A Method for Making Computational Sense of Situation Semantics

Gregers Koch

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 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. Here are discussed two problems concerning the handling of Situation Semantics in a computational framework: 1) The problem of how to translate automatically from text to situation schemata (or similar structures), 2) The problem of how to perform automatically a reasonable interpretation of the semantics of the situation schemata. Tentative solutions to both problems are outlined here. The present solutions are heavily concerned with the application of our homemade methods for performing automated semantic induction. The proces was divided into four mappings. As our meta programs performing the automated synthesis of the translation programs rely heavily on the underlying data flow analyses, we may conclude that the method advocated here in a sense dictates the separation into the four functionalities. Many technical details have been left out in this paper, among them the treatment of grammatical tense. But with these reservations we must maintain that we have outlined a simple way of implementing parsers for Situation Semantics, parsers to translate into situation schemata and parsers for partial translation into reasonable logical interpretations.

1 AUTOMATED SEMANTIC INDUCTION

1.1 *A System Performing Semantic Induction*

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.

1.2 *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 dataflow 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

$a(X, Y, Z) = \exists X[Y \& Z]$

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].

In the remainder of the present paper, we shall elaborate on similar ideas with respect to Situation Semantics.

1.3 *Some Remarks on Situation Semantics*

In Situation Semantics, the meaning of a simple declarative sentence φ is a relation between an utterance situation u and a described situation s

$u[SIT.\varphi]s$

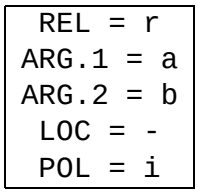
where $SIT.\varphi$ is a situation schema associated with φ. Here u must be an appropriate utterance situation with respect to the constraints induced by $SIT.\varphi$, sometimes written

$u \in MF[SIT.\varphi]$.

We shall make an attempt to spell out how a given u and a given $SIT.\varphi$ constrain or partially determine a described situation s. Here we assume the decomposition of the utterance situation u in two parts, d for discourse situation and c for the

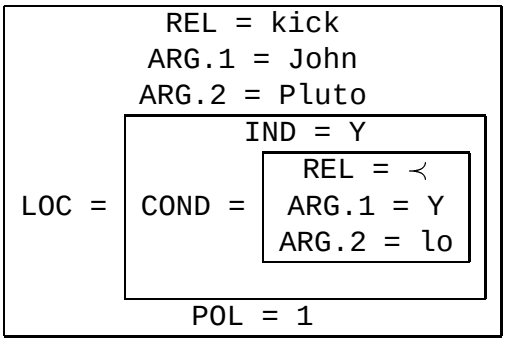
speaker's connection.

A simple sentence φ, such as "John kicked Pluto" or "Pluto was kicked by John" has a situation schema of approximately the following form



where r can be anchored to a relation, a and b to individuals, and i gives the polarity of the fact. LOC is a function which anchors the described fact relative to the discourse situation u = (d,c). We say that the partial function g anchors the location $SIT.\varphi.LOC$ in the discourse situation d,c if
g(lo) = ld
c(r),g(IND.α),ld;1
where ld is the discourse location and c(r) is the relation given by the speaker's connection c.

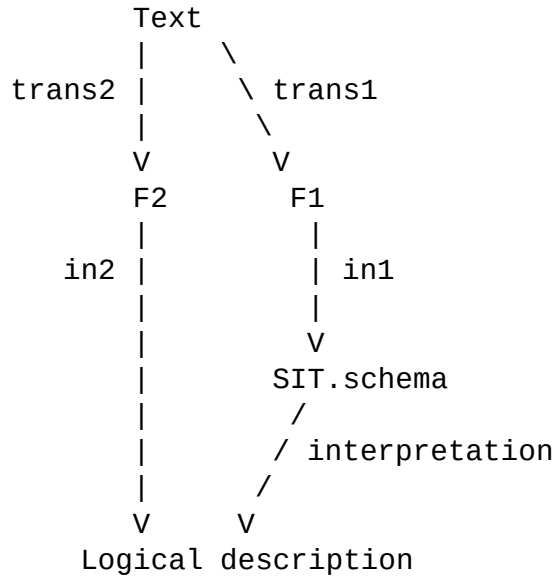
The situation schema corresponding to the sentences given here is now



Much more about Situation Semantics may be found in e.g. [Barwise and Perry 1983], [Fenstad et al. 1987], [Devlin 1991], [Loukanova 1995], [Benthem and ter Meulen 1997].

1.4 *Translation and Interpretation*

As far as we can see, the technical problems are most easily dealt with if the translation and interpretation are done in a somewhat roundabout manner:



The obvious way would be an implementation of $in1 \circ trans1$ and $interpretation$, respectively. And it is certainly possible to implement interpretation directly, only it would be slightly complicated, so instead we prefer to implement the following four functionalities: trans1, in1, trans2, and in2.

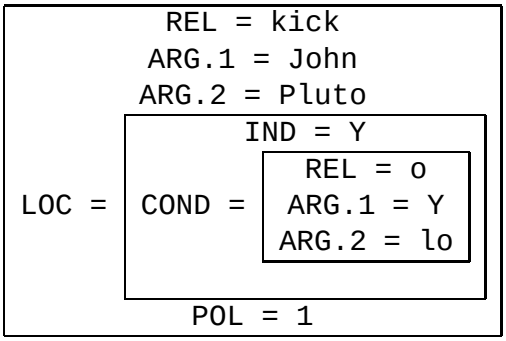
As an example, let us look at the sentence

$\varphi 3$: every boy kicks Pluto.

Here we get

$trans1(\varphi 3) = kick(every(X, boy(X)), Pluto)$

$in1(trans1(\varphi 3)) = SIT.\varphi 3 =$



$trans2(\varphi 3) = every(X, boy(X), kick(X, Pluto))$

The fourth functionality in2 is here preferred specified by means of a textual characterization of the logical relation that it designates:

$in2(trans2(\varphi 3))$ is the following logical description:

$d, c[SIT.\varphi 3]s$

iff $d, c \in MF[SIT.\varphi 3]$

and there exists anchor g on $SIT.\varphi 3.LOC$

such that in s:

if h $\supseteq$ g such that c(boy),h(X);1

then c(kick),h(X),c(Pluto);1

The interpretations may be realised by the following functionalities:

in2(W)(g) =

d,c[SIT.φ]s

iff d,c $\in MF[SIT.\varphi]$

and there exists anchor g on $SIT.\varphi.LOC$

such that in s: in2(W)(g)

in2(every(X,Y,Z))(g) = λh. if h $\supseteq$ g such that in2(Y)(h) then in2(Z)(h)

in2(a(X,Y,Z))(g) = Y(g) and Z(g)

in2(dog(X))(g) = c(dog),g(X);1

in2(kick(John,X))(g) = c(kick),c(John),g(X);1

Similarly,

in2(kick(X,Pluto))(g) = c(kick),g(X),c(Pluto);1

in2 interprets $not(W)$ by changing the polarity of the interpretation of W (e.g. from the value 1 to the value 0).

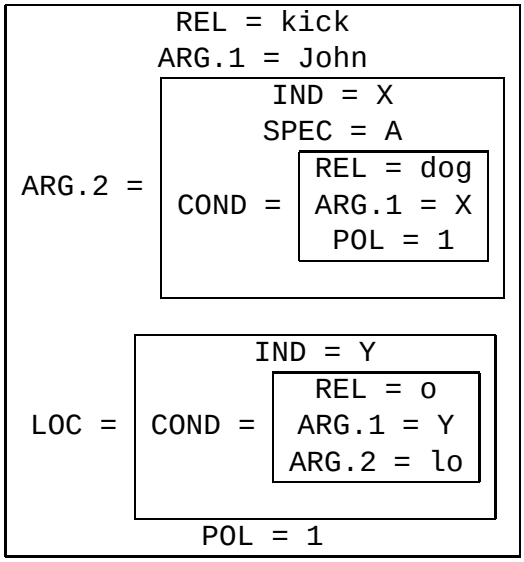
As another example,

$\varphi 4$: John kicks a dog.

In this case the four funtionalities become:

trans1($\varphi 4$) = kick(John,a(X,dog(X)))

in1(trans1($\varphi 4$)) = SIT.$\varphi 4$ =



trans2($\varphi 4$) = a(X,dog(X),kick(John,X))

in2(trans2($\varphi 4$)) is the following logical description:

d,c[SIT.$\varphi 4$]

iff d,c $\in MF[SIT.\varphi 4]$

and there exists anchor g on $SIT.\varphi 4.LOC$

such that in s:

c(dog),g(X);1

c(kick),c(John),g(X);1.

2 CONCLUSIONS

Finally a few concluding remarks:

The proces was divided into four mappings. The four mappings are easily realised by means of semantic induction, described in the previous section. The reason why we prefer to separate the two translations is this: The data flow naturally connected to the first translation (trans1) is completely different from the data flow naturally connected to the second translation (trans2). In short, the two underlying data flows diverge. As our meta programs performing the automated synthesis of the translation programs rely heavily on the underlying data flow analyses, we may conclude that the method advocated here in a sense dictates the separation into the four functionalities.

More could be said about the left side of the meaning relation to obtain a better analysis of the flow of information in a discourse. Also many technical details have been left out in this paper, among them the treatment of grammatical tense. But with these reservations we must maintain that we have outlined a simple way of implementing parsers for Situation Semantics, parsers to translate into situation schemata and parsers for partial translation into reasonable logical interpretations.

REFERENCES

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

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

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

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

M.A. Covington, *Natural Language Processing for Prolog Programmers*, (Prentice Hall, Englewood Cliffs, 1994).

P. Deransart and J. Maluszynski, *A Grammatical View of Logic Programming*, (MIT Press, 1993).

H. Kamp, A theory of truth and semantic representation, 277-322, J.A.G. Groenendijk, T.M.V. Janssen and M.B.J. Stokhof, eds., *Formal Methods in the Study of Language*, Mathematical Tract 135, Amsterdam, 1981.

G. Koch, Logics and informatics in an integrated approach to natural language

database interfaces, 602-616, S. Ohsuga et al., eds., *Information Modelling and Knowledge Bases III*, (IOS, Amsterdam, 1992).

G. Koch, Linguistic data-flow structures, 293-308, in [Brown 1991].

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

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

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

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

J. Barwise & J. Perry,*Situations and Attitudes*, MIT Press, 1983.

J.E. Fenstad et al., *Situations, Language and Logic*, D. Reidel, 1987.

K. Devlin, *Logic and Information*, Cambridge Univ. Press, 1991.

R. Loukanova, Solving natural language ambiguities in situation semantics, *Bits and Bytes*, Institute for Language and Communication, Odense University, 1996.

J. van Benthem & A. ter Meulen (eds.), *Handbook og Logic and Language*, North-Holland, 1997.

Gregers Koch is a professor of computer science and computational linguistics at the Department of Computer Science (DIKU) at Copenhagen University, Universitetsparken 1, DK 2100 Copenhagen, Denmark. He has written about 80 publications. He can be reached by email: gregers@diku.dk or Fax: (+45)35321401.