Relational-Grammar-Based Generation in the JETS Japanese-English Machine Translation System

David E. Johnson
IBM Research, T. J. Watson Research Laboratory
P.O. Box 218
Yorktown Heights, NY 10598 USA

Hideo Watanabe
IBM Research, Tokyo Research Laboratory
5-19 Sanbancho, Chiyoda-ku
Tokyo 102, Japan

Abstract

This paper describes the design and functioning of the English generation phase in JETS, a limited transfer, Japanese-English machine translation system that is loosely based on the linguistic framework of relational grammar. To facilitate the development of relational-grammar-based generators, we have built an NL-and-application-independent generator shell and relational grammar rule-writing language. The implemented generator, GENIE, maps abstract canonical structures, representing the basic predicate-argument structures of sentences, into well-formed English sentences via a two-stage plan-and-execute design. This modularity permits the independent development of a very general, deterministic execution grammar that is driven by a set of planning rules sensitive to lexical, syntactic and stylistic constraints. Processing in GENIE is *category-driven*, i.e., grammatical rules are distributed over a part-of-speech hierarchy and, using an inheritance mechanism, are invoked only if appropriate for the category being processed.

1- Introduction

This paper discusses relational-grammar-based generation in the context of JETS, a Japanese-English machine translation (MT) system that is being developed at the IBM Research Tokyo Research Laboratory. To put our work in perspective, we first explain the motivation for basing JETS on relational grammar (RG) and then sketch the processing flow in translation. With this background, we (i) describe and illustrate certain aspects of the rule-writing language, GEAR, in which the GENIE English generator has been written; (ii) comment on key aspects of the generator shell, GENSHELL, in which GENIE has been developed; and (iii) discuss the design and functioning of the GENIE English generator.

With few exceptions such as the work being done at CMU (cf. KBMT-89 (1989), Nirenburg (1987), and Nirenburg, et. al. (1988)), in the SEMSYN project at the University of Stuttgart (Rösner (1986)), and the joint work between the ISI Penman project and the University of Saarbrücken (Bateman, et. al. (1989)), generation within the area of machine translation has received very little attention. Typically, MT systems have no independently functioning, linguistically justified generation grammar. In the case of transfer systems, much of the target language grammar is typically built into the transfer component, resulting in a non-modular, rigid and linguistically inadequate system. It is the norm in MT systems for the linguistic complexities inherent in robust generation to be simply ignored, contributing to the inadequacy of MT systems.

In contrast, we have sought to shift more of the processing burden from transfer onto generation, allowing our system to incorporate a variety of results coming from theoretical linguistics. GENIE is an application-and-language-independent generator embodying

a robust, linguistically justified RG grammar of English. Moreover, GENIE incorporates a syntax planner that applies a set of planning rules determining which rules in the execution grammar should be applied. As long recognized in work on text generators, the incorporation of a syntax planner introduces the kind of flexibility required for robust generation.

JETS is a so-called **limited transfer** system, i.e., a system in which structural transfer is kept to a minimum. The key RG notion in our work is that of **canonical (relational) structure (CS)**, an abstract level of syntactic structure representing the basic predicate-argument structure of clauses in terms of a universal set of primitive (grammatical) relations such as subject, direct object, indirect object, chomeur.[1]

Given the basic assumption that one is developing a limited transfer system, implying deep analyses of both the source and target languages which converge on structurally similar internal representations for translation equivalents in a wide range of cases, it is critical to select a linguistic framework which supports the required analyses, enabling one to conceptualize the linguistic processing in a uniform manner. As discussed in Johnson (1988b), with respect to MT, RG is a logical choice of linguistic framework since CSs provide a natural syntactic bridge between languages as diverse in structure as Japanese and English. This is so for two reasons: (1) within one language, the CSs of paraphrases are typically the same or highly similar and (2) translation equivalents often have structurally similar if not isomorphic CSs.

One of the key advantages of RG comes from its explicit representation of grammatical relations like subject and direct object, which are argued to be universal. In contrast, structure-based frameworks such as transformational-generative grammar (TG) at best only implicitly represent grammatical relations such as subject and direct object in terms of linear precedence and dominance, which are language particular. If one considers the task of transfer, for instance, it is clear that representing basic clause structure in terms of explicitly marked, order-independent relations rather than in terms of language-dependent structural relations reduces the amount of structure changing to be done in the transfer component. This is especially true for languages like Japanese and English, which differ greatly in superficial structural properties (not to mention the fact that Japanese has very free word order, which arguably makes it even less suited to structure-based frameworks).

2- Processing Flow in JETS and GENIE

As in all transfer systems, linguistic processing in JETS can be divided into three phases: analysis, which consists of lexical analysis and parsing, transfer and generation. The output of analysis is a Japanese CS, which represents the basic predicate-argument structure of the Japanese sentence.[2] Transfer produces an English CS, which is often, but not always, isomorphic to the Japanese CS. The English CS is passed to the GENIE generator, whose task is to generate a grammatically correct and stylistically appropriate English sentence given a well-formed CS.

To illustrate, consider the following Japanese sentence and two of the possible English translations:

1.	**karera**	**wa**	**Tookyoo**	**e**	**itta**	**rashii**
	they	top	Tokyo	to	went	seem

2. **They seem to have gone to Tokyo.**

3. **It seems that they went to Tokyo.**

In translating (1), analysis maps the input string into the Japanese CS shown at the left in Figure 1 on the next page. Transfer then maps the Japanese CS into the English CS shown at the right in Figure 1.

---

[1] For theoretical background on RG, see the many articles listed in the bibliographic reference Dubinsky and Rosen (1987). Note that the following abbreviations are used in glosses of Japanese examples: **top** (topic), **nm** (nominalize), and **pp** (postposition).

[2] For discussion of parsing in JETS, see Maruyama, Watanabe and Ogino (1989).

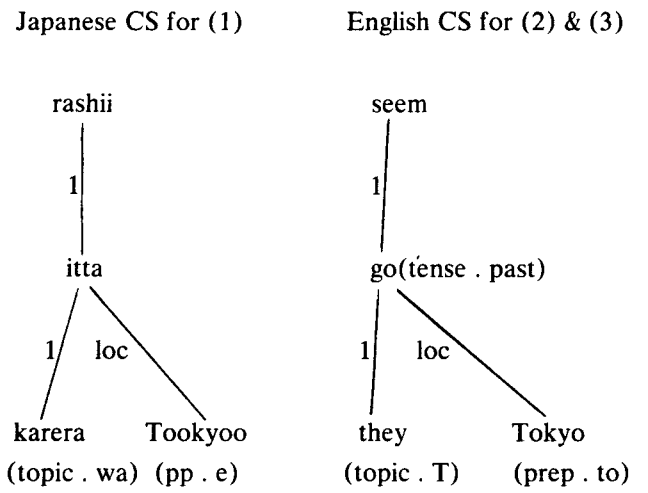


Figure 1. Canonical structures for (1), (2) and (3). Note that "1" means "subject", "loc" means "locative".

Given the English CS, it is up to the GENIE English generator to generate either (2) or (3). Based on the information that **they** in the English CS is marked as the topic of the sentence, GENIE will map the CS into the superficial (unordered) relational structure shown in Figure 2 via the relational rule of Subject-to-Subject Raising (so-called **A-raising**). Subsequent rules of Tense-Spelling and Linearization (including the spelling out of verbal forms and prepositions) will result in the string **They seem to have gone to Tokyo**, as shown in Figure 3.

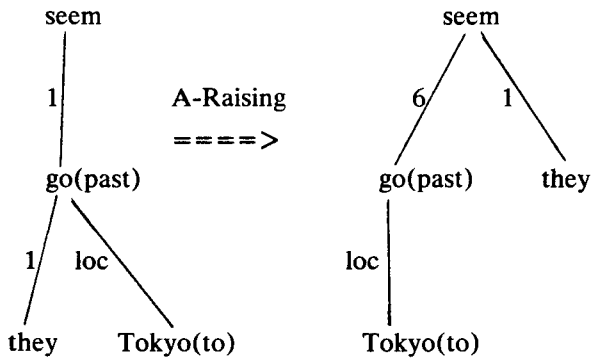
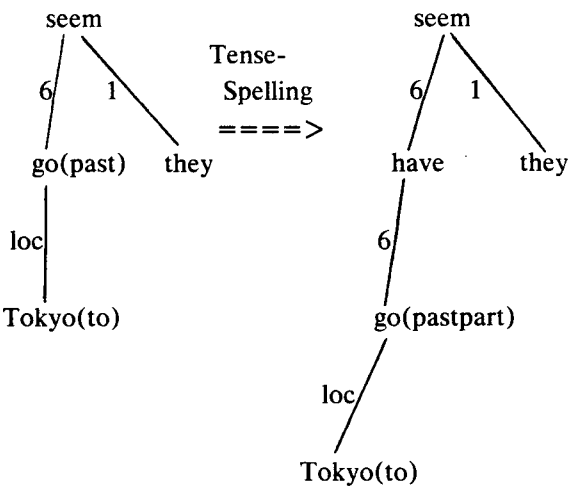


Figure 2. A-Raising Applied to the CS of (2) and (3). Note that "6" means "complement".



=== Linearization, etc. ===>

They seem to have gone to Tokyo

Figure 3. Rest of the Derivation of (2)

As illustrated above, RG, like TG, is a "multi-stratal" theory, i.e., clauses typically have more than one level of syntactic analysis, and these levels/strata are mediated by clause-level rules. In the case of TG, the structures are phrase-structure trees, and transformations map trees into trees; in the case of RG, the structures are edge-labelled trees (called **relational structures (RS)**), where the edge labels represent primitive relations, and the rules map RSs into RSs.

The use of multiple strata sets RG apart from functional frameworks such as FUG (Kay 1979) and LFG (Bresnan 1982), which also use primitive relations (functions), and from all other monostratal frameworks such as GPSG (Gazdar, et. al. 1985), whether functional or not. The manipulation of explicitly marked relations in unordered relational structures sets RG apart from TG. In our work on Japanese-English MT, the RG concept of multiple relational strata has proven to be of significant practical use — facilitating the design and development of a limited transfer component and a robust generation component, enhancing modularity, and allowing the linguistic processing to be conceptualized in a uniform fashion.

3- The RG Rule Writing Language: GEAR

One key aspect of our implementation of an RG generator is the **GEAR** rule-writing language. GEAR permits a grammar developer to write computationally powerful RG rules in a linguistically natural manner. GEAR rules identify grammatical objects via path specifications, of which there are two types: (1) **node-specifier**, consisting of a sequence of one or more relation names, and (2) **property-specifier**, consisting of a node-specifier followed by a property name. For instance, **1:1** indicates a node that is the subject of a node that is the subject of the node currently being processed (the **focus**) and **2.tense** denotes the value of the property **tense** of a node that is the direct object of the focus. GEAR path expressions are superficially similar to the expressions used in unification-based frameworks such as FUG and PATR (Shieber, et. al. (1983)). However, GEAR is not unification based, rather it provides a number of procedural operations, including node deletion and node creation.

Each rule consists of a sequence of statements, of which there are several types, e.g., **IF-THEN-ELSE**, **CALL**, **ON** and *restructuring* statements. **IF-THEN-ELSE** statements control the *rule internal* processing flow. **CALL** statements are used to invoke rules by name. An **ON** statement invokes a specified rule on a node reachable from the **focus** via a node-specifier.

There are several types of restructuring statement, e.g., **ASSIGN**, **CREATE**, **DELETE** and **COPY**. An **ASSIGN** statement is used to alter the relations of a node identified via a node-specifier; the new relation is also specified by a node-specifier. The core of GENIE's A-raising rule, whose relational changes are illustrated in Figure 2 above, is (using **6** for "complement"):

(ASSIGN 1 6) "Assign my subject as my complement"
(ASSIGN 6:1 1) "Assign my complement's subject as my subject"

The complete rule is shown in Figure 4.

```
%% Define the rule A-raising for intransitive verbs
(DEF-RULE A-Raising OF Intransitive-verb
%% If the A-raising rule switch is turned on
(IF (A-raise is 'yes)
%% then assign my subject as my complement
 THEN (ASSIGN 1 6)
%% and assign my complement's subject as my subject
      (ASSIGN 6:1 1)
%% and on my complement call the rule
%% which makes infinitives
      (ON 6 (CALL Make-Infinitive))))
```

Figure 4. GENIE's A-Raising rule

Creation, copying and deletion of nodes are also specifiable but space limitations preclude discussion.

4- The GENSHELL generator shell

Building on our experience with an earlier prototype developed by Schindler (1988), we have developed an NL-independent generator shell, **GENSHELL**, to facilitate the development of RG generators. For any given generator, grammar developers need only specify the designated grammatical relations, parts of speech, a part-of-speech hierarchy, dictionaries and grammars. GENSHELL takes this information and constructs a runtime generator.

One of the distinctive aspects of GENSHELL, due to Schindler (1988), is the concept of **category-driven processing**. In category-driven processing, parts of speech are represented as categories in a category hierarchy (POSH) and nodes in RSs are represented as objects which are instances of categories and thus can inherit properties via the POSH. Among the inheritable properties are grammar rules. For instance, the rules for Passive and Subject-to-Object Raising (so-called **B-Raising**; discussed later) would be associated with the class Transitive Verb, A-raising would be associated with the class Intransitive Verb, and Subject-Verb Agreement would be associated with the superordinate class Verb.

In our implementation, all rules are defined with respect to named **rule bundles**, and rule bundles are associated either with categories in the POSH, the general/default cases, or with lexical entries, the special cases. Rule definitions have the form:

(DEF-RULE rulename OF rule-bundle-name (rule-body)).

(As shown in Figure 4 above, a default rule bundle associated with a POS class is given the same name as that class.) When a node N associated with category C and lexical entry L is being processed, the rule search routine, given a rule named R — the latter comes from so-called **agenda** rules which are also associated with C — uses inheritance to first search for R among any rule bundles named in L, then searches for R among C's rules, then C's parent's rules and so on up to the top of the hierarchy until either some rule named R is found or the top category is reached and the process fails. In short, in category-driven processing, the grammar invoked on N is constructed as appropriate at processing time on the basis of lexically activated rules and the rules accessible to N's category using the POSH and inheritance.

One example is the ordering of adjectives and nouns. The class Noun is associated with a general/default linearization rule which orders adjectives before nouns, generating phrases like **tall woman**. Nouns like **someone**, **anyone**, etc. are associated with a lexically triggered linearization rule which places the adjective *after* the head noun. These two rules are both named **Linearize**. Thus, if the focus is **someone** and it is modified by **tall**, the search routine, looking for **Linearize**, will first find the special rule, correctly generating **someone tall**.

A category-driven system has two advantages over more conventional rule systems: (i) it provides a natural mechanism for dealing with special cases triggered by lexical items, while providing a fail-soft mechanism in the form of the general rules inherited from the POSH and (ii) only rules that in principle could be relevant to processing a given node in an RS will be tested for application. That is, the POSH provides a linguistically motivated means for organizing a large grammar into subgrammars.[3]

5- GENIE: the English generator

Generating from CSs requires a robust generation grammar of the target language, as well as a decision-making component that decides which surface form is to be generated. The generation grammar employed in GENIE is a (deterministic) relational grammar having a substantial number of **clause-level** rules which alter grammatical relations, e.g., Passive, A-raising and B-raising, as well as minor rules such as Tense-Spelling and Linearization (the latter of which does not alter grammatical relations).

As illustrated in Figure 1 above, CSs typically do not correspond directly to grammatical sentences. Further, any given CS typically constitutes the basis for the generation of a number of superficial forms, e.g., (2) and (3) above. This control problem has been addressed by splitting generation into two phases: a syntax planning phase and an execution phase. The function of GENIE's planner is quite different from that of other generators. Typically, generator planners decide "what to say", constructing some sort of internal representation that is then processed by a realization component. Typical planners will be concerned with chunking into sentences, topic selection and word choice (see, e.g., Appelt(1985), Danlos (1984), Hovy(1985), Kukich (1983), McKeown (1985), McDonald (1984)), and Mann (1983)).

In the case of JETS, however, since we are in the domain of transfer-based MT, all of these "high level" considerations are decided by the analysis and transfer components. In GENIE's case, the planner must, on the basis of a given CS, deal with a myriad of low-level syntactic conditions and their interactions (most of which have not been discussed or even recognized in the generation literature). Internal to GENIE, this means deciding which of the rules in the deterministic execution grammar should be applied. For instance, CSs with **seem** have a disjunctive grammatical condition: they must either be raised, yielding the pattern **NP seem to VP** (as in (2) above) , or extraposed, yielding the pattern **It seems that S** (as in (3) above). Failure to apply either A-raising or so-called **It**-Extraposition

[3] Earlier work using a lexical hierarchy and inheritance in natural language processing includes Wilensky (1981), Jacobs (1985) and Zernik and Dyer (1987). These works make heavy use of phrasal patterns (so-called pattern-concept pairs) and so the conception of grammar and lexicon and hence the notion of what is inherited in these works differ greatly from ours, which is part of the generative-linguistic tradition.

would result in the ungrammatical pattern ***That S seems** (in the case of Figure 1 above: ***That they went to Tokyo seems**). The decision to apply A-raising in the above example is stylistic ("make the topic the main clause subject, if possible"), but the disjunctive requirement ("apply either A-raising or It-Extraposition") is grammatical. Having no control over "what to say", GENIE's planner is conceptually part of the realization phase and not part of the typical "planning phase".

GENIE's planner communicates which rules should be applied to the execution grammar via a set of so-called **rule switches**, which are simply binary-valued properties whose property names are the names of execution rules, e.g., **(A-raise . Yes)**, **(Passive . No)**. As shown in Figure 4 above, **IF** statements are often used to test for a rule-switch value, which value is either set by a planning rule or comes from a lexical entry. Rule switches are a generalization of the earlier concept of transformational rule features (cf. Lakoff 1970); the generalization is that **rule switches can be dynamically set by planning rules, based on lexical, syntactic, semantic and stylistic considerations** (see Johnson 1988a for more examples and further discussion).[4]

For example, in (1) above, based on the information that **they** is the topic (this information comes from transfer), a syntax planning rule which is partly responsible for making topics surface subjects sets the switch **(A-raise . Yes)**, turning on A-raising, and the switch **(It-Extra . No)**, turning off **It-** extraposition, resulting in (2) rather than (3). GENIE's architecture is shown in Figure 5.

Planning rules insure that a multitude of lexico-syntactic and stylistic conditions are met, e.g., that clauses with modals do not undergo A-raising, preventing the generation of, e.g., ***They seem to can swim**; that clauses with verbs like **force** have passivized subordinate clauses where required to meet coreferential deletion conditions (cf. **She forced him to be examined by the doctor**, ***She forced him (for) the doctor to examine him**); and that verbs like **teach** undergo dative alternation if there is no specified direct object, generating **He taught her** rather than ***He taught to her** (cf. **sing**, which has the opposite condition - **He sang to her** but ***He sang her**).

It is also the responsibility of the planner to make sure island constraints are not violated. For instance, if a wh-nominal is in a sentential subject, then planning rules turn on execution rules such as A-raising resulting in sentences like **Who is likely to win** (via A-Raising) rather than ***Who is to win likely?** or the stylistically marginal **?Who is it likely (that) will win?**. This heuristic planning rule also insures that in the case of so-called Tough-Movement sentences, GENIE will generate sentences like **Who is easy to please?**, (via Tough-Movement) rather than either ***Who is to please easy?** or **?Who is it easy to please?**.

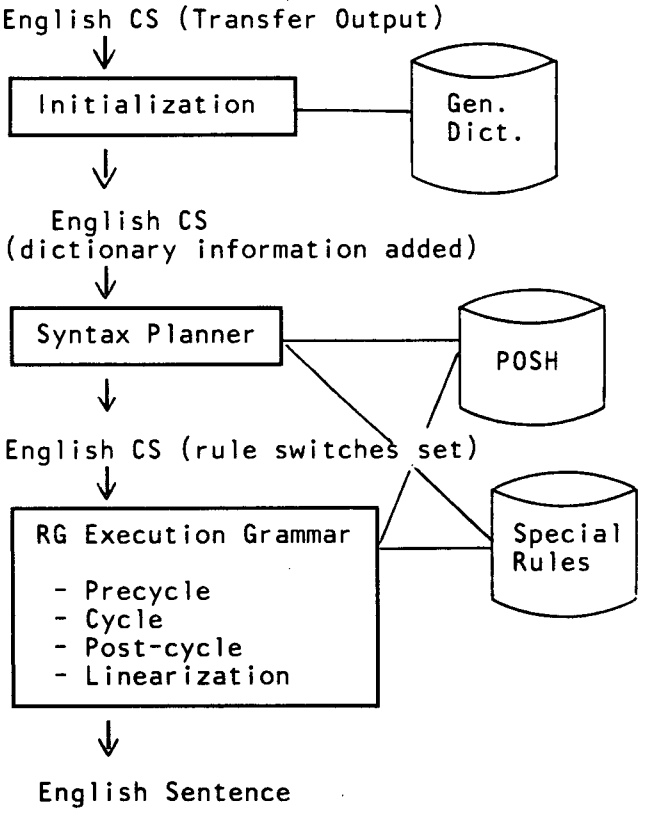


Figure 5. GENIE Components. Note that the POSH contains the agenda rules and the default planning and execution rules organized by POS.

[4] After completing this work, we discovered that Bates and Ingria (1981) also used a mechanism similar to our "rule switches" to control generation within a TG framework. Their transformational constraints, however, were set by a human who wished to test what a given set of constraints would produce. That is, their system had no syntax planner which would evaluate a given base structure via a set of planning rules and set constraints insuring the generation of only grammatical sentences.

Execution rules are turned on (or off) either by syntax planning rules or by lexical entries. To illustrate the use of lexical rule-switches, consider the following example from JETS involving verbs of prevention:

4. **kanojo wa kare ga iku no o habanda**
   she top he pp go nm pp prevent
5. **She prevented him from going.**

On the Japanese side, the postposition **ga** marks the subject of the embedded clause **kare ga iku**, which has been nominalized with the dummy noun **no**, which carries the direct object marker **o**. Following the arguments given in Postal (1974), we assume that **prevent** is a so-called B-raising trigger (B-raising is the controversial rule which relates sentences such as **He believes that she knows** (not raised) and **He believes her to know**, in which **her** is raised up as direct object of **believe**). The CS for (5) is as shown to the right in Figure 6 and the CS of the Japanese sentence (4) is shown to the left:[5]

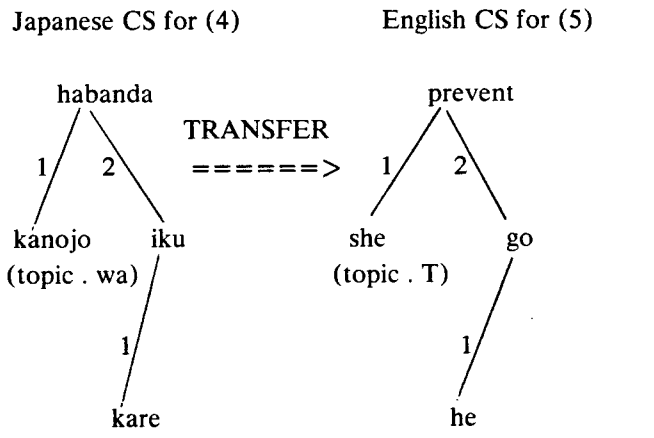


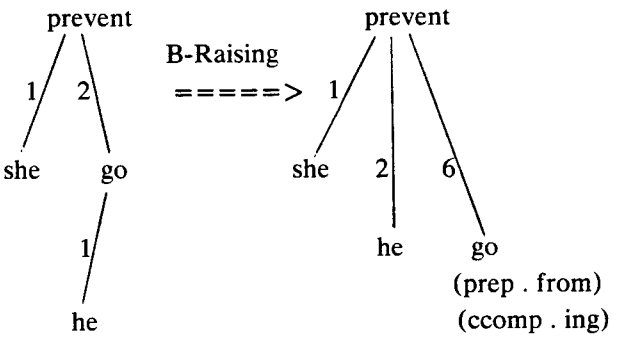
Figure 6. Canonical Structures for (4) and (5)

GENIE's rule of B-raising, given in Figure 7, maps the English CS into a superficial RS, as shown in Figure 8. As shown in Figure 6, the English and the Japanese CSs are isomorphic, i.e., there are no structural changes in transfer.

To produce (5) from the English CS in Figure 6, as illustrated in Figure 8, merely requires the dictionary entry depicted in Figure 9.

```
%% Define the rule B-raising for transitive verbs
(DEF-RULE B-Raising OF Transitive-Verb
%% If the B-raising rule switch is "yes"
 (IF (B-raise is 'yes)
%% then make my direct object my complement
  THEN (ASSIGN 2 6)
%% and make my complement's subject
%% my direct object
       (ASSIGN 6:1 2)
%% and on my complement call the rule
%% that makes infinitives
       (ON 6 (CALL Make-Infinitive))))
```

Figure 7. GENIE's B-Raising Rule



= Other Rules => She prevented him from going

Figure 8. Example of B-Raising Application

```
:lexical-form. prevent
:category. transitive-verb
:rep-lexical-form. nil
:rep-category. nil
:properties. (B-Raise . Yes) (cprep . from)(cvform . ing)
:additional-rule-sets. nil
```

Figure 9. Lexical entry for "prevent"

This lexical entry states that **prevent** is a transitive verb, hence has access to the rules defined for transitive verbs

[5] Postal's English-internal arguments were based on the fact that the direct object of **prevent** could be existential **there**, weather **it** and idiom chunks (cf. **She prevented there from being a riot/it from raining/the cat from being let out of the bag**).

in the POSH, e.g., Passive and B-raising (and the rules of superordinate classes), and that among its properties are the rule switch setting **(B-Raise . Yes)**, which triggers Subject-to-Object raising, the feature **(ccomp . from)**, which determines that the complement clause (fragment) will be flagged with **from** via a general rule, and the feature **(cvform . ing)**, which Make-Infinitive will use when called by B-Raising to determine the verb form **going** in the example. **Prevent** has no rep(lacement)-lexical-form, which is used, e.g., to map a single input form such as **look-up** into a verb **look** and a particle **up**, or more generally to map senses into lexical strings. "Rep-cat", also nil here, can be used to map one category system into another (not used in GENIE). "Additional-rule-sets", also nil, is the repository for the names of any rule bundles associated with a lexical entry (e.g., **easy**, **hard**, etc. would have the additional-rule-set name **tough-movement**, which contains the Tough Movement rule and the planning rule that turns Tough Movement on).

As depicted in Figure 5 above, the execution component consists of three relation-changing phases, called "pre-cycle", "cycle" and "post-cycle", in which execution rules are applied bottom-to-top, followed by a top-down linearization phase, which builds an output string that is then sent to the morphological component (not shown). Each phase has its own set of agenda rules, whose functions are to either call grammatical rules or shift control, i.e., agenda rules are a sequence of **CALL** statements. Agenda rules, like grammatical rules, are defined for classes, so that, e.g., the cyclic agendas for adjectives, nouns and verbs are different. For instance, part of the agenda for the cyclic phase of transitive verbs is: ... **(Call B-raising) (Call Dative) (Call Passive)** ..., but none of these rules are relevant to adjectives, nouns or intransitive verbs. It should be noted that rules called by a particular agenda might be accessed via inheritance. E.g., Reflexivization is called in the cyclic agenda for transitive verbs, but it is associated with the class Predicate so that it is available to adjectives in cases like **He is proud of himself** (it is assumed that Reflexivization is executed on the **proud** clause before A-Raising applies on **be**).

The grammar implemented in GENIE to date includes many of the important rules for English clause structure, including Yes/No questions, Wh-questions, relative clauses, subordinate clauses of various types, verb-particle combinations, raisings of various sorts, passives, and extrapositions.

6- Concluding Remarks

We have developed an application-and-NL-independent generator shell, GENSHELL, including a flexible dictionary system and a high-level rule-writing system, GEAR, to facilitate the development of category-driven RG generators. GENSHELL/GEAR provides a powerful computational framework for the development of RG-based natural-language-processing components. We have also implemented GENIE, a robust English generator, within GENSHELL/GEAR. Besides the novel use of RG and category-driven processing, GENIE is notable for its two-stage plan-and-execute design.

JETS and GENIE are currently being tested on sentences from Asahi newspaper editorials on economic matters, a challenging task since editorial sentences can be very long, with essentially unrestricted vocabulary. Nevertheless, we have found the initial tests of the generator encouraging, supporting our view that besides its intrinsic theoretical interest, RG has practical value in natural language processing.

References

Appelt, D. E. 1985. **Planning English Sentences**. ACL Series: Studies in Natural Language Processing. Cambridge UP, Cambridge.

Bateman, J., R. Kasper, J. Schütz and E. Steiner. 1989. "Interfacing an English Text Generator with a German MT Analysis," submitted to the European ACL, Manchester, 1989.

Bates, M. and R. Ingria. 1981. "Controlled Transformational Sentence Generation," **Proceedings of the 19th Annual Meeting of the Association for Computational Linguistics**, Stanford, CA.

Bresnan, J. (ed.) 1982. **The Mental Representation of Grammatical Relations**. MIT Press, Cambridge, Mass.

Danlos, L. 1984. "Conceptual and Linguistic Decisions in Generation," **Proceedings of COLING-84**, Stanford, pp. 501-504.

Dubinksy, S. and C. Rosen (eds) 1987. "A Bibliography on Relational Grammar Through 1987 with Selected Titles on Lexical-Functional Grammar,"

distributed by Indiana University Linguistics Club, Bloomington, Indiana.

Gazdar, G., E. Klein, G. Pullum and I. Sag. 1985. **Generalized Phrase Structure Grammar**. Harvard University Press, Cambridge, Mass.

Hovy, E. 1985. "Integrating Text Planning and Production in Generation," **Proceedings of IJCAI-85**, Los Angeles, CA.

Jacobs, P. S. 1985. **A Knowledge-Based Approach to Language Production**. PhD dissertation, UC Berkeley, Computer Science Division, UCB/CSD 86/254, Berkeley, CA.

Johnson, D. E. 1988a. "On the Linguistic Design of Post-Analysis in the JETS Japanese-English Machine Translation System", **Proceedings of the International Conference on Fifth Generation Computer Systems 1988**, Tokyo.

Johnson, D. E. 1988b. "A Relational Grammar Approach to Machine Translation," **Proceedings of the Information Processing Society of Japan, Natural Language Processing**, Vol. 88.61.

Kay, M. 1979. "Functional Grammar," **Proceedings 5th Annual Meeting of the Berkeley Linguistics Society**, Berkeley, CA, pp. 142-158.

KBMT-89. 1989. **KBMT-89 Project Report**, Center for Machine Translation, Carnegie Mellon University.

Kukich, K. 1983. **Knowledge-Based Report Generation: A Knowledge Engineering Approach to Natural Language Report Generation**. PhD dissertation, Information Science Department, University of Pittsburgh.

Lakoff, G. 1970. **Irregularity in Syntax**. Holt, Rinehart, Winston, New York.

Mann, W. 1983. "An Overview of the Penman Text Generation System," **Proceedings of the National Conference on Artificial Intelligence**, pp. 261-265.

Maruyama, H., H. Watanabe, and S. Ogino. 1989. "An Interactive Japanese Parser for Machine Translation," **Proceedings of COLING90**, Helsinki, to appear.

McDonald, D. 1984. "Description Direct Control: Its Implication for Natural Language Generation," in N. J. Cercone (ed.) **Computational Linguistics**, Pergamon Press, Oxford, pp. 403-424.

McKeown, K. 1985. **Text Generation**. Cambridge University Press, Cambridge.

Nirenburg, S. 1987. "A Distributed Generation System for Machine Translation," Technical Report, Center for Machine Translation, Carnegie Mellon University.

Nirenburg, S., R. McCardell, E. Nyberg, P. Werner, S. Huffman, E. Kenschaft, and I. Nirenburg. 1988. "Diogenes-88," Technical Report, Center for Machine Translation, Carnegie Mellon University.

Postal, P. M. 1974. **On Raising**. MIT Press, Cambridge.

Rösner, D. 1986. "When Mariko Talks to Siegfried - Experiences from a Japanese/German Machine Translation Project," **Proceedings of COLING-86**, Bonn.

Schindler, Peter A. 1988. "General: An Object-Oriented System Shell for Relational Grammar-Based Natural Language Processing", master's thesis, Department of Electrical Engineering and Computer Science, MIT.

Shieber, S. M., H. Uszkoreit, F.C.N. Pereira, J.J. Robinson and M. Tyson. 1983. "The Formalism and Implementation of PATR-II," In **Research on Interactive Acquisition and Use of Knowledge**, AI Center, SRI International, Menlo Park, CA.

Wilensky, R. 1981. "A Knowledge-Based Approach to Natural Language Processing: A Progress Report," **Proceedings Seventh International Joint Conference on Artificial Intelligence**, Vancouver.

Zernik, U. and M. G. Dyer. 1987. "The Self-Extending Phrasal Lexicon," **Computational Linguistics**, vol. 13, No. 3-4, pp. 308-325.