The General Architecture of Generation in ACORD*

Dieter Kohl
Universität Stuttgart
Keplerstraße 17
D-7000 Stuttgart 1 (West Germany)

Agnes Plainfossé
Laboratoires de Marcoussis
Route de Nozay
91460 Marcoussis (France)

Claire Gardent
University of Edinburgh
2 Buccleuch Place
Edinburgh EH8 9LW (Scotland)

Abstract

This paper describes the general architecture of generation in the ACORD project. The central module of this architecture is a planning component, which allows to plan single sentences as an answer to a KB query. The planner works for three different languages (English, French and German) and for sentence generators based on two different grammar formalisms (UCG for English and French, LFG for German) independent of the particular grammar or grammar formalism. It uses several knowledge sources of the ACORD system to make its decisions. The output of the planner is used for the language specific generators as well as for the update of information needed for pronoun resolution.

1 Introduction

The ACORD prototype offers an interactive update and query of a knowledge-base (KB). In the query situation the user asks the KB using natural language and possibly graphical pointing. The final response composed of natural language and also if appropriate, graphical highlighting, is generated with a language specific generator, in the three ACORD languages (English, French and German) using the same grammar formalisms and grammars employed in parsing: UCG for English and French, and LFG for German. The generators are fully described in [9] for the UCG framework, and in [3] and [5] for the LFG framework.

The objective of this paper is to describe the modules common to the three languages, which build the semantics of the answer to be generated using the semantics of the question posed to the system, the dialogue history, and the KB answer.

2 The Semantic Representation

Most components in the ACORD system share a semantic representation language called InL (*Indexed Language* (see [8])). InL is based upon Kamp's Discourse Representation Theory (see [1] and [2]). The generators work on a derived representation called SynInL, which was designed during the project.

2.1 Resolution within InL

The parsers produce information which allows a central component, the *resolver*, to determine the possibilities of coreference between anaphoric expressions and their antecedents (see [7]). This additional information is incorporated into an InL expression in the form of *occurrence information* or *lists*, stating for every element which may be coreferential with some other element properties relevant for determining coreference. We refer to InL expressions which incorporate such information as *unresolved* InLs and to InL expressions where this information has been used to determine coreference (and thereafter removed) as *resolved* InLs.

2.2 The problems encountered using InL for generation

Planning and generation operate on a different but derivated semantics formalism called SynInL. Several reasons brought us to design and use SynInL as opposed to InL:

First, to work efficiently the UCG generators require that their input be *canonical* with respect to the respective grammars. Canonicity means that only those InL formulas are treated, which could be produced by the parser, but not all formulas, which are logically equivalent[1]. In the context of InL, the notion of canonicity cannot be formalized outside the grammar definition. We then needed a semantic language where canonicity could always be provided, even though an expression was produced without any grammar dependent information.

Second, the generator needs NP planning to control the generation of referring expressions (see [6]). In order to specify information about the type of NP to be generated, a representation is required which allows the encoding of syntactic information in addition to semantic information. Furthermore, the individual bits of semantics must be related to the syntactic structure. More generally speaking, we need a mechanism for modifying or structuring the semantic representation to be generated prior to generation. Standard InL, being a purely semantic representation language, is inadequate for encoding this syntactic information.

Third, and most importantly, all of this has to be achieved in a language-, grammar- and formalism-independent way[2].

3 Designing SynInL

3.1 State of the art

There is a main difficulty in the concept of planning-based generation systems which explains the monolithic nature of many systems described in the relevant literature. If a planner plans a particular type of syntactic structure in the absence of grammatical information, there is no guarantee that the structure specified will actually be accepted by the grammar as being well-formed.

There are basically two solutions to this problem. One is to simply assume that the planner only specifies structures from which it will be always possible to generate. This works perfectly when there are no interactions between structures specified locally. An example of a grammar formalism with this "locality" property is the context free languages. However, for most modern approaches to grammar (including Government and Binding theory (GB) and all unification-based grammar formalisms), the locality property does not hold. In this case, we have to assume that the grammar is "loose enough" that anything we might plan can in fact be generated despite any interactions. Such a planning could

*The work reported here has been carried out as part of the ESPRIT project P393 ACORD on "The Construction and Interrogation of Knowledge-Bases using Natural Language Text and Graphics".

[1]To determine whether two syntactically distinct InL expressions are logically equivalent under laws such as commutativity and associativity is factorial in complexity.

[2]Language independence must be viewed as language independence regarding French, English and German.

be done deterministically. However, using this approach such a planner would always run the risk that it would fail to generate due to inconsistencies with the grammar.

The second solution is to interleave planning and generation and allow the possibility that failure to generate, results in different planning choices. Such systems also exist, although they seem to be comparatively recent in the literature. We did not investigate this possibility since it requires a fairly tight integration of planner and (grammar and formalism specific) generator which seems inconsistent with our requirement that we generate with three languages and two grammar formalisms.

3.2 Description of our approach

Our solution is to attempt an independent level of syntactic representation which abstracts away from the peculiarities of the surface syntactic structures of particular languages and deals directly with syntactic notions which are *language-independent*. Whether one thinks that this is possible, depends to a large degree on one's particular theoretical perspective.

What might such an "abstract" syntactic representation look like? There are several related concepts in diverse linguistic theories which satisfy the criteria. The most directly related concept is that of *D-structure* in GB. D-structure is a level of syntactic structure which mirrors semantic functor-argument structure directly (via the θ-criterion and the Projection Principle) and which is also related to surface syntactic structure by the rule of *Move-*α, a transformation that moves constituents from one position to another. Related notions of structure which capture the relation between semantic functor-argument structure (or predicate-argument structure) and "abstract" or "deep" syntactic structure are the *f*-structures of LFG and the grammatical function hierarchy-based accounts of subcategorisation in HPSG and UCG. All of these have the desirable property that they express a level of representation which relate subcategorisation, semantics and surface structure.

By using such representations which are hypothesized to be "linguistically universal" to associate partial semantic representations with abstract syntactic constituents, we also solve the other requirements mentioned above. First, most instances of noncanonicity are eliminated because sub-formulas are associated directly with syntactic constituents. Second, quantifier scope readings are eliminated from consideration at this level of representation. Third, since the level of representation is taken to be universal, there are language-dependent maps from the representation to surface syntactic structure.

3.3 SynInL description

The approach taken here is to encode syntactic structure in terms of schematic $\overline{\text{X}}$ theory familiar from most modern generative grammar formalisms. As mentioned above, this is most similar to D-structure in GB theory. SynInL expresses both syntactic and semantic information.

Idealizing considerably, SynInL formulas consist of four types: *heads, complements, modifiers* and *specifiers*. This corresponds directly to the standard constituent types in $\overline{\text{X}}$ theory. (We follow LFG *f*-structure and UCG subcategorisation structure in treating subjects as ordinary complements rather than specifiers of clauses). These four categories are ideal for attaining a level of language-independence in linguistic description and are general enough that it is reasonable to expect that such $\overline{\text{X}}$ representations can be mapped onto language-dependent surface syntactic structures.

The idea then is to encode this $\overline{\text{X}}$ structure in SynInL formulas. *Specifiers* in SynInL are of the general form:

```
specifier(Semantics, Head)
```

That is, they specify their own semantics and the properties of their head.

Heads are of the general form:

```
head(Semantics, ArgList, AdjunctList)
```

That is, they specify their own head semantics and a list of arguments and adjuncts which are also either specifier or head structures.

All of these structures also allow the encoding of syntactic requirements on arguments and adjuncts. However, there is no indication of either surface syntactic order of the complements and adjuncts or of the relative scope of quantifiers occurring in either complements or modifiers. The language generators are free to realize both scope and surface syntactic structure in any way which is consistent with the SynInL specification.

How is this augmented representation built? The parsers produce unresolved InL. This InL contains enough syntactic information for a unique mapping into an equivalent SynInL expression. This mapping is done by the InL $\rightarrow$ SynInL module.

Given an InL expression, it distinguishes between structural and prime predicates. For prime predicates there is always a mapping into a SynInL formula with a unique category. The structural predicates then determine how to merge the SynInL formulas which replace the original partial InL expression.

4 The Planning Component

The role of the planning component is to produce SynInL expressions from which phrases can be generated by the language specific generators and to decide whether any objects on the screen have to be highlighted.

Within ACORD, the planner gets as input the SynInL expression corresponding to the user question (yes/no question, wh-question or 'how much'/'how many'-question) and the KB answer. The planner output consists of an optional canned text marker and the SynInL of the answer to be generated.

The planner uses three sub-planners for planning verb phrases, NPs and modifications.

4.1 Architecture of the generator

The answer process consists of the following steps:

- The question is parsed. The output is the InL representation of the question with information for resolution.
- This InL expression is transformed into SynInL by the InL $\rightarrow$ SynInL module and also resolved using the occurrence information by the resolver. The resolver provides the generator with information which encodes the user's question as understood by the system.
- The resolved InL is passed on to the KB which provides the KB answer.
- The planner module takes as input the SynInL expression of the query and the KB answer. Depending on the type of questions asked, the planner makes decisions such as: what kind of canned text prefix is needed, what type of NP planning is necessary, what kind of answer is expected and what type of processing can be done on this answer. It calls the NP sub-planner in order to process all the NPs appearing in the question, as well as the KB answer which is transformed into an appropriate SynInL expression (generally an NP). The output

of the planner is a SynInL representation of the answer.

- The SynInL answer is the input to the language specific generator of the current language. The selected generator produces the final answer.

4.2 Planning the Structure of Verb Phrases

Within the ACORD lexicon, verbal predicates may only take arguments which refer to objects. This means that we do not do any planning for arguments which denote events or states, i.e., verbal or sentential complements. Consequently we only distinguish between two types of predicates: the copula, which only takes a subject and a noun phrase or PPs as complement, and all other verbs.

Other active verb forms take either one, two, or three arguments. The first argument always corresponds to the subject (in an active clause), the second to the object or a prepositional complement, and the third to the second object or a prepositional complement.

Given a list of arguments, the verb planner calls the NP planner on each argument, providing information relative to the function of the argument position under scrutiny, its position in the argument list, and the subject of the sentence in which the argument occurs.

The list of modifications of the original query (if any) is processed last. For each element of this list a call to the modification sub-planner is made.

4.3 Planning Noun Phrases

The planning component is responsible for providing the best expression for NPs. It uses the dialogue history as well as KB knowledge to decide whether to adopt a pronominalization strategy, or find a non-pronominal description for the NP under analysis.

The NP planner must be provided with enough information to decide whether and which kind of pronominalization is allowed, and whether a name could be used instead of a pronoun where such an option is available. It must also decide when to use demonstratives, definite or indefinite articles, and whether a complex description should include relative clauses and adjuncts. In addition, our planner has to decide which objects should be highlighted on the screen.

To do so, the NP planner needs a fully specified discourse referent and information about the syntactic environment of the NP to be produced.

The output of the NP planner is a fully specified SynInL expression, a possible extension of the list of objects to highlight on the screen, a possible extension of the list of local antecedents, and a possible change of the information corresponding to the answer in the event that the NP planner has produced the NP for the answer.

4.4 Planning modifications

Modifications appear either in the context of a verb or in the context of an NP. They express negation, PPs, relative clauses, adjectives and adverbs. The modification planner is currently handling relatives and PPs.

In the case of a relative clause, the identifier of the object of the verb is set to the NP discourse referent, and the verb planner is called.

In case of a PP with exactly one argument, if this argument is in the focus of a *wh*-question, the KB answer has to give both the internal name and the new argument of the preposition. If the answer is *no*, the planner fails, since we currently don't have a semantic definition for the various PP negations like 'nowhere' or 'never'. The overall result is then the canned text *I don't know*. Otherwise there is in general a list of adjunct-argument pairs. For each pair a SynInL expression for the preposition is generated, calling the planner recursively on the argument (pronominalization is not allowed in the context of a PP). If there is more than one pair in the list, a PP coordination is initialized and reduced as will be explained below.

Coordinated PPs are allowed to appear in answers. A list of SynInL expressions for PPs can be reduced, if the same preposition is used more than once, and the prepositional arguments are not demonstrative pronouns. The resulting SynInL expression contains the common preposition, and an NP coordination corresponding to the arguments of the former SynInL expressions. The NP coordination then can also be reduced as described in [4].

5 Conclusion

Generation in ACORD demonstrates how planning can be done for several languages with a minimum of language-specific information. The basis of our approach is the concept of SynInL which encodes language-independent syntactic information in addition to semantic information. A SynInL expression can be derivated from an InL expression using a deterministic process.

Language-specific dependencies are still necessary concerning gender and the syntactic function of NPs. They could be reduced further by adopting a slightly different architecture concerning the interelation of the generator and the resolver.

References

[1] **Kamp, H. [1981]** *A Theory of Truth and Semantic Representation.* In: Groenendijk, J.A. et. al. (eds.), *Formal Semantics in the Study of Natural Language*, Vol. I, Amsterdam 1981.

[2] **Kamp, H. and Reyle, U. [1990]** *From Discourse to Logic.* Reidel Dordrecht, to appear.

[3] **Kohl, D. [1988]** *Generierung funktionaler Strukturen aus einer semantischen Repräsentation.* Diplomarbeit. Institut für Informatik. Universität Stuttgart.

[4] **Kohl D., Plainfossé A., Reape M., Gardent C. [1989]** *Text Generation from Semantic Representation.* Acord deliverable T2.10

[5] **Momma, S. and Dörre, J. [1987]** Generation from *f*-structures. In: E. Klein and J. van Benthem (eds.) *Categories, Polymorphism and Unification*, Centre for Cognitive Science, University of Edinburgh.

[6] **Reape, M. and Zeevat, H. [1988]** Generation and Anaphora Resolution. Manuscript. Centre for Cognitive Science, University of Edinburgh. In: Institut für Maschinelle Sprachverarbeitung (eds.) *Extension of the Anaphora Resolution.* ACORD (P393) Report T1.7'b, Universität Stuttgart, March, 1989.

[7] **Zeevat, H. (ed) [1988]** *Specification of the Central Pronoun Resolver.* ACORD Deliverable T1.7'(a). Stuttgart 1988.

[8] **Zeevat, H. [1986]** *A Specification of InL.* Internal ACORD Report. Centre for Cognitive Science, Edinburgh 1986.

[9] **Zeevat H., Klein, E. and Calder, J. [1987]** An Introduction to Unification Categorial Grammar. In: Haddock, N.J., Klein, E. and Morril, G. (eds.) *Edinburgh Working Papers in Cognitive Science, Vol. 1: Categorial Grammar, Unification Grammar and Parsing.*