ON FORMALISMS AND ANALYSIS, GENERATION AND SYNTHESIS IN MACHINE TRANSLATION

Zaharin Yusoff

Projek Terjemahan Melalui Komputer

PPS. Matematik & Sains Komputer

Universiti Sains Malaysia

11800 Penang

Malaysia

Introduction

A formalism is a set of notation with well-defined semantics (namely for the interpretation of the symbols used and their manipulation), by means of which one formally expresses certain domain knowledge, which is to be utilised for specific purposes. In this paper, we are interested in formalisms which are being used or have applications in the domain of machine translation (MT). These can range from specialised languages for linguistic programming (SLLPs) in MT, like ROBRA in the ARIANE system and GRADE in the Mu-system, to linguistic formalisms like those of the Government and Binding theory and the Lexical Functional Grammar theory. Our interest lies mainly in their role in the domain in terms of the ease in expressing linguistic knowledge required for MT, as well as the ease of implementation in MT systems.

We begin by discussing formalisms within the general context of MT, clearly separating the role of linguistic formalisms on one end, which are more apt for expressing linguistic knowledge, and on the other, the SLLPS which are specifically designed for MT systems. We argue for another type of formalism, the general formalism, to bridge the gap between the two. Next we discuss the role of formalisms in analysis and in generation, and then more specific to MT, in synthesis. We sum up with a mention on a relevant part of our current work, the building of a compiler that generates a synthesis program in SLLP from a set of specifications written in a general formalism.

On formalisms in MT

The field of computational linguistics has seen many formalisms been introduced, studied and compared with other formalisms. Some get established and have been or are still being widely used, some get modified to suit newer needs or to be used for other purposes, while some simply die away. Those that we are interested in are formalisms which play some role in MT.

The MT literature has cited formalisms like the formalisms for the government and Binding Theory (GB) [Chomsky 81], the Lexical Functional Grammar (LFG) [Bresnan & Kaplan 82], the Generalized Phrase structure Grammar (GPSG) [Gazdar & Pullum 82] (here we refer to the formalisms provided by these linguistic theories and not the linguistic content), Context Free Grammar (CFG), Transformational Grammar (TG), Augmented Transition Networks (ATN) [Woods 70], ROBRA [Boitet 79], grade [Nagao et al. 80], metal [Slocum 84], Q-systems [Colmerauer 71], Functional Unification Grammar (FUG) [Kay 82], Static Grammar (SG) [Vauquois & Chappuy 85], String-Tree Correspondence Grammar (STCG) [Zaharin 87a], Definite Clause Grammar (DCG) [Warren & Pereira 80], Tree Adjoining Grammar (TAG) [Joshi et al. 75], etc. To put in perspective the discussions to follow, we present in Figure 1 a rather naive but adequate view of the role of certain formalisms in MT.
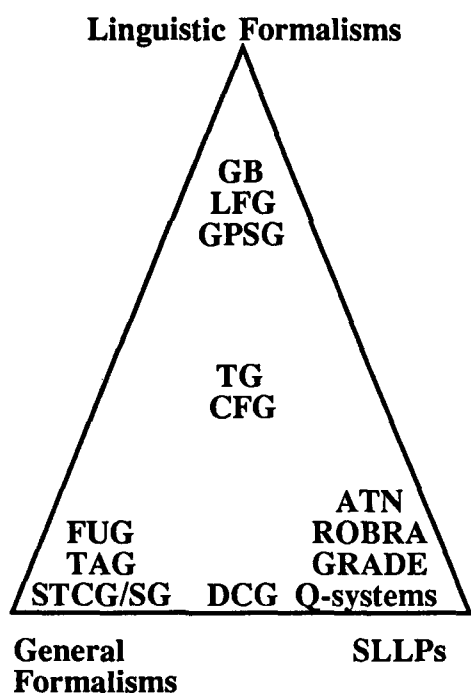
Linguistic Formalisms

```
                    GB
                    LFG
                    GPSG


                    TG
                    CFG

                                ATN
         FUG                    ROBRA
         TAG                    GRADE
        STCG/SG    DCG    Q-systems
```

General   SLLPs
Formalisms

Fig. 1 - The role of formalisms in MT.

GB, LFG and GPSG formalisms are classed as linguistic formalisms as they have been designed purely for linguistic work, clearly reflecting the hypotheses of the linguistic theories they are associated to. Although there have been 'LFG-based' and 'GPSG- inspired' MT systems, a LFG or GPSG system for MT has yet to exist. Whether or not linguistic formalisms are suitable for MT (one argues that linguistic formalisms tend to lean towards generative processes as opposed to analysis, the latter being considered very important to MT) is not a major concern to linguists. Indeed it should not be, as one tends to get the general feeling that formal linguistics and MT are separate problems, although tapping from the same source. If this is indeed true, there is no reason why one should try to change linguistic formalisms into a form more suitable for MT.

Linguistics has been, is still, and will continually be used in MT. What is currently been done is that linguistic knowledge, preferably expressed in formal terms using a linguistic formalism, is coded into a MT system by means of the SLLPs. SLLPs include formalisms like ATN, ROBRA, GRADE, METAL and Q- systems. Tree

structures are the main type of data structure manipulated in MT systems, and the SLLPs are mainly tree transducers, string-tree transducers and/or tree-string transducers. Such mechanisms are arguably very suitable for defining the analysis process (parsing a text to some representation of its meaning) and the synthesis process (generating a text form a given representation of meaning). SLLPs which work on feature structures have also been introduced, but these also work on the same principle.

Despite the fact that SLLPs are specifically designed for programming linguistic data, and that most of them separate the static linguistic data (linguistic rules) from the algorithmic data (the control structure), the problem is that they are still basically programming languages. Indeed, during the period of their inception, they may have been thought of as the MT's answer to a linguistic formalism, but it is no longer true these days. To begin with, most if not all SLLPs are procedural in nature, which means that a description can be read in only one direction (*not bidirectional*), either for analysis or for synthesis. Consequently, for every natural language treated in a MT system, two sets of data will have to be written: one for analysis and one for synthesis. Furthermore, also due to this procedural nature, linguistic rules in SLLPs are usually written with some algorithm in mind. Hence, although separated from the algorithmic component, these linguistic rules are not totally as declarative as one would have hoped (*not declarative*). For these reasons, as well as for the fact that SLLPs are very system oriented, data written in SLLPs are rarely retrievable for use in other systems (*not portable*).

It was due to these shortcomings that other formalisms for MT which are bidirectional, declarative and not totally system oriented have been designed. Such formalisms include the SG and its more formal version, the STCG. One first notes that these formalisms are not designed to replace linguistic formalisms. There may be some linguistic justifications (e.g. in terms of the linguistic model [Zaharin 87b], but

they are designed principally for bridging the gap between linguistic formalisms and SLLPs. Such formalisms are designed to cater for MT problems, and hence may not directly reflect linguistic hypotheses but simply have the possibility to express them in a manner more easibly interpretable for MT. They are declarative in nature and also bidirectional. Only one set of data is required to describe both analysis and generation. They are also general in nature, meaning that it is possible to express different linguistic theories using these formalisms, and also that it is possible to implement these formalisms using various SLLPs. One can view such formalisms as specifications for writing SLLPs, as illustrated in Figure 2 (akin to specifications used in software engineering).

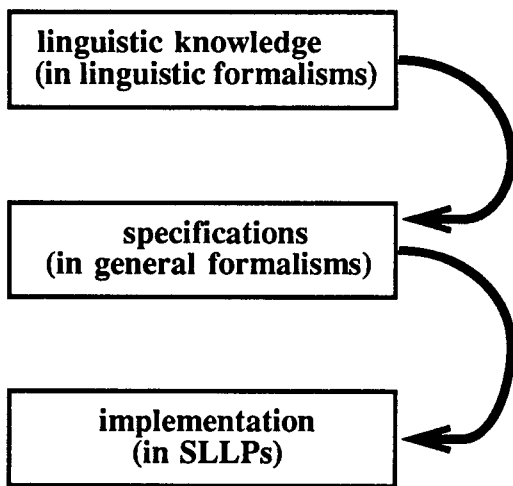Fig. 3 - the central role of general formalisms

Fig. 2 - General formalisms as specifications

Other formalisms that can be considered to be within this class of general formalisms are TAG, FUG, and perhaps DCG. With such formalisms, one may express knowledge from various linguistic theories (possibly a mixture), and that the same set of represented knowledge may be implemented for both analysis and synthesis using various SLLPs in different MT systems (as illustrated in Figure 3).
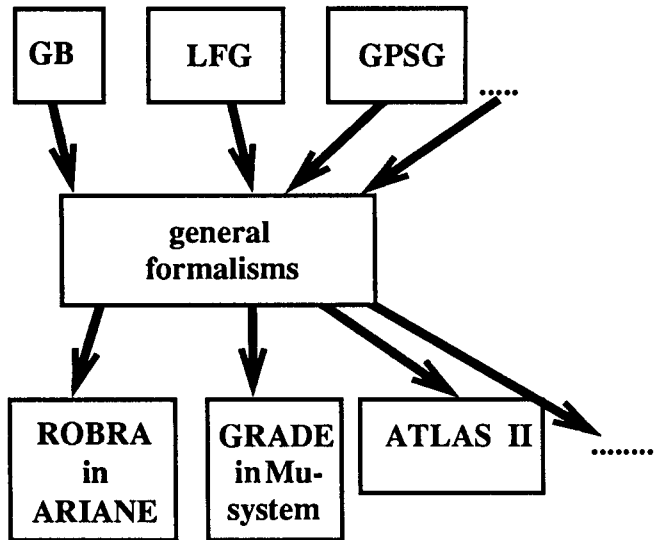
On specifications for analysis and synthesis

The two main processes in MT are analysis and synthesis (a third process called transfer is present if the approach is not interlingual). Analysis is the process of obtaining some representation(s) of meaning (adequate for translation) from a given text, while synthesis is the reverse process of obtaining a text from a given representation of meaning [1]. Analysis and synthesis can be considered to be two different ways of interpreting a single concept, this concept being a correspondence between the set of all possible texts and the set of all possible representations of meaning in a language. This correspondence is basically made up of a set of texts (T), a set of representations (S), and a relation between the two R(T,S), defined in terms of relations between elements of T and elements of S. We illustrate this in Figure 4.

S

```
      ┌─────────────┐
      │   Set  of   │
      │Representations│
      │      s      │
      └─────────────┘
```

relation between
texts and
representations
R(T,S) =
{R(T,S) : t ∈ T, s ∈ S}

T

```
      ┌─────────────┐
      │   Set  of   │
      │    Texts    │
      │      t      │
      └─────────────┘
```
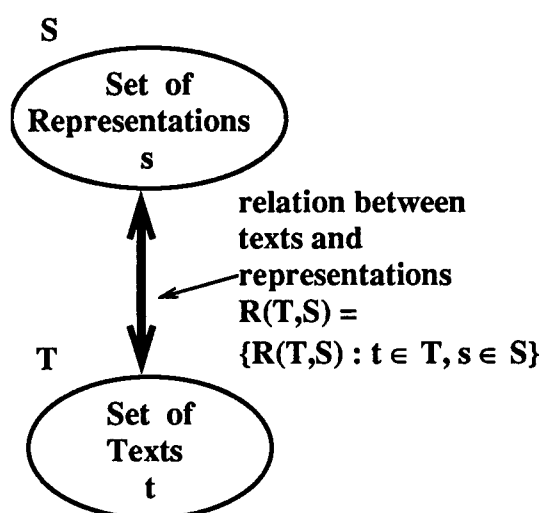
Fig. 4 - The correspondence between texts and their representations

Supposing that a correspondence as given in Figure 4 has been defined, analysis is then the process of interpreting the relation R(T,S) in such a way that given a text t, its corresponding representation s is obtained. Conversely, synthesis is the process of interpreting R(T,S) in such a way that given s, t is obtained. Clearly, a general formalism to be used as specifications must be capable of defining the correspondence in Figure 4.

Defining the correspondence may entail defining just one, two, or all three components of Figure 4 depending on the complexity of the results required. When one works on a natural language, one cannot hope to define the set of texts T (unless it is a very restricted sublanguage). Instead, one would attempt to define it by means of the definition of the other two components. As an example, the CFG formalism defines only the component R(T,S) by means of context-free rules. This component generates the set of texts (t) as well as all possible representations (S) given by the parse trees. The formalism of GB defines the relation R(T,S) by means of context-free rules (constrained by the Xbar-theory), move-α rules (constrained by bounding theory), the phonetic interpretative component and the logical interpretative component. This relation generates the

set of all texts (T) and all candidate representations (S) (logical structures). The set S is however further defined (constrained) by the binding theory, θ-theory and the empty category principle. As a third example, the STCG formalism defines R(T,S) by means of its rules, which in turn generates S and T. The set S is however further defined by means of constraints on the writing of the STCG rules.

Having set the specifications for analysis and synthesis by means of a general formalism, one can then proceed to implement the analysis and synthesis. Ideally, one should have an interpreter for the formalism that works both ways. However, an interpreter alone is not enough to complete a MT system : one has to consider other components like a morphological analyser, a morphological generator, monolingual dictionaries, and for non-interlingual systems, a transfer phase and bilingual dictionaries. In fact, such an interpreter alone will not complete the analysis nor the synthesis, a point which shall be discussed as of the next paragraph. For these reasons, the specifications given by the general formalism are usually implemented using available integrated systems, and hence in their SLLPs.

For analysis, apart from the linguistic rules given by the general formalism, there is the algorithmic component to be added. This is the control structure that decides on the sequence of application of rules. A general formalism does not, and should not, include the algorithmic component in its description. The description should be static. There is also the problem of lexical and structural ambiguities, which a general formalism does not, and should not, take into consideration either. A fully descriptive and modular specification for analysis should have separate components for linguistic rules (given by the formalism), algorithmic structure, and disambiguation rules. Apart from being theoretically attractive, such modularity leads to easier maintenance (this discussion is taken further in [Zaharin 88]); but most important is the fact the same linguistic rules given by the

formalism will serve as specifications for synthesis, whereas the algorithmic component and disambiguation rules will not.

In general, synthesis in MT lacks a proper definition, in particular for transfer systems[2]. It is for this reason (and other reasons similar to those for analysis) That the specifications for synthesis given by the general formalism play a major role but do not suffice for the whole synthesis process. To clarify this point, let us look at the classical global picture for MT in second generation systems given in Figure 5. The figure gives the possible levels for transfer from the word level up to interlingua, the higher one goes the deeper the meaning.

Interlingua

Semantic
Relations

Logical Relations

Syntactic Functions

Syntagmatic Classes

Lexical Units

Lemmas

Words

Source   Target
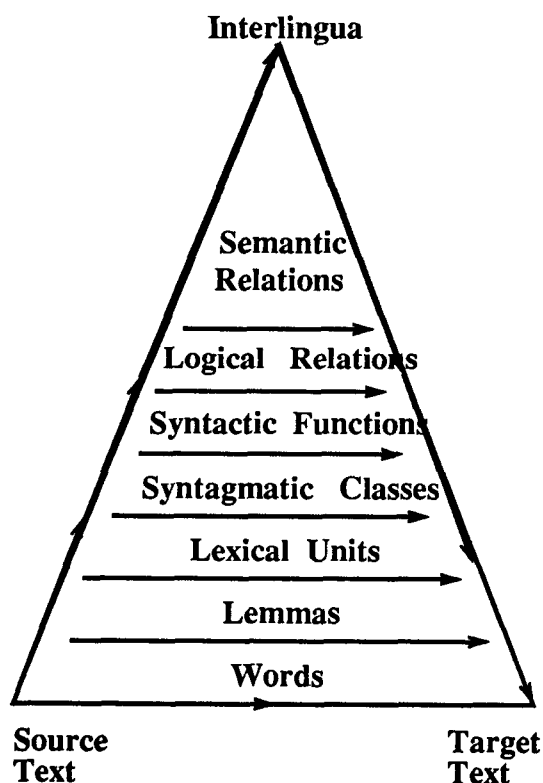Text   Text

Fig. 5 - The levels of transfer in second generation MT systems

Most current systems attempt to go as high as the level of semantic relations (eg. AGENT, PATIENT, INSTRUMENT) before embarking on the transfer. Most systems also retain some lower level information (eg. logical relations, syntactic functions and syntagmatic classes) as the analysis

goes deeper, and the information gets mapped to their equivalents in the target language. The reason for this is that certain lower level information may be needed to help choose the target text to be generated amongst the many possibilities that can be generated from a given target representation; the other reason is for cases that fail to attain a complete analysis (hence fail-soft measures).

The consequence to the above is that the output of the transfer, and hence the input to synthesis, may contain a mixture of the information. Some of this information are pertinent, namely the information associated to the level of transfer (in this case the semantic relations, and to a large extent the logical relations), while the rest are indicative. The latter can be considered as heuristics that helps the choice of the target text as described above. Whatever the level of transfer chosen, there is certainly a difference between the input to synthesis and the representative structure described in the set S in Figure 4, the latter being precisely the representative structure specified in the general formalism. In consequence, if the synthesis is to be implemented true to the specifications given by the general formalism (which have also served as the specifications for analysis), the synthesis phase has to be split into two subphases: the first phase has the role of transforming the input into a structure conforming to the one specified by the formalism (let us call this subphase SYN1), and the other does exactly as required by the general formalism, ie. generate the required text from the given structure (call this phrase SYN2). The translation process is then as illustrated in Figure 6.

As mentioned, the phase SYN2 is exactly as specified by the general formalism used as specifications. What is missing is the algorithmic component, which is the control structure which decides on the applications of rules. However, the phase SYN1 needs some careful study. Some indication is given in the discussion on some of our current work.
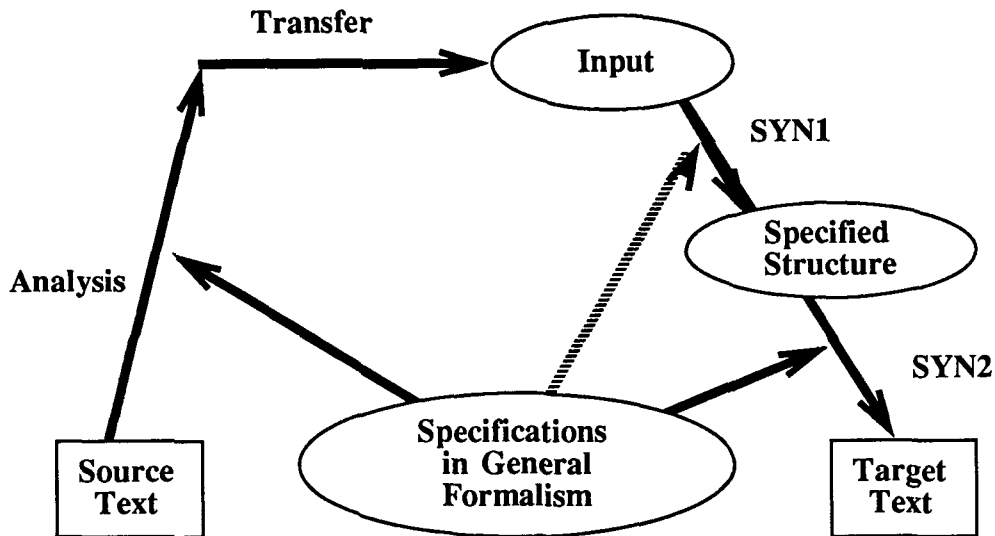
Fig.6 - The splitting of synthesis

Some relevant current work at PTMK-GETA

Relevant to the discussion in this paper, the following is some current work undertaken within the cooperation in MT between PTMK (Projek Terjemahan Melalui Komputer) in Penang and GETA (Groupe d'Etudes pour la Traduction Automatique) in Grenoble.

The formalisms of SG, and its more formal version STCG, have been used as specifications for analysis and synthesis since 1983, namely for MT applications for French-English, English-French and English-Malay, using the ARIANE system. However, not only the implementations have been in the SLLP ROBRA in ARIANE, the transfer from specifications (given by the general formalism) to the implementation formalism has also been done manually.

One project undertaken is the construction of an interpreter for the STCG which will do both analysis and generation. Some appropriate modifications will enable the interpreter to handle synthesis (SYN2 above). At the moment, implementation specifications are about to be completed, and the implementation is proposed to be carried out in the programming language C.

Another project is the construction of a compiler that generates a synthesis

program in ROBRA from a given set of specifications written in SG or STCG. Implementation specifications for SYN2 is about to be completed, and the implementation is proposed to be carried out in Turbo-Pascal. The algorithmic component in SYN2 will be automatically deduced from the REFERENCE mechanism of the SG/STCG formalism. The automatic generation of a SYN1 program poses a bigger problem. For this, the output specifications are given by the SG/STCG rules, but as mentioned earlier, the input specifications can be rather vague. To overcome this problem, we are forced to look more closely into the definitions of the various levels of interpretation as indicated in Figure 5, from which we should be able to separate out the pertinent from the indicative type of information in the input structure to SYN1 (as discussed earlier). Once this is done, the interpretation of SG/STCG rules for generating a SYN1 program in ROBRA will not pose such a big problem (the problem is theoretical, not of implementation - in fact, specifications for implementation for this latter part have been laid down, pending on the results of the theoretical research).

Concluding remarks

The MT literature cites numerous formalisms. The formalisms, can be generally classed as linguistic

formalisms, SLLPs and general formalisms. The linguistic formalisms are designed purely for linguistic work, while SLLPs, although designed for MT work, may lack certain desirable properties like bidirectionality, declarativeness and portability. General formalisms have been designed to bridge the gap between the two extremes, but more important, they can serve as specifications in MT. However, such formalisms may still be insufficient to specify the entire MT process. There is perhaps a call for more theoretical foundations with more formal definitions for the various processes in MT.

Footnotes

1. The term generation has sometimes been used in place of synthesis, but this is quite incorrect. Generation refers to the process of generating all possible texts from a given representation, usually an axiom, and this is irrelevant in MT apart from the fact that synthesis can be viewed as a subprocess of generation.

2. Interlingual systems may not lack the definition for synthesis, but they lack the definition for interlingua itself. To date, all interlingual systems can be argued to be transfer systems in a different guise.

References

Ch. Boitet - Automatic production of CF and CS-analyzers using a general tree transducer. *2. Internationale Kolloquium über Maschinelle Übersetzung, Lexicographie und Analyse*, Saarbrücken, 16-17 Nov. 1979.

J. Bresnan and R.M. Kaplan - Lexical Functional Grammar: a formal system for grammatical representations. In *The Mental Representation of Grammatical Relations*, J. Bresnan (ed), MIT Press, Cambridge, Mass., 1982.

N. Chomsky - *Lectures on Government and Binding* (the Pisa Lectures), Foris, Dordrecht, 1981.

A. Colmerauer - Les systèmes-Q ou un formalisme pour analyser et synthétiser des phrases sur ordinateur. TAUM, Université de Montréal, 1971.

G. Gazdar and G.K. Pullum - *Generalized Phrase Structure Grammar: a theoretical synopsis*. Indiana University Linguistics Club, Bloomington, Indiana, 1982.

A. Joshi, L. Levy and M. Takahashi - Tree Adjunct Grammars. *Journal of the Computer and System Sciences* 10:1, 1975.

M. Kay - Unification Grammar. Xerox Palo Alto Research Center, 1982.

M. Nagao, J. Tsujii, K. Mitamura, H. Hirakawa and M. Kume - A machine translation system from Japanese into English - another perspective of MT systems. *Proceedings of COLING 80*, Tokyo, 1980.

J. Slocum - METAL: The LRC machine translation system. ISSCO Tutorial on Machine Translation, Lugano, Switzerland, 1984.

B. Vauquois and S. Chappuy - Static Grammars: a formalism for the description of linguistic models. *Proceedings of the Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, Colgate University, Hamilton, NY, 1985.

D.H.D. Warren and F.C.N. Pereira - Definite Clause Grammars for language analysis. A survey of the formalism and a comparison with ATNs; *Artificial Intelligence* 13, 1980.

W.A. Woods - Transition Network Grammars for natural language analysis. *Communications of the ACM* 13:10, 1970.

Y. Zaharin - String-Tree Correspondence Grammar: a declarative grammar formalism for defining the correspondence between strings of terms and tree structures. *3rd Conference of the European Chapter of the Association for Computational Linguistics*, Copenhagen, 1987.

Y. Zaharin - The linguistic approach at GETA: a synopsis. *Technologos* 4 (printemps 1987), LISH-CNRS, Paris.

Y. Zaharin - Towards an analyser (parser) in a machine translation system based on ideas from expert systems. *Computational Intelligence* 4:2, 1988.