American Journal of Computational Linguistics *Microfiche 33 : 33*

A Lexical Process Model of Nominal Compounding in English

JAMES R. RHYNE

Department of Computer Science
University of Houston
Houston, Texas 77004

ABSTRACT

A theoretical model for nominal compound formation in English is presented in which the rules are representations of lexical processes. It is argued that such rules can be generalized to account for many nominal compounds with similar structure and to enable new compounds to be produced and understood. It is shown that nominal compounding depends crucially on the existence of a "characteristic" relationship between a nominal and the verb which occurs in a relative clause paraphrase of a compound which contains the nominal. A computer implementation of the model is presented and the problems of binding and rule selection are discussed.

Linguistic Issues.

Nominal compounds are sequences of two or more nominals which have the semantic effect of noun phrases with attached relative clauses. The rightmost nominal is generally the primary referent of the compound the other nominals restrict the reference of the rightmost nominal in much the same fashion that a relative clause does. There are, of course, exceptions in which the rightmost nominal is figurative or euphemistic (e.g. family jewels). Compounds occur frequently in English and Germanic languages, but infrequently in the Romance languages where their function is largely performed by nominal-preposition-nominal sequences (e.g. chemin de fer, agent de change).

The syntactic structure of nominal compounds is quite simple --the three variants are N-N, N-participle-N, and N-gerund-N. In the N-N form, either of the two nominals may in fact be yet another nominal compound, giving a structure like (N-N)-N or N-(N-N); the first of these forms seems to occur much more often than the second (examples of each type are: typewriter mechanic, liquid roach poison).

I assume that the process of nominal compounding is syntactically a process in which a relative clause is reduced by deleting all elements of the relative clause but one and preposing the single remaining element in front of the antecedent nominal. In addition, the clause verb may be nominalized and preposed. Other linguists have proposed different derivations for nominal compounds; Lees [3], for example, derives nominal compounds from nominal-preposition-nominal sequences. There are two reasons why I feel that Lees approach is wrong: (1) there are English compounds for which no reasonable equivalent nominal-preposition-nominal paraphrase can be given (e.g. windmill), and (2) there are subtle meaning differences between the nominal compounds and their nominal-preposition-nominal counterparts (county clerk vs. clerk for the county). If nominal compounds and nominal-preposition-nominal sequences are derived from forms like relative clauses, then the differences in meaning can be accounted for by deriving each form from a distinct relative clause; the relative clauses may, of course, be quite closely related to each other.

I have spoken rather loosely about deriving nominal compounds from relative clauses; I am not proposing a derivation system which operates on surface forms of the language, and what I intend that the reader should understand is that an underlying form for a nominal compound is derived from an underlying form for a relative clause by a language process which I term a lexical rule because, as we shall see, the operation of such rules depends crucially on the specific lexical items which are present in the underlying structures. Linguists have identified a number of lexical processes in English; some examples of such processes may be found in [1] and [2].

The underlying forms associated with relative clauses and nominal compounds in the model of nominal compounding being presented here are networks (trees for the most part) defined in terms of a case grammar which is closely related to that used by Simmons [5]. The cases which appear in this system fall into two general categories: (1) cases of the clause verb, which are the following -- Performer, Object, Goal, Source, Location, Means, Cause, and Enabler -- and (2) structural cases, which are RELCL (relative clause) and COMP (compound). I will not explain these cases in detail, as that is the subject of a forthcoming paper. But the following observations will illuminate the case system for verb cases. The case system distinguishes the immediate performer of an act from a remote cause or agent of the act. The reason for this distinction lies in an intimate connection between verbs and the assumed or habitual performer of the act which is the reference of the verb. The case system also distinguishes an active causative agent of an act from an agent which merely permits the act to occur; this distinction in the case system permits two classes of verbs to be distinguished according to whether the surface subject commonly causes the act or permits the act to occur.

The case system used in the present model of nominal compounding is not a deep case system; on the contrary, it seems that nominal compounding is a lexical process which occurs rather near the surface in a derivational grammar model. An example which can be given to support this is the compound _ignition key_; this is a key which turns a switch which enables a complex sequence of events to take place that ultimately result in the ignition of a fuel/air mixture in an engine, or one may describe it equivalently as a key which causes ignition. The first description corresponds to a deep case level of description while the second corresponds to the level at which the compound ignition key is formed. I would argue that if one takes the deep case approach, then one is forced to include a great deal of structure in the rules for nominal compounding; in particular, the rule for _ignition key_ must remove all of the links in the causal chain leading to the ignition act. The deletion of this intermediate information must be done to obtain the description given in the second case, and to include the deletion procedure in both a compounding rule and in the rule process which leads to the shorter description means unnecessarily duplicating the procedure. Moreover, if one derives compounds from paradigm relative clauses of the second sort, e.g. key which causes an action to occur, then it is possible to generalize compound forming rules so that a single rule may produce several compounds. It will not be possible to do this if deep cases are used as the deep case structure of _firing key_ will be quite different from that of _ignition key_.

In order to understand the model of compounding which is being presented here, it is essential to consider the function of compounding in language. In my view, compounding is a process which allows a speaker to systematically delete information from an utterance just when the speaker has reason to expect that the hearer can reconstruct that information. In effect, I consider compounding (and a great many other linguistic processes) to be examples of linguistic encoding which are used to speed up communication, and the grammar shared by the speaker and hearer must include the encoding and decoding functions.

Consider the nominal compound _steam distillation_, which refers to the distillation of some substance with steam; the hearer of the compound _steam distillation_ knows that _distillation_ is the derived nominal form of _distill_. The hearer also knows what the common or characteristic cases of the verb _distill_ are: the agent is invariably a person or machine (this would be the occupant of the Cause case slot in my system), the instrument (or Means) may be an apparatus or a heated medium such as steam and the Goal is a liquid which is missing some of the constituents that it entered the distillation process with.

It happens that in English, whenever a derived nominal of an act is the right element in a compound, then the left element is almost always an occupant of one of the case slots of the verb. In order to recreate the underlying relative clause structure, it is only necessary for the hearer to properly choose the case for the nominal _steam_. A great deal of lexical information can be brought to bear on this question; for example, _steam_ is not a liquid, it is water vapor and thus it cannot be the starting substance or the end product of a distillation process. _Steam_ might be the Cause of the act of distillation except that there do not seem to be any compounds in English which have distillation as the right element and a Cause as the left element. Thus the hearer can assign _steam_ to the Means case with some assurance.

In another example, _shrimp boat_, the hearer can ascertain by lexical relations involving the word _boat_, that boats are characteristically used to catch marine life. One choice for the main verb in a synonymous relative clause is _catch_, which will have _boat_ as an element of the Means case. The Cause for _catch_ is commonly a person or perhaps a sophisticated machine designed to catch things (i.e. a _trap_). The Object is characteristically an animal. There is a strong characteristic relation between the animal being caught and the means used to catch it, for example mink is trapped, calves are roped, birds are netted, and fish are caught with a boat. This relation exists as a rule in the lexicon of both the speaker and the hearer and it enables the speaker to produce the nominal compound and the hearer to understand it.

Furthermore, shrimp boat is one member of a class of closely related nominal compounds which includes lobster boat, whale boat, tuna boat and many others. It would be most interesting if a single rule could be formulated which would generate all of these compounds. A lobster boat is a boat which is used to catch lobster, a tuna boat is a boat which is used to catch tuna, and so forth. All of these examples are identical except for the particular marine animal being caught. The logical next step is the creation of a rule which generalizes the individual marine animals to the common category of marine animal. This rule will state that a marine animal boat is a boat which is used to catch marine animals.

In making this generalization, I have given the rule the power to help interpret novel compounds and to generate them. With this power comes a difficulty, which is constraining the rule so that it does not generate bad compounds or produce incorrect interpretations. The key to this constraint lies in what I will term the characteristic or habitual aspect of nominal compounds. In the case of the boat compounds, a boat will only be a shrimp boat if it is characteristically, usually, habitually or invariably used to catch shrimp. So the operation of a compounding rule is enabled only if a characteristic aspect is associated with the verb; in English, this is usually indicated by an adverb or an adverbial phrase. If the speaker is willing to assert that a boat is characteristically used to catch turtles, then the nominal compound turtle boat may be used. The hearer will use the general rule to place turtle and boat in the proper case slots, and because a compound was used by the speaker, the hearer will infer that the boat is one which is characteristically used to catch turtles.

There are other problems which arise with the generalization of rules; for example, compounding never produces a compound in which the left element is a proper noun, unless the proper noun is the name of a process (e.g. Markov process) or is a Source, Performer, or Goal of an act of giving. It also seems to be true that compounds are not generally formed when a lexical item is several levels below the genéral term which appears in the rule (e.g. repairmidget) or when a cross-classificatory term is used (e.g. automobile Indian as an Indian who repairs automobiles). With all of the preceding discussion in mind, I would now like to turn to the model of nominal compounding which I have presently implemented and running.

The Computer Model

The computer model of compounding accepts relative clause structures as input and produces nominal compound structures as output when the input is appropriate. It is written in a language with many parentheses the language was chosen for its program development facilities, i.e. built-in editor, rather than for its interpretive capabilities. The program which produces nominal compounds is a pattern matching interpreter; it applies a rule of compound formation by matching one side of the rule with the input structure, and if certain criteria are satisfied by the match, items from the input structure are bound into the rule, transferred to the other side of the rule, and a copy is then made of the other side of the rule. The result is a nominal compound structure.

The model has two components: a rule interpreter and a lexicon of rules for compounding. There is nothing tricky about rule application. Consider the nominal compound flower market and its associated relative clause paraphrase market where flowers are characteristically sold. These phrases have in my system the underlying structures shown in Figure 1. The notation in square braces means that the verb sell has the characteristic aspect in this instance.

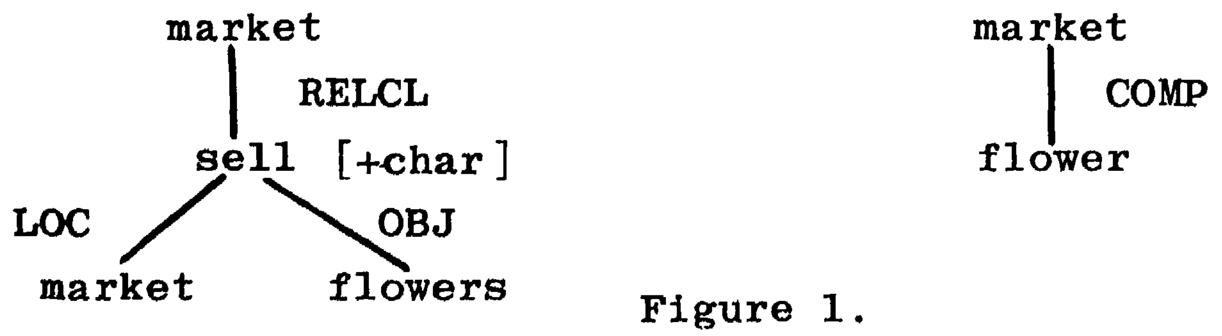

Figure 1.

These two structures can be made into a rule by linking them together. Whenever a relative clause structure identical to that in Figure 1 is received, the rule applies and a copy is created of the nominal compound flower market. The matching procedure is a relatively straightforward, top down, recursive process which has backtracking capability in the event that a structure or case occurs more than once at any given level of the structure. There are two problems which arise; however: if the rule is generalized to account for compounds other than flower market, then the lexical items in the rule will behave as variables and some provisions must be made for binding of values to these variables; also, the rule interpreter must have some heuristics for selecting appropriate rules if the time required to produce a compound is not to increase exponentially with the size of the lexicon.

The present version of the model only partly solves the binding problem. Consider the rule given in Figure 2 which is a generalization of that given in Figure 1.

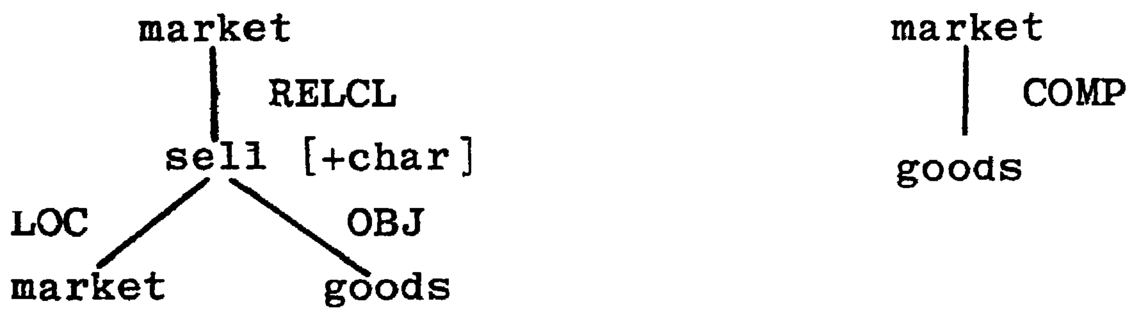

Figure 2.

If this rule is to apply to the relative clause structure given in Figure 1 and generate the compound flower market, then the rule interpreter must recognize that the relative clause in Figure 1 is an instance of that given in Figure 2. The matching procedure does this by determining that the reference set of the nominal flowers is a subset of the reference set of the nominal goods.

In addition, the nominal flowers must be carried across to the other side of the rule and substituted there for goods before the other side of the rule is copied. Thus market and goods must be bound across the rule so that whatever lexical item matches either of these nominals becomes the value associated with these nominals on the other side of the rule.

In the initial version of the model, this binding was established explicitly when the rule was entered into the lexicon, but this seemed unsatisfactorily ad hoc. In a subsequent version, the identity of the lexical items on both sides of the rule was the relation used to establish binding relationships. Consider, however, the structure shown in Figure 3.

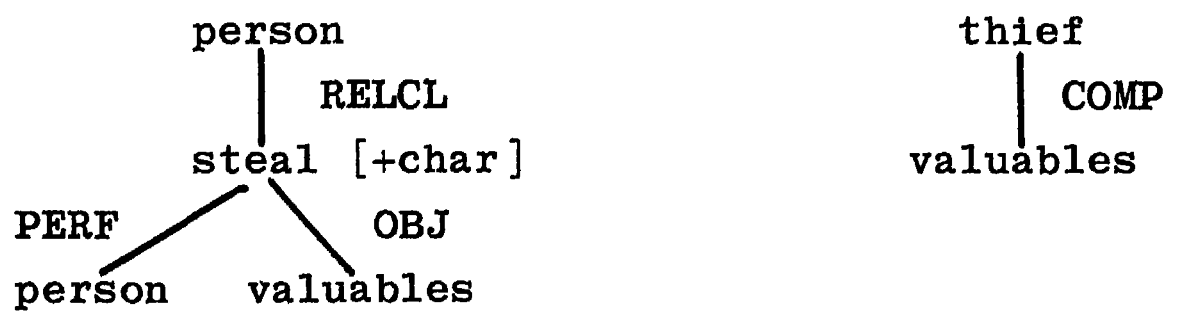

Figure 3

Here person should be bound to thief but the previous technique is not able to establish this binding. The reason that we know that person and thief should be bound is because we know that a thief is a person who steals characteristically. In the most recent version of the model, this information is used to find the binding relationships when the rule of identity does not work. The lexicon is searched for a rule which can be used to establish this binding. The rule which is used in the example shown in Figure 3 is displayed below in Figure 4.

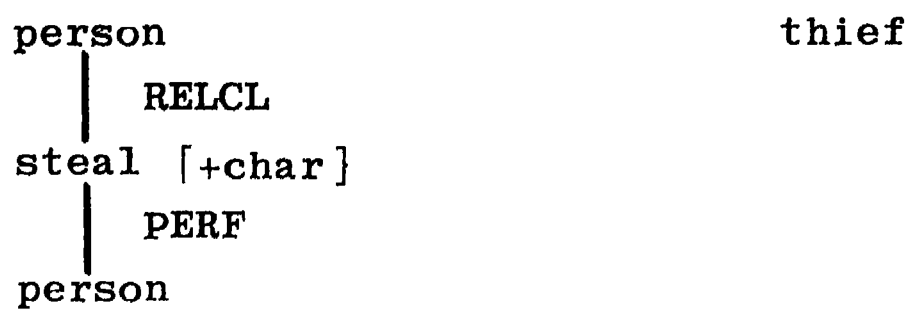

Figure 4

From the structures given in Figure 4, one can see that person should be bound to thief because the rule states that the reference set of thief is the same as the reference set of person as restricted by the relative clause.

The technique of using lexical rules to establish bindings works in virtually every instance, but it has the defect of requiring that the information that a thief is a person who steals things be represented in the lexicon twice at least. A new model is under construction which attempts to reduce this redundancy by allowing the rules to have multiple left and right parts.

The problem of selecting appropriate rules is rather easier to solve. In most compounds in English, there is a characteristic association between the right element of the nominal compound and the main verb of the associated relative clause paraphrase. These two elements which occur on opposite sides of the compounding rule supply a great deal of information about the possibilities for application of the rule. So, in the model, each rule in the lexicon is indexed by the main verb of the relative clause and by the right element of the nominal compound. This index actually contains some environmental information as well; for the clause verb, this environmental information is the case frame of the verb and the fact that it is the main verb of the relative clause -- for the compound nominal, the environmental information is just the fact that the nominal is the rightmost one in a nominal compound.

The basic model has been tested with a set of several hundred nominal compounds and is very successful in coping with a wide variety of compound types. The productivity of the rules varies greatly; some rules may produce hundreds of compounds while other rules may only result in one or two compounds. Frozen forms such as keel boat are handled by a rule which generates only one compound; there is a rule for each frozen form. The rule structures contain exclusion lists associated with each lexical item in the rule, and these exclusion lists prevent the rule from operating whenever a lexical item matches one of the items on an exclusion list if the items occur at corresponding locations in the structures.

The model is quite quick in operation; on a high speed display console, it will generally produce compounds much faster than a person sitting at the console can conveniently read them. This is mainly due to the rule selection heuristic, but the match procedure has been carefully optimized as well.

Conclusions

The model program is an excellent demonstration of the appropriateness of the basic theory; moreover, the rules themselves can be generalized to deal with syntactic processes, so there is no discontinuity in the grammar model between the lexical processes and the syntactic processes. It seems clear that the rules could also be used to represent other lexical processes in language and this is currently being pursued.

There is no reason why the rules could not be used for recognition as well as for the production of nominal compounds. The bindings are not one-way, and the matching procedure will work equally well for compound structures. The reasons why the computer model is a production model are: (1) that the computer model assumes the semantic correctness of the input relative clause structures, and (2) that compounds are often ambiguous and may be paraphrased by two or more relative clauses, while the converse of this is almost never true. A recognition model would have to generate underlying relative clause structures for each ambiguity and a semantic component would have to screen the relative clauses for semantic errors.

I hope that the reader has noticed the avoidance of rule procedures in this model. When I began working on the design of the computer programs, I had in mind the creation of a model which once implemented in LISP could be extended merely by adding new rules without having to construct any additional LISP programs. I ultimately wanted to have a model which could "learn" new rules by systematic generalization and restriction of existing rules. I feel that this would be relatively easy with rule structures and extremely difficult with rule procedures written in a programming language. Furthermore, I subscribe to Karl Popper's ideas of scientific endeavour, and rule structures appealed because it would be more difficult to bury flaws or ill understood aspects of compounding and rule processes in structures than in procedures where the computational power of the programming language permits and even encourages _ad hoc_ solutions to be found to problems.

Acknowledgements

I would like to here acknowledge the suggestions made by Robert F. Simmons, Carlota Smith, Mary Ross T. Rhyne, Laurent Siklossy, and Stanley Peters which have helped improve my understanding of nominal compounding.

1. Chomsky, N. "Remarks on Nominalization," in *Readings in English Transformational Grammar*, Jacobs, R. and Rosenbaum, P. eds. Ginn, Waltham, Massachusetts, 1970.
2. Gruber, J. "Studies in Lexical Relations." Ph. D. thesis, MIT, 1965.
3. Lees, R. *The Grammar of English Nominalizations*. Mouton, The Hague, 1968.
4. Rhyne, J. "Lexical Rules and Structures in a Computer Model of Nominal Compounding in English." Ph. D. thesis, The University of Texas at Austin, 1975.
5. Simmons, R. "Semantic Networks: Their Computation and Use for Understanding English Sentences," in *Computer Models of Thought and Language*, Schank, R. and Colby, K. eds. W. H. Freeman, San Francisco, 1973.