CAT2 - Implementing a Formalism for Multi-Lingual MT

Randall Sharp
IAI
Martin-Luther-Strasse 14
D-6600 Saarbrücken
West Germany

randy%iaisun%sbsvax.uucp(3)Germany.CSnet

Abstract

Following on research that resulted in the translation methodology known as the <C,A>,T framework, the present paper reports on the further development and implementation now referred to as CAT2. The basic philosophy is the creation of separate grammars, in homogeneous notation, for multiple levels of linguistic representation, and a means of describing relations between the levels. Each level is represented as a derivation tree generated by a context-free grammar augmented by feature lists. The translation relations are recursive tree-to-tree mappings. The basic operation is unification, making CAT2 similar to unification-based formalisms. As a development tool, it shares many characteristics of PATR-II. The descriptive nature of the framework motivates the use of Prolog as the implementation language. The report describes the CAT2 formalism and its implementation. Some comparisons are made between CAT2 and similar frameworks. An example is presented showing the basic strategy in analysis, transfer, and generation. The conclusion is reached that although the system is still in a primitive stage of development, it is already adequate for handling some complex translation issues in a linguistically motivated way, and the simplicity of its design allows for a controlled incremental refinement.

CAT2 - Implementing a Formalism for Multi-Lingual MT

One of the many major contributions of the EUROTRA MT project to the scientific investigation of automatic translation is the instigation of parallel research, both within the official EUROTRA community and in independent centers. Examples of the latter include joint research projects in Stuttgart and Berlin, the former applying MT concepts within the formalism of LFG (Kaplan & Bresnan 1982; Netter & Wedekind 1986), and the latter within the GPSG formalism (Gazdar, *et al.* 1985; Hauenschild & Busemann forthcoming). Here in Saarbrücken we are extending both the implementation and the formalism of the <C,A>,T framework (Arnold, *et al.* 1985,1986; Arnold & des Tombes 1987), the predecessor of the current EUROTRA framework. We call our experimental implementation, appropriately, CAT2.

The approach taken in CAT2 is similar to virtually all MT systems: posit levels of representation between the source and target texts such that translations between intermediate levels is simpler than a direct text-to-text translation. The levels usually correspond to stages of syntactic and semantic analyses. What CAT2 offers is a formalism for describing levels of representation and a means of relating adjacent levels. That is, CAT2 is simply a formalism for describing grammars and translators.

This very general formalism does not restrict or otherwise prescribe the number or interpretation of levels of representation. In our research we work with three levels: one based on surface syntactic structure, another on grammatical functions, and a third on semantic role relations. We could instead choose a surface structure/deep structure division. Furthermore, the formalism allows for either a transfer-based or interlingual strategy. We prefer the former since it offers more independent treatment of specific language pairs without a commitment to a single "universal" representation.

Using Arnold *(ibid)* as a starting point, our goal is to make the ideas explicit and precise by making a faithful and efficient implementation, changing the formalism itself where appropriate. The major differences reflected in CAT2 over Arnold are: (1) consistent treatment of the lexicon as structural rules, (2) the admission of logic variables in all rule types, and (3) the elimination of constructor names as formal properties of rules. These minor changes in principle are accompanied by a major improvement in the space and time performance of the software.

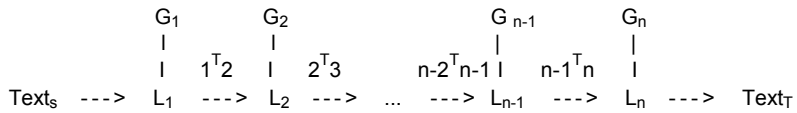
Section 1 presents a brief description of the CAT2 formalism; Section 2 describes the implementation and compares it to some existing formalisms. Section 3 illustrates the operation of CAT2 in analysis, transfer, and generation. Section 4 concludes with some directions for further research.

1. THE FORMALISM

The CAT2 formalism is extremely simple, and correspondingly restrictive. As such it is highly unlikely to be adequate for full-scale translation. Nevertheless we can already achieve a great deal within the confines of the formalism, and its simplicity allows for a controlled incremental refinement.

We retain the basic methodology as elaborated in Arnold (1987), i.e. the development of a generating device G for each representational level L, and a translation relation T between Ls:

(1)



The input is a source language text, and the output the target language text. Thus one of the Ts is the transfer component, although transfer plays no privileged role in this scheme; it is just another (very large) T relation. (Alternatively, some *Gi* could generate a pivot structure; this possibility is not pursued here further.)

At present, we treat each G as a sentence-level generator, so the input text is a string of "tokens" comprising a sentence; similarly for the output text. The initial and final Ts are special cases, the former being realized by a parser and the latter by a tree-to-string transducer.

It is tempting to consider using the CAT2 formalism in the treatment of both word morphology and text structure. Although theoretically possible, no serious experimentation has yet been undertaken in this direction.

1.1 Grammars

Each level L is represented in CAT2 as a standard derivation tree, generated by a grammar G of augmented context-free rewrite rules, where each node in the tree is a feature list of attribute/value pairs. Every terminal ("atomic") and nonterminal ("constructor") production in the grammar is expressed by rules in the following form:

(2) ROOT.LEAVES

where ROOT is the root node of a local tree and LEAVES is a list of immediate daughters under ROOT; each daughter is a root node corresponding to either an atomic or constructor rule. For terminals, LEAVES is the empty list.

Feature lists are attribute/value pairs of the form:

(3) {a1=v1,a2=v2,...,an=vn}

where each a*i* is an atomic term and each *vi* is either atomic or itself a feature list. They express such information as person, number, tense, frame, etc.

With each grammar is associated a distinguished feature. For the syntactic level, this is the syntactic category; for the functional level, the grammatical function; for the semantic level, the semantic role (e.g. agent, attribuant). The notation of nodes in the grammar is:

(4) (DF,FL)

where DF is the value of the distinguished feature and FL is the feature list of augmented features. For convenience, the following notational conventions are adopted: (1) a node whose feature list is unspecified may be indicated by the distinguished feature alone; (2) a node whose distinguished feature is unspecified is indicated by '?' in the position of the distinguished feature.

Given this basic notation, we can write atomic rules as in (5) and constructor rules as in (6):

(5) a. (verb,{lu=gone, stem=go,infl=en}).[ ]
b. (process,{lu=advise,participants=
{role1=sender,role2=receiver,role3=message}». [ ]

(6) a. (s,{mode=decl}).[(np,{per=P,num=N}), (vp,{per=P,num=N})]
b. (scomp,{cat=}).[(pred,{cat=verb,frame={arg1=subj,arg2=pobj,pform=PF}}),
(subj, {cat=np, case=nom},
(pobj,{cat=pp,lu=PF})]

The syntactic atom in (5a) represents the verb entry for "gone", the past participle of "go". The semantic rule in (5b) is an entry for the process "advise" which takes the semantic roles sender, receiver, and message (in systemic grammar terminology; cf. Halliday (1985), Steiner *et al.* (forthcoming)). The constructor (6a) is a typical syntactic rule for a declarative sentence, while (6b) shows a functional constructor for the sentence. Both illustrate the use of logical variables within feature lists.

The two fundamental constructs within CAT2 being structures and feature lists, the formalism supports two corresponding rule types: b-rules ("building rules"), as in (5) and (6), and a-rules ("attribute rules") as in (7) below. The former are used to express structural relations, the latter for feature relations, such as default feature values, co-occurrence restrictions, or completeness and coherence tests. To this end, a-rules are classified as "gentle" (need not be applicable), "strict" (must be applicable), or "filter" (must not be applicable). An example of a gentle a-rule is the following which assigns (by unification) the default feature {wh=no} to NPs which are not already marked for {wh=yes}:

(7) (np,{wh=no}).[* (?)]

The set of notational devices in CAT2 is extended to include expressions of optionality (^), Kleene star (*), disjunction (;), and sequencing (,) of nodes. For example, the specification in (7) indicates zero or more arbitrary daughter nodes. Within feature lists, we allow for the expression of inequality and strict equality of feature values (but only within rule descriptions, not in representations).

1.2 Translators

Objects (=trees) at one level of representation L_i are related to objects at an adjacent level L_{i+1} by a translation relation (=t-rule) ${}_iT_{i+1}$, denoted as:

(8) $$L_i \underset{{}_iT_{i+1}}{\Longrightarrow} L_{i+1}$$

For lexical transfer, e.g. English to German, the relation is expressed simply as an atom-to-atom t-rule:

(9) (process,{lu=go}).[ ] => (process,{lu=gehen}). [ ]

By the principle of compositionality (i.e. that "translations of expressions are normally related to the translations of their parts" (Arnold 1987)), the translation relation recursively applies to subobjects within the object described by L. By strict compositionality, we obtain:

(10) $$ROOT_s.[\ O_{s1}, O_{s2}, ..., O_{sn}\ ] \underset{{}_sT_t}{\Longrightarrow} ROOT_t.[\ O_{t1}, O_{t2}, .., O_{tn}\ ]$$

where for all *i*, 1<=*i*<=n

(11) $$O_{si} \underset{{}_sT_t}{\Longrightarrow} O_{ti}$$

Strict compositionality cannot always be maintained, by the nature of translational equivalency, so the formalism allows for selectively marking subobjects for translation and possible reordering. Thus we can express a relation such as:

(12) $$ROOT_s.[\ \$1{:}O_{s1},\ O_{s2},\ \$2{:}O_{s3}\ ] \underset{s\ t}{\overset{T}{\Longrightarrow}} ROOT_t.[\ \$2,\ \$1\ ]$$

in which the translations of O_{s1} and O_{s3} are reordered in the target object, and O_{s2} disappears from the representation.

Translation relations do not always obtain between root-daughter descriptions of depth one. For example, syntactic representations tend to have deep trees whereas functional representations are somewhat flattened. So either the lefthand or righthand side of the relation may describe arbitrarily deep structures, while still maintaining (partial) compositionality.

The following t-rule is a typical example of such an operation:

```
(13)    s.[ $1:np, vp.[ $2:v, $3:(^np) ], $4:(^pp) ]
        =>
        scomp. [ $2, $1, $3, $4 ]
```

in which the VP node is deleted and the translations of its daughters become sisters to the translations of the higher nodes.

Notice that because the translation relation is a (many-to-many) mapping of objects to objects, we can in theory treat the rule as bi-directional; we simply change '=>' to '<=>'. This has not been implemented but is a planned extension. Note further that phrasal structures may be related to phrasal and/or atomic structures, and vice versa.

2. THE IMPLEMENTATION

Because of the declarative semantics of the formalism, we use Prolog as our implementation language. This allows us to exploit unification as the principal operation, and to use list structures for the representation of trees and feature lists. It also entails a minimal difference between the formal specification and its implementation.

Important in any such research is an adequate testing environment. For system developers, Prolog is of course an excellent medium for rapid prototyping, in that it allows us to easily make provisional changes to the formalism and test their effects. For the linguists, we need facilities that provide for a convenient tool, since grammars and translators are constantly undergoing modification. Response times must also be within reasonable limits, although this is of secondary importance; more efficient software and hardware can be used once the formalism is shown to reach a certain adequacy in handling the MT problem. The Prolog program can then be viewed as the logical specification of an MT machine.

2.1 System Description

At the system level, CAT2 consists of an Earley parser, a translator, an input/output module, a rule compiler, and a command interpreter:

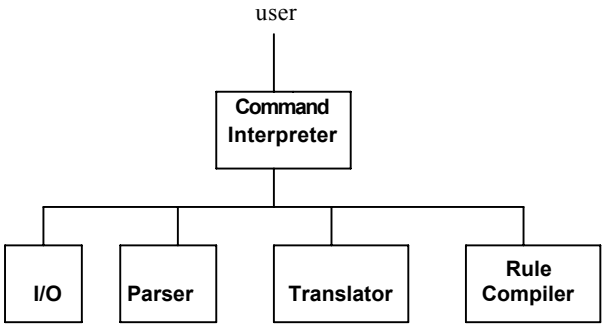


Fig. 1 The CAT2 System

All user interactions with the system are through the command interpreter, for which a command language is provided. The commands allow the user to load and compile rules, input source sentences either from files or directly at the terminal, and initiate the translation process. Online help information is provided for all commands, and a variety of command switches may be set, such as turning on and off the debugger and execution timer and setting the start symbol. Operating system commands may be executed without leaving the command interpreter, including calling the system editor.

The main utility of the command interpreter is in the maintenance of linguistic objects. Objects are created when sentences are read in, and are simply the words segmented into tokens. These string objects are parsed, creating tree objects, which then become input for subsequent translations. The user controls which objects are to be translated, and has commands to delete, save, reload, compare, and display objects, the latter two in a variety of formats, the most usual being graphic trees.

At the application level, the user writes rule compiler directives within the grammar and translator files that define the parameters for each level of representation (level name, type, and distinguished feature), demarcate the t-, b-, and a-rules, and chain to other files, thereby simplifying the loading procedure when many external files are involved.

The rules, whose syntax has been illustrated in Section 1, are converted to unit clauses in which certain features (e.g. syntactic category and lexical value) become clausal parameters, optimizing direct access. Particular attention has been given to providing informative error messages during the compilation phase.

The parser uses an Earley algorithm in which the completer step is generalized to include scanning (Kilbury 1985); i.e. scanning is completion of terminal rules. During initialization, all atomic rules in the grammar file corresponding to the input tokens are loaded into the parse chart. Tokens for which no atomic values are found are immediately reported, saving the user from having to wait for a parse only to find that, for example, the last word in the string is not in the lexicon.

The translator operates by unifying an input object with the lefthand side of available t-rules. On matching, any marked subobjects are recursively translated. The translations of these subobjects are then reconstructed according to the ordering specified on the righthand side of the t-rule. The target generator validates this resulting structure, to which any corresponding a-rules are applied. Target objects surviving this ordeal are saved as well-formed subobjects; those not surviving are saved as ill-formed subobjects. Because Prolog backtracking is applied at all stages in order to generate all possible solutions, these saved well- and ill-formed subobjects can be repeatedly accessed, improving the overall execution substantially.

2.2 Comparisons

As stated previously, the principal operation is unification, both in the sense of Prolog term unification and in graph unification of features within feature lists. A-rules, for example, whose structural configuration match that of the (sub)object become applicable for feature list unification. Success or failure of unification determines the outcome of the object depending on the classification of a-rule, whether gentle, strict, or filter. This can render a grammar nonmonotonic, in that the results are sensitive to a-rule ordering. The grammar writer must be aware of this situation, but can also take advantage of it, for example, in specifying default values (cf. Shieber (1986) for a general discussion of (non)monotonicity in unification-based grammars).

The role of unification in CAT2 resembles somewhat that of other unification-based formalisms such as LFG and FUG (Kay 1984). A major difference. however, is that in the latter, feature sets make up the sole data structure, whereas in CAT2 derivation trees are the primary data structure, feature sets being a component of the nodes. (Here there is a strong resemblance to GPSG in which the nodes in the phrase structure represent complex categories.) These two data types cannot mix; i.e. one cannot assign a tree as value to some attribute, in distinction to the unification-based approaches in which a "tree" is just another feature set. This kind of correspondence is generally made via t-rules in CAT2, as will be shown in Section 3.

As a tool for linguistic development, CAT2 shares many similarities to the PATR-II family of formalisms (e.g. Karttunen's D-PATR (1986)), keeping in mind the previous comments on unification-based formalisms, of which PATR-II is also an example. Both include many of the same command level facilities, although PATR-II offers an extensive windowing environment. Both have debugging facilities to trace the parsing (and translation in CAT2), and both have appropriate means of displaying results.

However, they differ in some fundamental respects. For one, PATR-II is based on function application, making LISP a natural implementation language (but see Hirsh (1987) on P-PATR, a Prolog-based implementation). More importantly, the translation relations form an integral part of CAT2, as does the notion of levels of representation. In PATR-II, all linguistic information (syntactic, semantic, etc.) is integrated in a single grammar, whereas in CAT2 these may be kept separate. This promotes modularity by allowing autonomous descriptions of constituent structure. functional structure, semantic structure, etc., integrated via t-rules.

Currently, each translation is performed sequentially. That is, an input text is translated to $L_{1'}$ producing a number of objects. These are then translated to L_2, and so on. This has the advantage of being able to develop each G independently

but has the disadvantage of overgeneration, since semantic information belonging to a G. is not available to disambiguate the syntactic Gi. A line of research we intend to pursue is similar to that taken by the Carnegie-Mellon University machine translation project (Carbonell & Tomita 1987) in which the syntactic and semantic grammars remain modularized knowledge sources but are precompiled into a single large grammar. This grammar can then be further compiled to produce efficient run-time code. This becomes somewhat more complex in our case since integration is via the translation relations.

3. OPERATION OF CAT2

One of the basic principles of MT is to simplify the transfer stage, if necessary at the expense of analysis and/or synthesis. This makes sense when, given nine EEC languages, we have nine complex monolingual components and 72 simple transfer components, as opposed to nine simple monolingual and 72 complex transfer components.

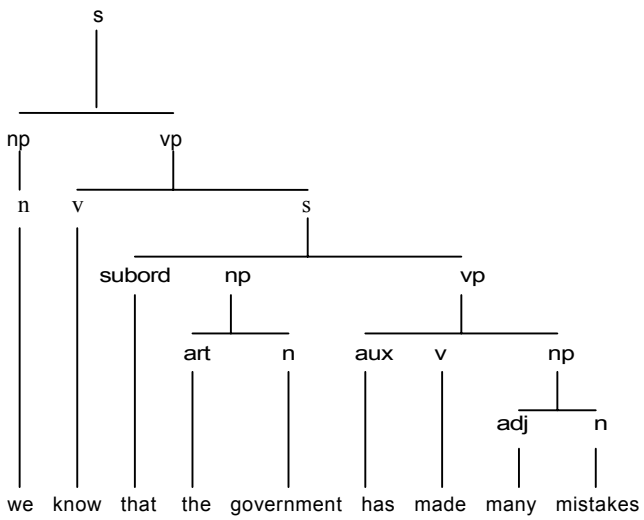
Actualising this principle entails, for example, reducing structural complexity by featurizing verbal auxiliaries, articles, particles, and grammatical formatives. This tends to normalize the monolingual structures at the interface levels, simplifying transfer.

To illustrate, take the English sentence (14) as input to the translation process:

(14) We know that the government has made many mistakes.

The syntactic structure of (14), drastically simplified for the sake of illustration, is shown in (15), much as it appears using CAT2's graphic tree displayer, where only the distinguished feature is shown on the nodes and the lexical value on the leaves. The actual object, of course, has much richer feature lists in the nodes, which can also be displayed in a "long" format.

(15)



For simplicity, we translate this structure directly to a transfer level by using the set of t-rules in (16). In the first t-rule, the subject and object (which happens to be sentential) are made sisters to the verb. This same rule applies to the sentential object by recursive t-rule application. The embedded subordinate conjunction and verbal auxiliary are not translated. In the NP t-rule, the article is converted to a feature for definiteness; a gentle a-rule (not shown) sets the default value {def=none} in case no article was present. Nouns and verbs are translated to "entity" and "process", respectively, and quantifying adjectives are made into

"modifier" structures.

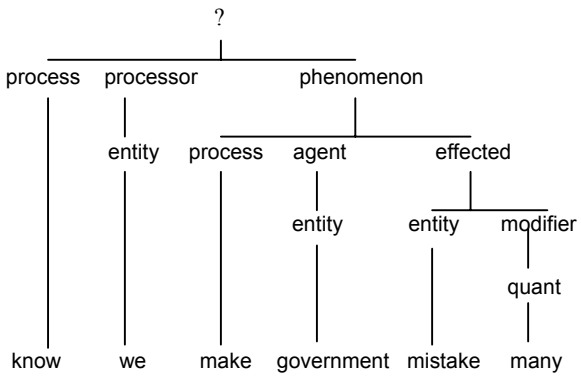
```
(16) s.[ ^subord, $1:np, vp.[ ^aux, $2:v, $3 ]]
     =>
     (?,{cat=s}).[ $2, $1, $3 ]

     np. [ ^ (art,{def=D}), $1:<^adj), $2:n ]
     =>
     (?,{cat=np,def=D}).[ $2, $1 ]

     (n,{stem=LU}).[ ] => (entity,{cat=n, lu=LU}). [ ]
     (v,<stem=LU}). [ ] => (process,{cat=v, lu=LU}). [ ]
     (adj,{type=quant,stem=LU}).[]=> modifier,[(quant,<cat=adj, lu=LU})]
```

The result is the normalized form shown in (17), assuming a very simple target grammar shown in (18):

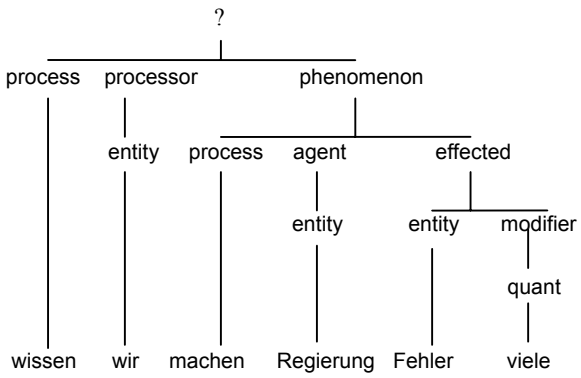
(17)




```
(18)  (?).[   (process,{participants=<role1=R1,role2=R2})),
            ^ (?,{role=R1}),
            ^ (?,{role=R2}) ]
      (.?).[   entity,   *modifier   ]
      (process,{lu=know,participants={role1=agent,role2=affected}}). [ ]
      (process,{lu=know,participants={role1=processor,role2=phenomenon}}).[ ]
      (process,{lu=make,participants={role1=agent,role2=effected}}). [ ]
      (entity,{lu=we}).[ ]
      (entity,{lu=government}).[ ]
```

In the ideal case, such as this one, transfer structures are isomorphic between levels, so that translation is performed strictly compositionally. Translation into German, for example, will yield:

(19)



This example points out a typical case of lexical ambiguity. The verb "know" translates as "kennen" (persons) and "wissen" (facts). If we take the strong but plausible position that semantic roles are maintained across levels, then we need only the two very simple atomic t-rules in (20) to correctly select the second reading of "know" and reject the first:

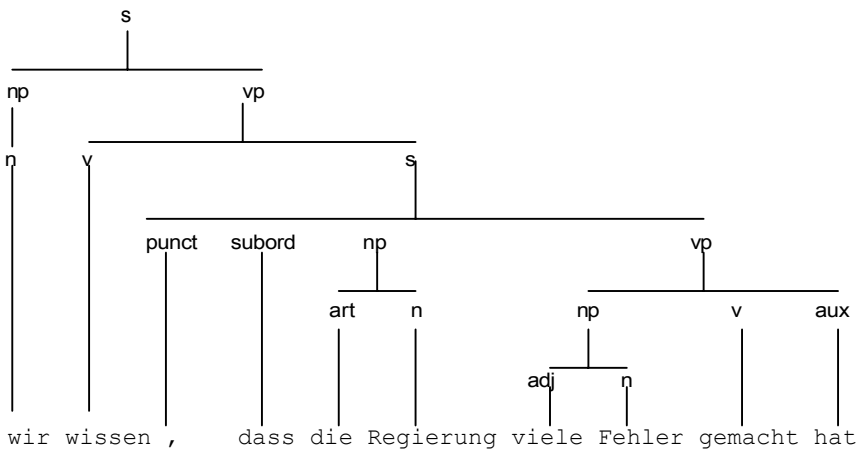
(20) (process,{lu=know,participants=P}). [ ] => (process,{lu=kennen,participants=P}) [ ]
(process,lu=know,participants=Q}). [ ] => (process,{lu=wissen,participants=Q}). [ ]

To effect generation, we basically turn the t-rules around so that articles, auxiliaries, and other necessary lexical items are restored. These appear on the righthand sides of t-rules, as in:

(21) (?,{cat=np,def=D}).[$1:entity, $2:(^modifier)]
=>
np. [^ (art,{def=D}), $2, $1]

The resulting constituent structure has the form shown in (22); a final stage would take the terminal elements in the tree and display them as in (23), capitalizing the first word in the string and inserting final punctuation:

(22)



(23) Wir wissen, dass die Regierung viele Fehler gemacht hat.

4. FURTHER RESEARCH

CAT2 as an MT system is still in its primitive stages, although the implementation is such that we can comfortably test reasonably complex linguistic phenomena in all stages of analysis, transfer, and synthesis. At present, we are concentrating on the German monolingual components, which consist of a lexicon of some 1500 words with 30-50 grammar rules at each level and about 40 t-rules. Components for French and English are underway, along with the respective transfer components.

With respect to the system software, we are currently using C-Prolog, because of its high portability to other installations. To improve performance we are investigating conversion to a compilable Prolog and interfacing to an external lexical DBMS. We are also rewriting the parsing algorithm to take advantage of logic-based parsers (Matsumoto, *et al.* 1983; Okunishi, *et al.* 1987).

Linguistically, we can express some interesting phenomena like certain long

distance dependencies, control, and raising. A degree of disambiguation has been achieved using valency information, semantic roles and semantic features (Zelinsky-Wibbelt forthcoming). We have yet to develop efficient strategies for phenomena such as coordination and ellipsis. Complex translation problems such as idioms and paraphrase have to be worked out. Furthermore, intersentential phenomena such as pronoun resolution have not yet been dealt with. All of these areas are of course on our research agenda. Nevertheless, at present we can investigate a wide range of MT issues within a theoretically attractive formalism.

ACKNOWLEDGEMENTS

The original implementation of <C,A>,T was produced primarily by Mike Rosner and Dominique Petitpierre of ISSCO (Geneva), whose efforts resulted in an amazingly stable and usable first prototype. The development of CAT2 has benefitted from discussions with them, as well as from ongoing support from my colleagues at IAI, in particular Paul Schmidt, Jorg Schütz, and Johann Haller.

REFERENCES

Arnold, D., L.Jaspaert, R.Johnson, S.Krauwer, M.Rosner, L.des Tombe, G.B.Varile & S.Warwick (1985). "A *MU1* View of the <C,A>,T Framework in EUROTRA" in *Proceedings of the Conference on Theoretical & Methodological Issues in Machine Translation of Natural Languages,* Colgate Univ., Hamilton, N.Y., pp. 1-14.

Arnold, D., S.Krauwer, M.Rosner, L.des Tombe & G.B.Varile (1986). "The <C,A>,T Framework in EUROTRA: A Theoretically Committed Notation for MT" in *Proceedings of COLING'86,* Univ. of Bonn, pp. 297-303.

Arnold, D. & L.des Tombe (1987). "Basic theory and methodology in EUROTRA" in S.Nirenburg (ed): *Machine Translation: Theoretical and Methodological Issues,* Cambridge University Press, Cambridge, England, pp. 114-135.

Carbonell, J. & M.Tomita (1987). "Knowledge-based machine translation, the CMU approach" in S.Nirenburg (ed): *Machine Translation: Theoretical and Methodological Issues,* Cambridge University Press, Cambridge, England, pp. 68-89.

Gazdar, G., E.Klein, G.Pullum & I.Sag (1985). *Generalized Phrase Structure Grammar,* Basil Blackwell, Oxford.

Halliday, M.A.K. (1985). *An Introduction to Functional Grammar,* Edward Arnold, London.

Hauenschild, C. & S.Busemann (forthcoming: 1988). "A Constructive Version of GPSG for Machine Translation" in E.Steiner, P.Schmidt & C.Zelinsky-Wibbelt (eds): *From Syntax to Semantics: Insights from Machine Translation,* Frances Pinter, London.

Hirsh, S. (1987). "P-PATR: A Compiler for Unification-based Grammars" in *Proceedings of the 2nd International Workshop on Natural Language Understanding and Logic Programming,* Simon Fraser Univ., Vancouver, Canada, pp. 63-74.

Kaplan, R. & J. Bresnan (1982). "Lexical Functional Grammar: A Formal System for Grammatical Representation" in J.Bresnan (ed): *The Mental Representation of Grammatical Relations,* MIT Press, Cambridge, Massachusetts.

Karttunen, L. (1986). D-PATR: A Development Environment for Unification-Based Grammars, CSLI Report No. CSLI-86-61, SRI, Stanford, California.

Kay, M. (1984). "Functional Unification Grammar: A formalism for machine translation" in *Proceedings of COLING'84,* Stanford Univ., California, pp. 75-78.

Kilbury, J. (1985). Chart Parsing and the Earley Algorithm, KIT-Report 24, Technische Universitat Berlin, Berlin.

Matsumoto, Y., H.Tanaka, H.Hirakawa, H.Miyoshi & H.Yasukawa (1983). "BUP: A Bottom-Up Parser Embedded in Prolog" in *New Generation Computing* 1:2, pp. 145-158.

Netter, K. & J.Wedekind (1986). "An LFG-based Approach to Machine Translation" in *Proceedings of IAI-MT86,* IAI, Saarbrücken, pp. 199-209.

Okunishi, T., R.Sugimura, Y.Matsumoto, N.Tamura, T.Kamiwaki & H.Tanaka (1987). "Comparison of Logic Programming Based Natural Language Parsing Systems" in *Proceedings of the 2nd International Workshop on Natural Language Understanding and Logic Programming,* Simon Fraser Univ., Vancouver, Canada, pp. 90-102.

Schieber, S. (1986). *An Introduction to Unification-Based Approaches to Grammar,* CSLI Lecture Notes No. 4, CSLI, Stanford, California.

Steiner, E., U.Eckert, B.Roth & J.Winter-Thielen (forthcoming: 1988). "The Development of the EUROTRA-D System of Semantic Relations" in E.Steiner, P.Schmidt & C.Zelinsky-Wibbelt (eds): *From Syntax to Semantics: Insights from Machine Translation,* Frances Pinter, London.

Zelinksy-Wibbelt, C. (forthcoming: 1988). "From Cognitive Grammar to the Generation of Semantic Interpretation in Machine Translation" in E.Steiner, P.Schmidt & C.Zelinsky-Wibbelt (eds): *From Syntax to Semantics: Insights from Machine Translation,* Frances Pinter, London.