ALE FOR SPEECH: A TRANSLATION PROTOTYPE

*Gerald Penn**
SFB 340
Kl. Wilhelmstr. 113
72074 Tübingen, Germany
gpenn@sfs.nphil.uni-tuebingen.de

Bob Carpenter
Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974, USA
carp@research.bell-labs.com

In this paper, we describe The Attribute Logic Engine (ALE) and enhancements to it that enable it to serve as a complete grammatical infrastructure for applications such as spoken language translation. We indicate how ALE was expanded and combined with off-the-shelf speech components to develop an application that translates English speech to German speech. The translation operates by way of a semantic representation based on typed feature structures, with information on thematic roles ("who did what to whom") and agreement information that can be used to guide search in less restricted domains and map expressions to more felicitous (but semantically equivalent) constructions in the target language than a more literal, surface-oriented method would admit.

INTRODUCTION

The goal of the present work was to provide an illustration of how large-scale dialogue systems can incorporate and benefit from a fine-grained representation of meaning, and dispel the myth that such systems are inherently inefficient. By "fine-grained" representation, we specifically refer to one in which participants and their relation to each other as described in a sentence, i.e., "who did what to whom," are explicitly represented along with some syntactic information, such as gender and number. With the right representation and the right computational treatment of it, dialogue systems that avail themselves of meaning representations not only can operate at a competitive level of efficiency, but can actually stand at an advantage over shallower approaches, e.g., cosine similarity measures, aligning parallel texts, in applications with less restricted or unrestricted domains, where the semantic representation itself can actually be used to guide search. It is also of benefit in applications that, by their own nature, require the interpretation or preservation of meaning down to a very fine level, such as in machine translation or, in certain domains, query formulation. Such a representation can also be used as an interlingua to improve the scalability of an application across multiple languages.

The application examined further here is an English-to-German speech-to-speech machine translation system. The system is intended for use in medium-scale domains that require robustness and precise semantic representations, and for easy portability across such domains. A typical system in which this could be embedded, for example, would be telephone transaction system, such as an automated telephone banking center. It is, in fact, comparable in its intentions to the system(s) currently under development in the Verbmobil project, only it was achieved here using off-the-shelf components that are already available. The system was built around the Attribute Logic Engine (ALE) [5] both for parsing English input into a semantic representation based on the logic of typed feature structures, and for concept-based generation in German using two closely related grammars, one for each language. The logic, ALE, the grammars, and their integration with speech components are described in further detail in the sections that follow.

TYPED FEATURE STRUCTURES

Typed feature structures [1] are a generalization of the frames found in artificial intelligence classifiers and partial record structures in databases. They were first introduced by linguists to characterize natural language grammars in terms of well-formedness constraints [7]. The logic of typed feature structures is strongly typed, with the types being arranged in a meet semi-lattice that organizes or classifies the information that the types encode. Less specific types in the lattice can also be used as a logical means of vaguely specifying one or more possible types that can be refined later without non-deterministic search. For example, in Figure 1, the type *non_dat* can be used to describe all possible grammatical cases except the dative (which, in certain constructions in German, must be treated specially), and can be refined by unification later to any one of the other cases.

Typed feature structures can also bear features. Every meet semi-lattice of types must come with a set of *appropriateness conditions*, that specifies, for each type, the set of features for which every object of that type can and

* We gratefully acknowledge Bernd Möbius and Jialin Zhong at Bell Laboratories for providing the face animation and German TTS, respectively, that were used in this system.

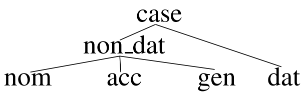

Figure 1: A type semi-lattice for grammatical case.

$$\begin{bmatrix} \text{throwing} & \\ \text{THROWING} & \begin{bmatrix} \text{index} & \\ \text{PERSON} & \boxed{1}\ \text{person} \\ \text{NUMBER} & \text{singular} \\ \text{GENDER} & \text{masculine} \end{bmatrix} \\ \text{THROWN} & \begin{bmatrix} \text{index} & \\ \text{PERSON} & \boxed{1} \\ \text{NUMBER} & \text{singular} \\ \text{GENDER} & \text{neuter} \end{bmatrix} \end{bmatrix}$$

Figure 2: A semantic representation of throwing with structure-sharing.

must have a value. As a result, while objects can acquire more features as their type is refined, all of the objects of a given type have the same arity. This gives typed feature structures many of the advantages that underspecified structures can have, such as a compact representation and a terse description language, along with many of those of fixed arity terms such as Prolog terms, in which the exact number and position of feature values in an internal representation can be determined at compile-time.

Typed feature structures also allow for *structure-sharing*, a kind of coreference that allows for further underspecification, in that two features can be said to have the same value without having to commit to what that value is. Figure 2, for example, illustrates a feature structure of type *throwing*, with appropriate features, THROWER and THROWN — all throwing events must have these. Their values' respective PERSON features have the same value, as indicated by the numeric tag, although it is still unknown whether this value is of type *first*, *second*, or *third*, as indicated by the less specific type, *person*.

THE ATTRIBUTE LOGIC ENGINE

ALE is a logic programming language very similar to Prolog, except that its terms are typed feature structures. It is freely available, fully documented, and has been widely adopted (over 150 research centers and universities) for implementing feature-based grammars, The combination of typing and appropriateness allows ALE to compile extensively all of the basic operations it performs on typed feature structures [4]. The most important basic operation is unification, the consistent combination of partial information from two or more feature structures.

ALE not only can perform the same resolution method as Prolog over programs, but also has a built in bottom-up chart parser for tabulating grammatical substrings [6] and a semantic-head-driven generator [9]. Both of these are guided by (reversible) phrase structure rules, in which structure-sharing enforced by unification is used to combine pieces of partial information from feature-structure representations of subphrases to compose a representation of a larger phrase.

ALE is implemented in Prolog and compiles all basic operations over feature structures into Prolog code, which is then submitted to a Prolog compiler. ALE compilation can thus be viewed as a preprocessing step similar to what YACC performs with grammars before C compilation. Since ALE 3.0 was released in early 1998, ALE has been enhanced in a number of ways that improve its suitability for large-scale applications. Perhaps most importantly, its more extensive compilation at the Prolog level and better indexing of predicates has a greatly improved performance on both large-scale parsing and large-scale generation. That compilation includes tracking the binding of ALE variables to determine where structure-sharings can be made safely at compile-time, and compilation of ALE predicates all the way down to parallel Prolog predicates to eliminate the overhead of a run-time meta-interpreter. ALE also compiles its parsing code so that all tabulated substrings can be maintained on the heap to avoid excessive copying of potentially large feature structures from Prolog's internal database.

ALE also incorporates a numerical component. Any number can serve as a featureless, maximally specific type in an ALE feature structure. These numbers can be probabilities that can be used to resolve non-determinism preferentially, or "goodness" estimates to compute thresholds for a beam-filtered search during parsing or generation. In fact, typed feature structures can also be used at a lower linguistic level as representations of candidate phonemic words in a word lattice or representations for phones themselves. We did not train a statistical model at any level for the present machine translation component, however.

ALE has a Prolog-based module system so that more than one grammar can be used at once. It has also has universal implicational constraints with type antecedents for a limited degree of constraint logic programming functionality, hooks to Prolog, and a macro facility, as well as several tools for handling large lexica, including a lexical rule formalism.

GRAMMARS

Both the English and the German grammars were implemented using a hybrid of analyses taken from Head-driven Phrase Structure Grammar (HPSG) [8] and multi-modal categorial grammar [2, 3]. The semantic representations themselves consist of a feature-based encoding of terms from the untyped lambda calculus, in which argument positions conventionally encode the semantic roles of participants, e.g., the innermost argument of a verbal phrase's semantics always stands for the agent, and functional symbols themselves are chosen from semantic constants (types) to provide lexical semantic information.

In keeping with categorial grammar, only a few basic syntactic categories were assumed, but were closed under the functional categories of categorial grammar to provide the categories of heads that take arguments to either the left (\) or right (/). For example, intransitive verbs can normally be assigned a category $np\backslash s$ because they take noun phrases to their left in a string to form sentences. In addition, appropriate features were used to attach syntactic agreement features where needed, to attach the semantic labellings from the lambda calculus to their phrases' representations, as well as to mark barrierhood for quantificational and wh-islands.

The grammars handle quantification, interrogatives and relative clauses by means of a higher-order empty category that mimics the effect of multi-modal categorial grammar analyses of long-distance phenomena. They also incorporate an analysis of coordination that is completely integrated with these effects, as well as a productive grammar of number terms.

German was chosen as the target language in order to avoid having to cope with the freer word order of German syntax. The German grammar thus generates (and recognizes) expressions within a limited but grammatical range of word-order possibilities.

Both grammars consist of roughly 150 forms, closed under inflectional paradigms and the production rules of the number term grammar.

INTEGRATION

The English grammar's lexicon was exported by ALE at compile-time and converted by the Bell Labs Lexical Tools to produce English pronunciations. Total compilation time, including loading the system, compiling both grammars, and this conversion takes approximately 5 minutes on an single SGI Indy.

For speech-to-speech translation, ALE's parser was configured to take a list of n best paths through a word lattice, and successively parse each one in order with the English grammar until one is found with a grammatical parse. Once a path with a grammatical parse is found, ALE will backtrack through all possible parses for that path, if necessary.

Speaker independent continuous speech recognition for English is handled by Entropics Cambridge Research Labs' (ECRL) HTK API. We used the ECRL context-independent phoneme models for American English, which provide a standard set of phonemic representations, with 17 vowels and 24 consonants in addition to silence models. The only difference from the set of phones provided by the Bell Labs TTS system is that the ECRL set includes the dipthong 'oi' as in 'noise'. The signal processing involved 12 mel cepstral features, 12 Δ features, 12 Δ^2 features and one total energy feature, with cepstral mean subtraction. A 10ms frame rate was used smoothed with a 25ms Hamming window.

The 25 best paths through resulting lattice are piped to ALE and parsed as described above. The resulting feature structure — our fine grained representation, containing syntactic and semantic information — is then passed to ALE's generator, which produces annotated text with the German grammar. Although we did not do so for our prototype, another potential use for ALE is in direct content-to-speech generation where a more sophisticated markup language for the output could be used to guide the synthesizer.

German synthesis from the text ALE generates is carried out using the Bell Labs Multilingual Text-to-Speech system [10], enhanced with several text tools[1] and synchronized with a simple talking head.

EVALUATION

The entire system runs in real time on a single SGI Indy. Because the null model was used with HTK, the speech recognition phase is by far the slowest. The benchmarks given here are for the text-to-text translation only, i.e., the part that ALE itself was responsible for; and were made on a dual-250-MHz SPARC Ultra 450 with 512 MB of RAM.

There are two kinds of performance improvements that have resulted from this research program to be measured. The first (Figure 3) is the relative improvement in parsing time between the present English grammar (CG/HPSG) and the HPSG grammar distributed with ALE, a naive, straightforward implementation of the English grammar presented in the first five chapters of Pollard and Sag's seminal book on HPSG [8] with roughly the same coverage. Both grammars are highly lexicalized, with large feature structures (between 100 and 200 nodes), and only a handful of phrase structure rules (4 for the new grammar, 9 for the HPSG grammar). The naive HPSG grammar, however, massively overgenerates with respect to its representation of quantifier scoping effects, and relies heavily on recursive data types, which results in much slower performance. The parsing times are taken from the improved version of ALE, to be released soon as ALE 3.2, over 11 sentences of various parsing complexities. The number of edges in the parsing chart is also provided (Figure 4) for each grammar as a measure of parsing complexity and overgeneration. The integration of categorial-grammar-style analyses is directly responsible for the improvement, which ranges from a factor of 5.7 on the smallest parse to a factor of 323.5 on the largest. The comparison cannot be made for generation, because the HPSG grammar's analysis of quantification is not easily reversible.

The second performance improvement has been in the system itself. To measure this, we consider both grammars running in ALE 3.0 on the same benchmarks, also shown in Figure 3. The performance improvement between systems ranges between factors of 3 and 5.8 on the

[1] The text tools, however, are not distributed with the Bell Labs Multimedia TTS system that is available for purchase. These are the only part of the system that is not off-the-shelf.

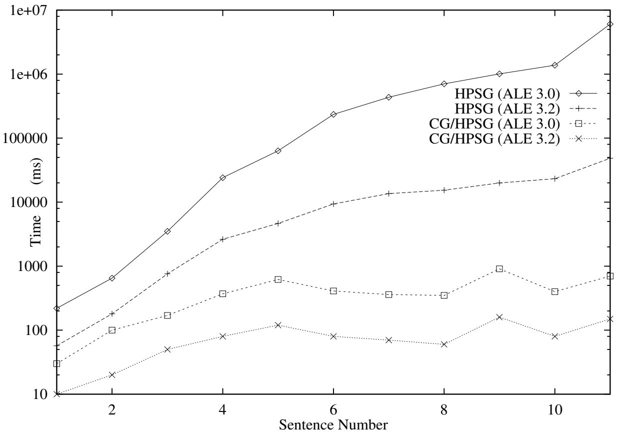

Figure 3: Grammar/System Performance: Time.

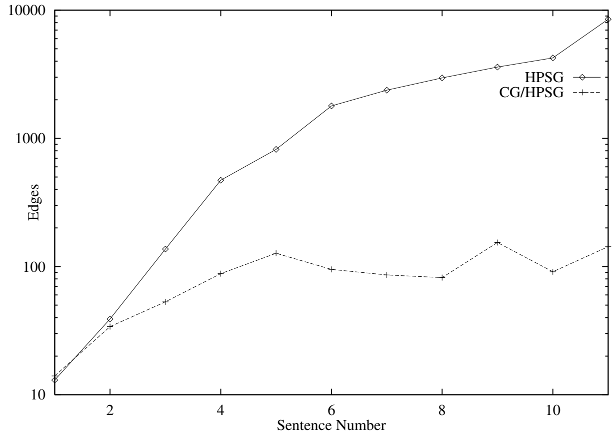

Figure 4: Grammar Performance: Edges.

CG/HPSG grammar and of 3.6 and 124.7 on the HPSG grammar. These factors do not include the relative improvements between the grammars. On the largest parse, the new CG/HPSG grammar running on ALE 3.2 has a combined improvement over the HPSG grammar running on ALE 3.0 of a factor of 40,340.4.

The difference in performance between system versions made in semantic-head-driven generation has been far more modest, to a great extent because ALE 3.0 used an indexing strategy for semantic values of lexical entries that is currently not supported in the new version of the system. For lack of space, a more detailed comparison cannot be made here; but the performance of the generator over the German grammar on ALE 3.2 over a test suite of 64 semantic descriptions of grammatical sentences has yielded generation times of up to 820 ms with an average time of 214 ms.

CONCLUSION

We have shown that ALE can be integrated with off-the-shelf speech components to construct an English-to-German speech-to-speech machine translation component over a significant grammar fragment. The transfer between grammars is through a fine-grained representation of meaning based on the logic of typed feature structures. The slowest part of the system was speech recognition because of our election to use context-independent phoneme models; but even then, the performance of the system has proven to run in real time on available hardware.

One direction that must be investigated in the near future is the use of ALE's numerical component for probabilistic language modelling for better parsing and generation. The integration of statistically derived weights is necessary to improve the efficiency of non-determinism resolution further. An important related issue is the automated acquisition of feature-structure-based grammars and categorial grammars through machine learning techniques.

REFERENCES

[1] B. Carpenter. *The Logic of Typed Feature Structures*. Cambridge, 1992.

[2] B. Carpenter. A type-logical grammar for German, 1998. Current Topics in Constraint-based Theories of Germanic Syntax, ESSLLI' 98, Saarbrücken.

[3] B. Carpenter. *Type Logical Semantics*. MIT Press, 1998.

[4] B. Carpenter and G. Penn. Compiling typed attribute-value logic grammars. In H. Bunt and M. Tomita, editors, *Recent Advances in Parsing Technologies*. Kluwer, 1996.

[5] B. Carpenter and G. Penn. *ALE 3.1 User's Manual*, September 1998. Available from the ALE Homepage: `http://www.sfs.nphil.uni-tuebingen.de/~gpenn/ale.html`.

[6] G. Gazdar and C. Mellish. *Natural Language Processing in PROLOG: an introduction to computational linguistics*. Addison-Wesley, 1990.

[7] C. Pollard and I. Sag. *Information-based Syntax and Semantics*. CSLI Publications, 1987.

[8] C. Pollard and I. Sag. *Head-driven Phrase Structure Grammar*. Chicago, 1994.

[9] S. M. Shieber, C. N. Pereira, G. van Noord, and R. C. Moore. Semantic-head-driven generation. *Computational Linguistics*, 16(1):30–42, 1990.

[10] R. Sproat. *Multilingual Text-to-Speech Synthesis: the Bell Labs Approach*. Kluwer, 1997.