AN ENGLISH GENERATOR FOR A CASE-LABELLED DEPENDENCY REPRESENTATION

John Irving Tait
Acorn Computers Ltd.
Fulbourn Road
Cherry Hinton
Cambridge CB1 4JN
U. K.

Abstract

The paper describes a program which has been constructed to produce English strings from a case-labelled dependency representation. The program uses an especially simple and uniform control structure with a well defined separation of the different knowledge sources used during generation. Furthermore, the majority of the system's knowledge is expressed in a declarative form, so in priciple the generator's knowledge bases could be used for purposes other than generation. The generator uses a two-pass control structure, the first translating from the semantically orientated case-labelled dependency structures into surface syntactic trees and the second translating from these trees into English strings.

The generator is very flexible: it can be run in such a way as to produce all the possible syntactically legitimate variations on a given utterance, and has built in facilities to do some synonym substitution. It has been used in a number of application domains: notably as a part of a free text retrieval system and as part of a natural language front end to a relational database system.

1. Introduction

This paper describes a program which has been constructed to translate from Boguraev's case-labelled dependency representations (Boguraev, 1979: see also Boguraev and Sparck Jones, 1982) to English strings. Although the principles on which the program has been constructed are primarily a new mix of established ideas, the generator incorporates a number of novel features. In particular, it combines an especially simple and uniform control structure with a well defined separation of the different knowledge sources used during generation. It operates in two passes, the first translating from the semantically orientated case-labelled dependency structures into surface syntactic trees and the second translating from these trees into English strings.

The translation from dependency structures to surface syntactic trees is the more complex of the two passes undertaken by the generator and will be described here. The other, translation from instantiated surface trees to text strings is relatively straightforward and will not be dealt with in this paper. It is fundamentally a tree flattening process, and is described in detail in Tait and Sparck Jones (1983).

2. The Generator's Knowledge Structures

The generator's knowledge is separated into four sections, as follows.

1) a set of bare templates of phrasal and clausal structures which restrict the surface trees other parts of the system may produce by defining the branching factor at a given node type. For example, the patterns record that English has intransitive, transitive and ditransitive, but not tritransitive, verb phrases. The bare template for noun phrases is illustrated in Figure 1.

2) a lexicon and an associated morphological processor.

3) a set of production rules which fill out partially instantiated syntactic trees produced from the phrasal and clausal patterns. These rules contain most of the system's knowledge about the relationship between the constructs of Boguraev's representation language and English forms.

4) another set of production rules which convert filled out surface trees to English strings.

```
                  /-Quantifier
                  |-Determiner
                  |-Ordinal
Noun Phrase ==    |-Number
                  |-Adjective-list
                  |-Nominal-modifier-list
                  |-Head
                  \-Post-modifers
```

Figure 1
Template for Noun Phrase

These four knowledge sources represent the generator's entire knowledge of both English and Boguraev's representation language. Although they are obviously interrelated, each is distinct and separate. This well defined separation greatly

increases the extensability and maintainability of the system.

As noted in the previous section the application of the rules of section 4 will not be discussed in this paper. The remainder of the paper discusses the use made of the first three knowledge sources.

3. Translation from Dependency Structures to Surface Syntactic Trees

The primary work of conversion from the dependency representations to the surface syntactic trees is undertaken by a set of production rules, each rule being associated with one of the case labels used in Boguraev's representation scheme. These rules are applied by a suite of programs which exploit information about the structure of Boguraev's dependency structures. For example they know where in a nominal dependency structure to find the word sense name of the head noun ('oscillator1' in Figure 2) and where to find its case list (to which the production rules should be applied).

```
(n (oscillator1 THING
     (@@ det (the1 ONE))
     (## nmod
        ((((trace (clause v agent))
           (clause
              (v (be2 BE
                 (@@ agent
                    (n (frequency1 SIGN)))
                 (@@ state
                    (st (n (nameless NIL))
                    (val (high3 KIND))))
           ))) ))) )))
```

Figure 2
Boguraev Representation used for
"the high frequency oscillator"

It must be emphasized that Boguraev's use of the term case is much wider than is common in linguistics. Not only is it used to cover prepositional attachment to nouns as well as verbs; it is also used to cover some other forms of attachment to, and modification of, nouns, for example by determiners (like "a") and even for plural or singular number. In the phrase "the high frequency oscillator", whose representation is illustrated by Figure 2, the link between 'oscillator1' (standing for "oscillator"), and the determiner ('(the1 ONE)', representing "the") is the so-called case-label det. Similarly the prenominal modifier "high frequency" (represented by the complex structure to the lower right of the figure) is linked to 'oscillator1' by nmod.

Each case-associated production rule takes four inputs, as follows:

1) the dependent item attached to the case link, for example '(the1 ONE)' in the case of det given below;

2) an environment which is used to pass information from the processing of higher levels of the representation down to lower levels: for example tense from the sentential level into an embedded relative clause; the environment is also used to allow various kinds of control over the generation process: for example to determine how many paraphrases of a sentence are produced;

3) a partially instantiated phrase or clause template, which will ultimately form part of the surface syntactic tree output by the first pass of the generator;

4) the dictionary entry for the dominant item of the current case list: in Figure 2 this is the entry for 'oscillator1', presented in Figure 3.

```
(oscillator1
   (oscillator1-#1
      (root oscillator)
      (syntax-patterns Noun-phrase-pattern)))
```

Figure 3
Dictionary entry for 'oscillator1'

The rules vary greatly in complexity: the structure illustrated in Figure 2 requires the use of both the simplest and most complex form of rule.

The det production rule may be described in pseudo-English as:

> If the partially instantiated template is for a noun phrase then look up the lexical items (potentially synonyms) associated with the word sense name 'the1', and insert each in the determiner slot in a new copy of the syntactic node.

(Of course for English there is only one lexical item associated with 'the1': "the".) At the other extreme is the production rule for the nmod case. The nmod case in Boguraev's dependency structures is used to associate the pre-nominal modifiers in a compound nominal with the head noun. The pre-nominal modifiers are represented as a list of simple nominal representations.

```
(Noun-Phrase (NIL the NIL NIL NIL
       ((Noun-Phrase NIL NIL NIL NIL
            (high) NIL frequency NIL))
       oscillator NIL))
```

Figure 4
Surface Structure Tree for
"the high frequency oscillator"

In English the nmod production rule might be

expressed as:

> If the partially instantiated template is for a noun phrase, apply the processor which, given an existing nominal representation, instantiates a corresponding phrasal template, to each nominal representation in the dependent item list: form the results into a set of lists, one for each combination of possible results for expressing each nominal: insert each result list into a copy of the partially instantiated template originally passed to the rule.

The surface structure tree produced after these rules have been applied to the representation of Figure 2 is given in Figure 4. Note that the tree contains syntactic category names, and that unfilled slots in the tree are filled with NIL. Thus if the phrase to be generated was "all the high frequency oscillators", the first NIL in the surface syntactic tree (representing the unfilled quantifier slot of the dominant noun phrase node) would be replaced by "all". The order of the words in the surface syntactic tree represents the order in which they will be produced in the output sentence.

These two production rules, for the det and nmod case labels, are fairly typical of those used elsewhere in the system. There is, however, an important feature they fail to illustrate. In contrast with more conventional cases, nmod and det do not require the identification of a lexical item associated with the case-label itself. This is of course necessary when expressing prepositional phrases.

4. Distinctive Features of this Translation Process

The two most noteworthy features of the generation phase which produces surface structure trees are the control structure employed and distribution of the systems language knowledge between its different components.

No mention of the system's control structure was made in the previous section. The structure used is sufficiently powerful and elegant that it could be ignored entirely when building up the systems knowledge of Boguraev's representation language and of English. However, the efficiency of the generator described here is largely a result of the control structure used. It is rare for this system to take more than a few fractions of a second to generate a sentence: a sharp contrast with approaches based on unification, like Appelt's (1983) TELEGRAM.

First the current representational structure is classified as clausal, simple nominal, or complex (typically relativised) nominal. Second, a suitable structure dismantling function is applied to the structure which identifies the head lexical token from the structure and separates out its case-list. Third the dictionary entry for the head lexical item is obtained, and, after checking the syntactic markers in the dictionary entry and phrasal or clause templates suitable for the environment are identified. Fourth, appropriate production rules are applied to each element of the structure's case list in order to instantiate the templates. Frequently this whole process is applied recursively to some dependent representation level. So, for example, the representation for "high frequency" is processed by a second call of the noun phrase processor from within the call dealing with the dominant nominal, 'oscillator1'. When the case list has been completely processed, the dismantling function applies any necessary morphological processing to the head lexical item (for example to reflect subject/verb and person/number agreement).

This simple framework covers all the processing done by the generator.

The split between the syntactic knowledge represented in the phrasal and clausal templates and in the production rules is also unusual. The templates define the shape of the surface syntactic trees which the system can produce. It places no restrictions on the form of the fillers for any slot in a grammar node. The production rules enforce categorial and ordering restrictions. So, for example, the templates reflect the fact that English possesses intransitive, transitive and ditransitive verbs, whilst the production rules ensure that the subject of a clause is of a suitable syntactic category, and that the subject precedes the verb in simple declarative sentences.

The surface structure trees produced contain all the words in the sentence to be produced in the order and form in which they are to be output. Thus it is a straightforward matter to generate English strings from them.

5. Conclusion

The generator presented here is in essence a development of the Micro-Mumble generator described in Meehan (1981). But in the process of extending Meehan's framework for a wide coverage system, his original design has been radically transformed. Most notably, the system described here has its syntactic knowledge largely separated from its knowledge of the input representation language. It has, however, retained the elegant control structure of Meehan's original. This distinguishes it from the early generators in the same style, like Goldman's (1975) BABEL.

At the same time the generator described here is very flexible: it can be run in such a way as to produce all the possible syntactically legitimate variations on a given utterance, and has built in facilities to do some synonym substitution. The environment mechanism is very (perhaps too) powerful, and could be used to dynamically select possible ways of expressing a given structure in almost any way required.

The system's knowledge of natural language and of

the representation language is expressed in a fundamentally rule-like way, most notably without the use of an assignment mechanism. In principle such rules could be used backwards, that is they could be used to parse incoming English. However no work has been done to develop a parser which uses the generators rules, so this possibility remains pure speculation at present.

The generator described here, it must be emphasized, covers only part of the task of generation. Unlike, for example, McKeown's (1980) system, it deals not with what to say, but only with how to say it. Boguraev's representation identifies sentence boundaries and the majority of content words to be used in the utterance being produced (see Figure 1), making the task of the generator relatively straightforward. However, the techniques used could deal with a representation which was much less closely related to the surface text provided this representation retained a fairly straightforward relationship between propositional units of the meaning representation and the clausal structure of the language. For example, a representation language which represented only states and times, but not the events which linked different states and times would probably require a more powerful framework than that provided by the generator described here. However, another case-labelled dependency language, like Schank's (1975) Conceptual Dependency (CD) Representation, could be handled by providing the generator with a new set of syntactico-semantic production rules, a new lexicon and the replacement of the functions for dismantling Boguraev's dependency representation with functions for dismantling CD structures.

The framework of the generator has been completely implemented and tested with a lexicon of a few hundred words and a grammar covering much of the English noun phrase and a number of the more straightforward sentence types. It has been used in a number of applications, most notably document retrieval (Sparck Jones and Tait, 1984a and 1984b) and relational database access (Boguraev and Sparck Jones, 1983).

The program described here is efficient (rarely taking more than a few fractions of second to generate a sentence) in contrast with approaches based on complex pattern matching (like Appelt (1983), and Jacobs (1983)). On the other hand, the essential simplicity and uniformity of the approach adopted here has meant that the generator is no more difficult to maintain and extend than more linguistically motivated approaches, for example Appelt's. Thus it has demonstrated its usefulness as a practical tool for computational linguistic research.

ACKNOWLEDGEMENTS

This work was supported by the British Library Research and Development Department and was undertaken in the University of Cambridge Computer Laboratory. I would like to thank Bran Boguraev, Ted Briscoe and Karen Sparck Jones for the helpful comments they made on the first draft of this paper. I would also like to thank my anonymous referees for the very helpful comments they made on the an earlier draft of the paper.

REFERENCES

Appelt, D.E. (1983) TELEGRAM: A Grammar Formalism for Language Planning. Proceedings of the Eighth International Joint Conference on Artificial Intelligence. Karlsruhe.

Boguraev, B. K. (1979) Automatic Resolution of Linguistic Ambiguities. Technical Report No. 11, University of Cambridge Computer Laboratory.

Boguraev, B.K. and K. Sparck Jones (1982) A natural language analyser for database access. In Information Technology: Research and Development; vol. 1.

Boguraev, B.K. and K. Sparck Jones (1983) A natural language front end to data bases with evaluative feedback. In New Applications of Databases (Ed. Garadin and Gelenbe), Academic Press, London.

Goldman, N. (1975) Conceptual Generation. In Conceptual Information Processing, R.C. Schank, North Holland, Amsterdam.

Jacobs, P. S. (1983) Generation in a Natural Language Interface. Proceedings of the Eighth International Joint Conference on Artificial Intelligence. Karlsruhe.

McKeown, K.R. (1980), Generating Relevant Explanations: Natural Language Responses to Questions about Database Structure. Proceedings of the First Annual National Conference on Artificial Intelligence, Stanford, Ca.

Meehan, J. (1981) Micro-TALE-SPIN. In Inside Computer Understanding, R.C. Schank and C.K. Riesbeck, Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Schank, R. C. (1975) Conceptual Information Processing, North Holland, Amsterdam.

Sparck Jones K. and J. I. Tait (1984a), Automatic Search Term Variant Generation. Journal of Documentation, Vol 40, No. 1.

Sparck Jones, K. and J. I. Tait (1984b), Linguistically Motivated Descriptive Term Selection. Proceedings of COLING 84, Association for Computational Linguistics, Stanford.

Tait, J.I. and K. Sparck Jones (1983), Automatic Search Term Variant Generation for Document Retrieval; British Library R&D Report 5793, Cambridge.