ZERO MORPHEMES
IN UNIFICATION-BASED COMBINATORY CATEGORIAL GRAMMAR

Chinatsu Aone
The University of Texas at Austin
&
MCC
3500 West Balcones Center Dr.
Austin, TX 78759
(aone@mcc.com)

Kent Wittenburg
MCC
3500 West Balcones Center Dr.
Austin, TX 78759
(wittenburg@mcc.com)

ABSTRACT

In this paper, we report on our use of *zero morphemes* in Unification-Based Combinatory Categorial Grammar. After illustrating the benefits of this approach with several examples, we describe the algorithm for compiling zero morphemes into unary rules, which allows us to use zero morphemes more efficiently in natural language processing.[1] Then, we discuss the question of equivalence of a grammar with these unary rules to the original grammar. Lastly, we compare our approach to zero morphemes with possible alternatives.

1. Zero Morphemes in Categorial Grammar

In English and in other natural languages, it is attractive to posit the existence of morphemes that are invisible on the surface but have their own syntactic and semantic definitions. In our analyses, they are just like any other overt morphemes except for having null strings (i.e. " "), and we call them zero morphemes. Most in Categorial Grammar and related forms of unification-based grammars, on the other hand, take the rule-based approach. That is, they assume that there are unary rules that change features or categories of their arguments (cf. Dowty 1977, Hoeksema 1985, Wittenburg 1986, Wood 1987). Below, we will discuss the advantages of our zero morpheme approach over the rule-based approach.

Zero morphemes should be distinguished from so-called "gaps" in wh-questions and relative clauses in that zero morphemes are not traces or "place holders" of any other overt morphemes in a given sentence. There are at least two types of zero morphemes: zero morphemes at the morphology level and those at the syntax level.

A zero morpheme at the morphology level applies to a free morpheme and forms an inflected word. Such examples are present tense zero morpheme (PRES) as in "I like+PRES dogs" and a singular zero morpheme (SG) as in "a dog+SG". These two are the counterparts of a third person singular present tense morpheme ("+s" as in "John like+s dogs" and a plural morpheme ("+s" as in "two dog+s"), respectively.

```
(1)      dog            +SG
     N[num:null]  N[num:sg]\N[num:null]

         dog            +s
     N[num:null]  N[num:pl]\N[num:null]
```

Notice that, unlike the rule-based approach, the declarative and compositional nature of the zero morpheme approach makes the semantic analysis easier, since each zero morpheme has its semantic definition in the lexicon and therefore can contribute its semantics to the whole interpretation just as an overt morpheme does. Also, the monotonicity of our "feature adding" approach, as opposed to "default feature" approach (e.g., Gazdar 1987), is attractive in compositional semantics because it does not have to retract or override a semantic translation contributed by a word with a default feature. For example, "dog" in both "dog+SG" and "dog+s" contributes the same translation, and the suffixes "+SG" and "+s" just add the semantics of number to their respective head nouns. In addition, this approach helps reduce redundancy in the lexicon. For instance, we do not have to define for each base verb in the lexicon their present-tense counterparts.

[1] The work described here is implemented in Common Lisp and being used in the Lucy natural language understanding system at MCC.

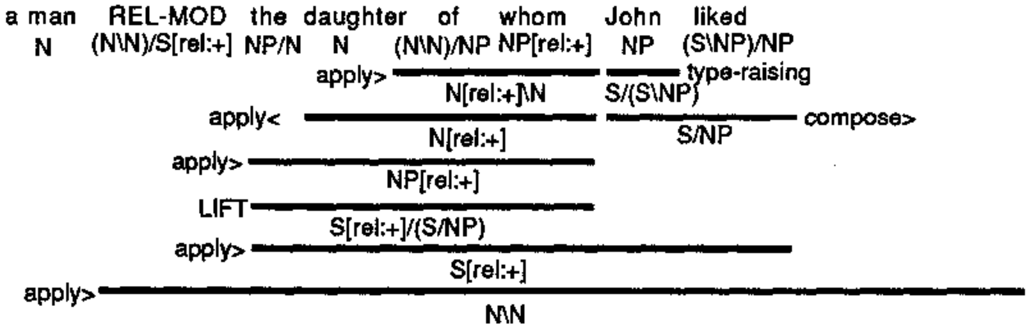
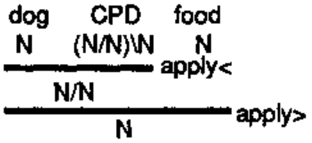


Figure 1: Derivation of "a man the daughter of whom John liked"

Some zero morphemes at the syntax level are those which may apply to a constituent larger than a single word and change the categories or features of the constituent. They are like ordinary derivational or inflectional morphemes except that their application is not confined within a word boundary. In English, one example is the noun compounding zero morpheme (CPD), which derives a noun modifier from a noun. In Categorial Grammar, its syntactic type is (N/N)\N.[2] For instance, a noun compound "dog food" might have the following derivation.

(2)



In knowledge-based or object-oriented semantics (cf. Hirst 1987), which our LUCY system uses, the treatment of compound nouns is straightforward when we employ a zero morpheme CPD.[3] In LUCY, CPD has a list of translations in the semantic lexicon, each of which is a slot relation (a two-place predicate as its syntactic type) in the knowledge base. For example, for "dog food" CPD may be translated into (food-typically-eaten x y), where x must be an instance of class Animal and y that of Food. Thus, a translation of CPD is equivalent to a value bound to the "implicit relation" called *nn* that Hobbs and Martin (1987) introduce to resolve compound nouns in TACITUS. In our case, having CPD as a lexical item, we do not have to introduce such an implicit relation at the semantics level.

An analogous zero morpheme provides a natural analysis for relative clauses, deriving a noun modifier from S. This zero morpheme, which we call REL-MOD, plays an important role in an analysis of pied-piping, which seems difficult for other approaches such as Steedman (1987, 1988). (See Pollard (1988) for his criticism of Steedman's approach.) Steedman assumes that relative pronouns are type-raised already in the lexicon and have noun-modifier type (N\N)/(S/(S|NP). In Figure 1, we show a derivation of a pied-piping relative clause "a man the daughter of whom John liked " using REL-MOD.[4] [5]

Other zero morphemes at the syntax level are used to deal with *optional* words. We define a zero morpheme for an invisible morpheme that is a counterpart of the overt one. An example is an accusative relative pronoun as in "a student (who) I met yesterday". Another example of this kind is "you" in imperative

[2] CPD is leftward-looking to parallel the definition of a hyphen as in "four-wheeler".

[3] Some compound nouns are considered as "idiomatic" single lexical entries, and they do not have a CPD morpheme. (e.g. "elephant garlic")

[4] We assume that accusative wh-words are of basic NP type in the lexicon. A unary rule LIFT, which is similar to type-raising rule, lifts any NP of basic type with [rel:+] feature to a higher type NP, characteristic of fronted phrases. This feature is passed up by way of unification.

[5] We actually use predictive versions of combinators in our runtime system (Wittenburg 1987).

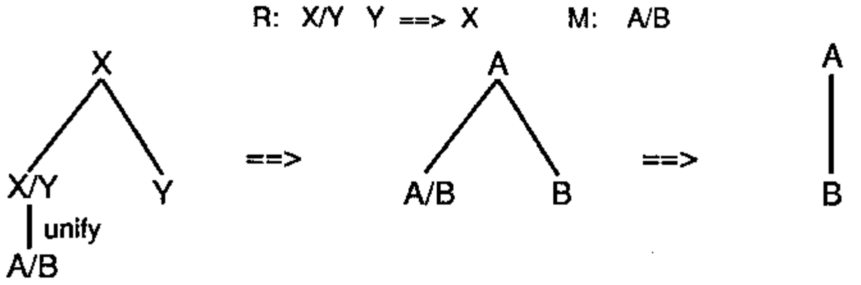


Figure 2: Compiling a zero morpheme

sentences. Having a zero morpheme for the unrealized "you" makes parsing and the interpretation of imperative sentences straightforward.[6]

(3)

```
        IMP       IMP-YOU    finish       dinner
S[mood:imp]/S    NP      (S\NP)/NP       NP
              [case:nom]
                   apply>  ____________________
                                  S\NP
           apply< _____________________________
                               S
     apply> ___________________________________
                               S
                           [mood:imp]
```

Analogous to the treatment of optional words, VP-ellipsis as in "Mary likes a dog, and Bill does too" is handled syntactically by defining a syntax-level zero morpheme for an elided verb phrase (called VP-ELLIPSIS). During the discourse process in LUCY, the antecedent of VP-ELLIPSIS is recovered.[7]

(4)

```
Bill       does         VP-ELLIPSIS
 NP     S\NP/(S\NP)       S\NP
        _________________________ apply>
                  S\NP
_________________________________ apply<
            S
```

Now to summarize the advantages for having zero morphemes, first, zero morphemes like PRES and SG reduce redundancy in the lexicon. Second, zero morphemes seem to be a natural way to express words that do not appear on the surface but have their overt counterparts (e.g., null accusative relative pronouns, vp-ellipsis). Third, since each zero morpheme has its own syntax and semantic interpretation in much the same way as overt morphemes, and since the semantic interpretations of binary rules that combine a zero morpheme with its argument (or functor) are kept as simple as they are in Categorial Grammar, semantic interpretations of sentences with zero morphemes are compositional and straightforward. Typically in the rule-based approach, the semantic operations of unary rules are more complicated: they might perform such operations as introducing or retracting some semantic primitives that do not exist in the semantic lexicon. But with our zero morpheme approach, we can avoid such complication. Lastly, using zero morpheme REL-MOD makes the analysis of pied-piping and preposition fronting of relative clauses in Categorial Grammar possible.

In the following section, we propose an approach that keeps all these advantages of zero morphemes while maintaining the efficiency of the rule approach in terms of parsing.

2. Compiling Zero Morphemes

In natural language processing, simply proposing zero morphemes at each juncture in a given input string during parsing would be a nightmare of inefficiency. However, using the fact that there are only a few binary rules in Categorial Grammar and each zero morpheme can combine with only a subset of these rules because of its directionality compatibility, we can pre-compile zero morphemes into equivalent unary rules and use the latter for parsing. Our approach is an extension of the predictive combinator compilation method discussed in Wittenburg (1987). The idea is that we first unify a zero morpheme M with the left or right daugh-

[6] Each sentence must have one of the three mood features -- declarative, interrogative, and imperative mood. They are added by zero morphemes DECL, QUES, and IMP, respectively.

[7] See Kameyama and Barnett (1989).

```
Let M be a zero morpheme, R be a binary rule.  For each M in the grammar, do the following:
For each binary rule R  in the grammar
        if the syntax graph of M unifies with the left daughter of R
            then call the unified binary graph R', and
                make the right daughter of R' the daughter of a new unary rule R1
                make the parent of R' the parent of R1
        if the syntax graph of M unifies with the right daughter of R
            then call the unified binary graph R'
                make the left daughter of R' the daughter of a new unary rule R1
                make the parent of R' the parent of R1.
```

Figure 3: Algorithm for compiling zero morphemes

ter of each binary rule R. If they unify, we create a specialized version of this binary rule R', maintaining features of M acquired through unification. Then, we derive a unary rule out of this specialized binary rule and use it in parsing. Thus, if M is of type A/B, R is forward application, and M unifies with the left daughter of R, the compiling procedure is schematized as in Figure 2.

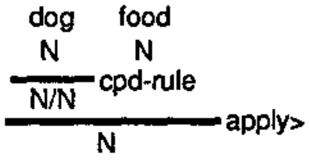
Now I shall describe the algorithm for compiling zero morphemes in Figure 3. During this compiling process, the semantic interpretation of each resulting unary rule is also calculated from the interpretation of the binary rule and that of the zero morpheme. For example, if the semantics of M is **M'**, given that the semantic interpretation of forward application is λfun-λarg(fun arg), we get λarg(**M'** arg) for the semantic interpretation of the compiled unary rule.[8]

We also have a mechanism to merge two resulting unary rules into a new one. That is, if a unary rule R1 applies to some category A, giving A', and then a unary rule R2 applies to A', giving A", we merge R1 and R2 into a new unary rule R3, which takes A as its argument and returns A". For example, after compiling IMP-rule and IMP-YOU-rule from zero morphemes IMP and IMP-YOU (cf. (3)), we could merge these two rules into one rule, IMP+IMP-YOU rule. During parsing, we use the merged rule and deactivate the original two rules.

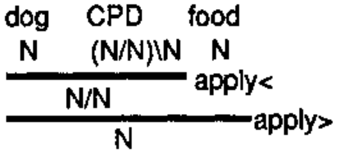
3. The Grammar with Compiled zero morphemes

The grammar with the resulting unary rules has the same generative capacity as the source grammar with zero morphemes in the lexicon because these unary rules are originally derived by only using the zero morphemes and binary rules in the source grammar. Thus, a derivation which uses a unary rule can always be mapped to a derivation in the original grammar, and vice versa. For example, look at the following example of CPD-RULE vs. zero morpheme CPD:

(5) a.



b.



Now, if we assume that we use Categorial Grammar with four binary rules, namely, apply>, apply<, compose>, and compose<, as Steedman (1987) does, we can predict, among 8 possibilities (4 rules and the 2 daughters for each rule), the maximum number of unary rules that we derive from a zero morpheme according to its syntactic type.[9] If a zero morpheme is of type A/B, it unifies with the left daughters of apply>, apply< and compose> and with the right daughters of apply> and com-

[8] See Wittenburg and Aone (1989) for the details of Lucy syntax/semantics interface.

[9] Zero morphemes do not combine with wh-word type-raising rule LIFT, which is the only unary rule in our grammar besides the compiled unary rules from zero morphemes.

pose>. Thus, there are 5 possible unary rules for this type of zero morpheme. If a zero morpheme is of type A\B, there are also 5 possibilities. That is, it unifies with the left daughter of apply< and compose<, and the right daughters of apply>, apply< and compose<. If a zero morpheme is of basic type, there are only 2 possibilities; it unifies only with the left daughter of apply< and the right daughter of apply>.

Furthermore, in our English grammar, we have been able to constrain the number of unary rules by pre-specifying for compilation which rules to unify a given zero morpheme with.[10] We add such compiler flags in the definition of each zero morpheme. We can do this for the *morphology*-level zero morphemes because they are never combined with anything other than their root morphemes by binary rules, and because we know which side of a root morpheme a given zero affix appears and what are the possible syntactic types of the root morpheme. As for zero morphemes at the *syntax* level, we can ignore composition rules when compiling zero morphemes which are in islands to "extraction", since these rules are only necessary in extraction contexts. CPD, REL-MOD and IMP-YOU are such syntax-level zero morphemes. Additional facts about English have allowed us to specify only one binary rule for each syntax-level zero morpheme in our English grammar. An example of a zero morpheme definition is shown below.

```
(6) (defzeromorpheme PRES
        :syntax  S[tns:pres]\S[tns:null]
        :compile-info (:binary-rule compose<
                       :daughter R))
```

4. Comparison in View of Parsing Zero Morphemes

In this section, we compare our approach to zero morphemes to alternative ways from the parsing point of view. Since we do not know any other comparable approach which specifically included zero morphemes in natural language processing, we compare ours to the possible approaches which are analogous to those which tried to deal with gaps. For example, in Bear and Karttunen's (1979) treatment of wh-question and relative pronoun gaps in Phrase Structure Grammar, a gap is proposed at each vertex during parsing if there is a wh-question word or a relative pronoun in the stack. We can use an analogous approach for zero morphemes, but clearly this will be extremely inefficient. It is more so because 1) there is no restriction such as that there should be only one zero morpheme within an S clause, and 2) the stack is useless because zero morphemes are independent morphemes and are not "bound" to other morphemes comparable to wh-words.

Shieber (1985) proposes a more efficient approach to gaps in the PATR-II formalism, extending Earley's algorithm by using *restriction* to do top-down filtering. While an approach to zero morphemes similar to Shieber's gap treatment is possible, we can see one advantage of ours. That is, our approach does not depend on what kind of parsing algorithm we choose. It can be top-down as well as bottom-up.

5. Conclusion

Hoeksema (1985] argues for the rule-based approach over the zero morpheme approach, pointing out that the postulation of zero morphemes requires certain arbitrary decisions about their position in the word or in the sentence. While we admit that such arbitrariness exists in some zero morphemes we have defined, we believe the advantages of positing zero morphemes, as discussed in Section 1, outweigh this objection. Our approach combines the linguistic advantages of the zero morpheme analysis with the efficiency of a rule-based approach. Our use of zero morphemes is not restricted to the traditional zero-affix domain. We use them, for example, to handle optional words and VP-ellipsis, extending the coverage of our grammar in a natural way.

ACKNOWLEDGEMENTS

We would like to thank Megumi Kameyama and Michael O'Leary for their help.

REFERENCES

Bear, John and Lauri Karttunen. 1979. PSG: A Simple Phrase Structure Parser. In R. Bley-Vroman and S. Schmerling (eds.), *Texas Linguistic Forum*. No. 15.

[10] In fact, we use more than two kinds of composition rules for the compilation of the morphology-level zero morphemes. (e.g. PRES in (1)) But this does not cause any "rule proliferation" problem for this reason.

Dowty, David. 1979. *Word Meaning and Montague Grammar.* D. Reidel Publishing Company.

Gazdar, Gerald. 1987. Linguistic Applications of Default Inheritance Mechanisms. In P. Whitelock et al. (eds.), *Linguistic Theory and Computer Applications.* Academic Press.

Hirst, Graeme. 1987. *Semantic Interpretation and the Resolution of Ambiguity.* Cambridge University Press.

Hobbs, Jerry and Paul Martin. 1987. Local Pragmatics. In *Proceedings IJCAI-87.*

Hoeksema, Jack. 1985. *Categorial Morphology.* Garland Publishing, Inc. New York & London.

Kameyama, Megumi and Jim Barnett. 1989. VP Ellipsis with Distributed Knowledge Sources. MCC Technical Report number ACT-HI-145-89.

Pollard, Carl. 1988. Categorial Grammar and Phrase Structure Grammar: An Excursion on the Syntax-Semantics Frontier. In R. Oehrle et al. (eds.), *Categorial Grammars and Natural Language Structures.* D. Reidel Publishing Company.

Shieber, Stuart. 1985. Using Restriction to Extend Parsing Algorithms for Complex-Feature-Based Formalisms. In *Proceedings of the 23rd Annual Meetings of the Association for Computational Linguistics.* 145-152.

____________ 1986. *An Introduction to Unification-Based Approaches to Grammar.* CSLI, Stanford University, California.

Steedman, Mark. 1985. Dependency and Coordination in the Grammar of Dutch and English. *Language.* 61:523-568.

____________ 1987. Combinatory Grammars and Parasitic Gaps. *Natural Language and Linguistic Theory.* 5:403-439.

____________ 1988. Combinators and Grammars. In R. Oehrle et al. (eds.), *Categorial Grammars and Natural Language Structures.* D. Reidel Publishing Company.

Wittenburg, Kent. 1986. *Natural Language Parsing with Combinatory Categorial Grammar in a Graph-Unification-Based Formalism.* Doctoral dissertation, The University of Texas, Austin.

____________ 1987. Predictive combinators: A Method for Efficient Processing of Combinatory Categorial Grammars. In *Proceedings of the 25th Annual Meetings of the Association for Computational Linguistics.*

Wittenburg, Kent and Chinatsu Aone. 1989. Aspects of a Categorial Syntax/Semantics Interface. MCC Technical Report number ACT-HI-143-89.

Wood, Mary. 1987. *Paradigmatic Rules for Categorial Grammars.* CCL/UMIST Report, Centre for Computational Linguistics, University of Manchester, Institute of Science and Technology.