *Natural Language Understanding and Logic Programming II*. Amsterdam:North-Holland:107-134.

G. Koch, 1997. Semantic analysis of a scientific abstract using a rigoristic approach. In Kangassalo, H. et al. (eds.). *Information Modelling and Knowledge Basis VIII.* Amsterdam:IOS:361-370.

G. Koch, 1999. Discourse representation theory and induction. In Bunt, H.C. & E.G.C. Thijsse (eds.). *Proceedings of the Third International Workshop on Computational Semantics* (IWCS-3), Tilburg University:401-403.

F.C.N. Pereira and S.M. Shieber, 1987. *Prolog and Natural-Language Analysis*, CSLI, Stanford University.

S. Yoshihara, M. Wakiyama, and E. Kawaguchi 2000. An experiment on Japanese-sentence generation from SD-formed semantic data. In Kawaguchi, E., H. Kangassalo, H. Jaakkola, and I. A. Hamid (eds.) *Information Modelling and Knowledge Bases XI*, Amsterdam:IOS:205-221.