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

David E. JOHNSON\textsuperscript{a} and Hideo WATANABE\textsuperscript{b}

\textsuperscript{a} IBM Research, Thomas J. Watson Research Center, Yorktown Heights
\textsuperscript{b} IBM Research, Tokyo Research Laboratory, Tokyo

ABSTRACT: This paper describes the design and function of the English generation phase in JETS, a minimal transfer, Japanese-English machine translation system that is based on the linguistic framework of relational grammar. To facilitate the development of relational grammar generators, we have built a generator shell that provides a high-level relational grammar rule-writing language and is independent of both the natural language and the application. The implemented English generator (called 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. The modularity inherent in the plan-and-execute design permits the development of a very general and stable deterministic execution grammar. Another major feature of the GENIE generator is that it is category driven, i.e., planning rules and execution rules are distributed over a part-of-speech hierarchy (down to individual lexical items) and are invoked via an inheritance mechanism only if appropriate for the category being processed. Category-driven processing facilitates the handling of exceptions. The use of a syntactic planner and category-driven processing together provide a great deal of flexibility without sacrificing determinism in the generation process.

KEYWORDS: generation, natural language processing, relational grammar

1. INTRODUCTION

This paper discusses relational-grammar-based generation in the context of JETS, a general purpose Japanese-to-English machine translation 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 and then sketch the processing flow in translation. With this background, we (i) describe and illustrate certain aspects of the relational grammar rule-writing language, GEAR, in which the GENIE English generator has been written; (ii) comment on key aspects of the generator shell, GENESHELL, in which GENIE has
been developed; and (iii) discuss the design and function of the GENIE English generator.

With relatively few exceptions, generation within the area of machine translation (MT) has received comparatively little attention. It is not uncommon for an MT system to have a highly articulated analysis grammar, but a quite sketchy generation grammar that would not qualify as an independently justified, linguistically adequate grammar of the target language. 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.

We have sought in our work to shift as much of the processing burden as possible from transfer onto generation, facilitating the incorporation into the generation phase of a variety of analyses and insights that have come from theoretical linguistics. The primary methodological assumptions guiding our work have been two: (i) generation should embody a linguistically justified and independently functioning grammar of the target language, and (ii) the role of structural transfer should be reduced to a bare minimum. In our view, observance of these two principles is essential for the development of linguistically adequate MT systems. This general view, which we refer to loosely as minimal transfer, is, of course, not new. It can be traced, in part, back to the CETA project at the University of Grenoble, which was greatly influenced by Mel'čuk's meaning-text model, and has roots in the early work at the Linguistics Research Center at the University of Texas, which was heavily influenced by early transformational generative grammar (Hutchins, 1986, Chapter 10). The historical source for our position comes directly from generative linguistics, including both Chomskyan transformational grammar and relational grammar.

As is no doubt obvious, however, increasing the role of generation does come at a cost — as the input structures (transfer output) become more abstract and the "generation path" becomes longer, the control problem increases (cf. McDonald, 1987). The innovative aspect of our approach resides mainly in the combined use of (i) a plan-and-execute design for solving the control problem; (ii) relational grammar as the underlying linguistic framework; and (iii) category-driven processing, described later, as the means for organizing a large grammar into subgrammars and for handling exceptional cases.

The key relational grammar notion in our work is that of canonical (relational)}

1The exceptions include the GETA-ARIANE project (Boitet et al., 1985); the work being done at Carnegie Mellon University (cf. KBMT-89, 1989; Nirenburg, 1987; and Nirenburg et al., 1988); the SEMSYN project at the University of Stuttgart (Rösner, 1986); the Rosetta Project (Landsbergen et al., 1989); and the joint effort of the ISI Penman project and the University of Saarbrücken (Bateman et al., 1989)

2The felicitous term "minimal transfer" was suggested to us by Jaime Carbonell.

3The "control problem" in MT involves two related considerations: (i) the complexity of potential rule interactions, which can increase dramatically when generating from highly abstract (language-neutral) structures, and (ii) the decision as to which surface sentence to generate.
structure, 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 and chomeur. Given the basic assumption that one is developing a minimal transfer system — implying deep analyses of both the source and target languages that converge, in a wide range of cases, on structurally similar internal representations for translation equivalents — it is critical to select a linguistic framework that supports the required analyses and enables one to conceptualize the linguistic processing in a uniform manner. As discussed in Johnson (1988b), with respect to MT, relational grammar is a logical choice of linguistic framework since canonical structures provide a relatively language-neutral syntactic bridge between languages as diverse in structure as Japanese and English. This is so for two reasons: (i) within one language, the canonical structures of paraphrases are often the same or highly similar, and (ii) translation equivalents often have structurally similar if not isomorphic canonical structures.

One of the major advantages of relational grammar comes from its explicit representation of grammatical relations like subject, direct object and indirect object, which are argued to be universal. In contrast, structure-based frameworks such as transformational-generative grammar at best only implicitly represent grammatical relations in terms of linear precedence and dominance, which are language particular. If one considers the task of transfer between two languages as diverse in structure as Japanese and English, 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, can substantially reduce the amount of structure changing required in the transfer component. By analyzing both languages within the multistratal framework of relational grammar, we can provide a linguistically motivated, language-neutral syntactic representation of a broad range of clause types — that is, it is possible to approach a pivot representation in a large number of cases without being committed to mapping into and out of a yet more highly abstract semantic interlingua.

Relational grammar was first proposed around 1972-73 by David M. Perlmutter and Paul M. Postal (cf. Perlmutter and Postal, 1974; Postal, 1974). For background on relational grammar, see Blake (1990); Johnson (1977b, 1979); Perlmutter (1980); Perlmutter (1983); Perlmutter and Rosen (1984); Postal and Joseph (1990); as well as the many articles listed in the bibliographic reference Dubinsky and Rosen (1987).

Relational grammar postulates many more primitive grammatical relations than subject, direct object and indirect object, e.g., chomeur, genitive, predicate, union and various oblique relations such as benefactive, locative and temporal. There is no motivated basis for configurational definitions of most of these grammatical relations (for some discussion on this issue, cf. Johnson, 1977a, 1979; for a longer list of relations and discussion, cf. Johnson and Postal, 1980: 198 and passim).
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 canonical structure, which represents the basic predicate-argument structure of the Japanese sentence. Transfer produces an English canonical structure, which is often, but not always, isomorphic to the Japanese canonical structure. The English canonical structure is passed to the GENIE generator, whose task is to generate a grammatically correct and stylistically appropriate English sentence given a well-formed canonical structure.

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 unordered Japanese canonical structure shown at the left in Figure 1. Transfer then maps the Japanese canonical structure into the (isomorphic) unordered English canonical structure shown at the right in Figure 1.

---

6 For discussion of parsing in JETS, see Manuyama, Watanabe and Ogino (1990).

7 For example, as discussed in Johnson (1988b), it is far from clear that Japanese "affected (adversity) passives" such as those shown below have, in all cases, canonical structures isomorphic to their English counterparts. If one adopts the view that all such clauses involve higher predicates representing the relation of "affectedness" between the affected nominal and the rest of the clause, then a uniform analysis could be given. However, there is no apparent syntactic motivation for this analysis in (i.c), as opposed to (ii.c), which involves the higher predicate have. (Note that the following abbreviations are used in glosses of Japanese examples: 'DO' direct object, 'NM' nominalizer, 'TOP' topic and 'PP' preposition/postposition.)

i. a. watashi wa ame ni furareta
   b. I TOP rain by fell (passive-past)
   c. I was rained on. (It rained on me.)
   d. *I had it rain (on me).

ii. a. watashi wa sensei ni e o homerareta
   b. I TOP teacher by picture DO praise (pass-past)
   c. I had (a) picture praised by (the) teacher.
   d. *The teacher praised a picture on me.

Other mismatches involve certain uses of the various Japanese verbs of giving that have no direct parallel in English; for example:

i. a. Haha wa keiki o yaite kureta
   b. mother TOP cake DO baking gave
   c. My mother baked a cake for me.
Given the English canonical structure, it is up to the GENIE English generator to generate either (2) or (3). Based on the information that they in the English canonical structure is marked as the topic of the sentence, GENIE will map the canonical structure into the (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.\(^8\)

As illustrated in Figures 2 and 3, relational grammar, like transformational grammar, is a multistratal theory, i.e. clauses typically have more than one level of syntactic analysis and these strata are mediated by clause-level rules. In the case of transformational grammar, the structures are phrase structure trees, and transformations are operations on phrase structure trees; in the case of our version of relational grammar, the structures are edge-labeled trees (called relational structures), where the edge labels represent primitive grammatical relations, and the rules are operations on unordered relational structures.\(^9\)

\(^8\)Information regarding tense, verb form, prepositions, etc. is carried along as features on nodes. In the example in Figure 3, since the complement clause has been definitezed by A-raising, the past tense feature is realized as the verb have and the feature (vform . pastpart) on the complement verb go.

\(^9\)We say "our version" because relational grammar is, in fact, a non-derivational theory in which rules are considered to be well-formedness conditions on "relational networks" (which are not generally trees in that a node can bear relations to more than one mother node) (cf. Johnson and Postal, 1980: Chapter 2). The version reported on here most closely corresponds to the derivationally based framework described in Johnson (1979). It was selected for practical reasons, since at the time this
Figure 2. A-raising applied to the canonical structure of (2) and (3).

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

3. THE GEAR RULE-WRITING LANGUAGE

One key aspect of our implementation of a relational grammar generator is the GEAR rule-writing language. GEAR permits a grammar developer to write computationally powerful relational grammar rules in a linguistically natural manner. GEAR rules identify grammatical objects via path specifications, of which there are two types: (i) A node-specifier, consisting of a sequence of one or more relation names, and (ii) a property-specifier, consisting of a node-specifier followed by a property name. For instance, SU:SU indicates a node that is the subject of a node that is the subject of the node currently being processed (the focus).
and \texttt{DO.tense} denotes the value of the property (\texttt{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 functional unification grammar 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., \texttt{IF-THEN-ELSE}, \texttt{CALL}, \texttt{ON} and \texttt{restructuring} statements. \texttt{IF-THEN-ELSE} statements control the \textit{rule internal} processing flow. \texttt{CALL} statements are used to invoke rules by name. An \texttt{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. \texttt{ASSIGN}, \texttt{CREATE}, \texttt{DELETE} and \texttt{COPY}. An \texttt{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, is the following:

$$\text{ASSIGN CMP:SU SU}$$

"Assign my complement's subject as my subject"

The complete rule is given just below. Notice that the core of the rule consists of
a maximally simple statement reassigning the relation of the complement subject:

(DEF-RULE A-raising OF intransitive-verb ;Define rule for intransitive verbs.
  (IF (A-raise is 'yes) ;If A-raising rule switch is turned on,
    THEN ((ASSIGN CMP:SU SU) ;assign my complement’s subject as my subject,
      (ON CMP (CALL make-infinitive))) ) ) ;and, on my complement, call the rule
that makes infinitives.

Unlike phrase structure transformations, constituents are not identified configurationally and no reordering is involved; hence rules do not need to (and cannot) specify linearly intervening constituents, and there are no problems determining derived constituent structure. That is to say, a number of the problems often associated with phrase structure transformations are avoided. Furthermore, relational rules such as A-raising are good candidates for universal rules. The superficial “side-effects” — word order and morphological reflexes — of a rule such as A-raising differ from language to language, but the relation-changing operation is language neutral. (This kind of result was, in fact, the original motivation for the development of relational grammar.)

As mentioned earlier, creation, copying and deletion of nodes are also specifiable but discussion is beyond the scope of this paper (see Johnson and Watanabe, 1990).

4. THE GENERATOR SHELL

Building on our experience with an earlier prototype developed by Schindler (1988), we have developed a generator shell, GENSHELL, to facilitate the development of relational grammar generators. GENSHELL is independent of both the natural language and the application. 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; nodes in relational structures are represented as objects which are instances of categories and thus can inherit properties via the hierarchy. 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 which are associated either with categories in the category hierarchy, the general (default) cases; or with lexical entries, the special cases. Rule definitions have the
form:

(DEF-RULE rule-name OF rule-bundle-name (rule-body)) \(^{10}\)

When a node \(N\) associated with category \(C\) and lexical entry \(L\) is being processed, the rule search routine, given a rule name \(R\), uses inheritance to search for \(R\) first among any rule bundles named in \(L\), then among \(C\)'s rules, then \(C\)'s parents' 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.\(^{11}\) 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 through the category hierarchy.

One example is the ordering of adjectives and nouns. The class "noun" is associated with a general 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 similar strategy is used to order the adverb enough, which exceptionally follows the adjective it modifies (compare very big/big enough and very oddly/oddly enough).

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 category hierarchy; and (ii) only rules that in principle could be relevant to processing a given node in a relational structure will be tested for application. That is, the category hierarchy provides a linguistically motivated means for organizing a large grammar into subgrammars.\(^{12}\)

5. GENIE: THE ENGLISH GENERATOR

Generating well-formed sentences from canonical structures requires a fairly complicated generation grammar. The complexity of the grammar can be substantially simplified through the use of a syntax planner that evaluates input canonical structures and determines which surface form is to be generated. The generation

\(^{10}\)As shown in the trace of the A-raising rule from the previous section, a default rule bundle associated with a category hierarchy class is given the same name as that class.

\(^{11}\)The rule name \(R\) comes from agenda rules which are also associated with the category \(C\).

\(^{12}\)Earlier work using a lexical hierarchy and inheritance in natural language processing includes that of Wilensky (1981), Jacobs (1985) and Zemik and Dyer (1987). These works make heavy use of phrasal patterns ("pattern-concept pairs") and so the conceptions 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.
grammar employed in GENIE is a deterministic relational grammar having a substantial number of clause-level rules altering grammatical relations, e.g., passive, A-raising, B-raising, object raising, various extrapositions, as well as minor rules such as tense-spelling and preposition-spelling.\(^{13}\) Linearization, however, does not alter grammatical relations; it is simply a top-down depth-first recursive procedure that creates an output string on the basis of the word-order specifications associated with the categories in the final relational structure. That is, unlike many other generators, GENIE never creates a phrase structure tree. In our experience, conversion to phrase structure representation is an unnecessary extra step, and so we have eliminated it.

As illustrated in Figure 1, canonical structures typically do not correspond directly to grammatical sentences. Further, any given canonical structure 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 such matters as 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 canonical structure, deal with a myriad of low-level syntactic conditions and their interactions (most of which appear not to have been considered in the literature on generation). Internal to GENIE, this means deciding which of the rules in the execution grammar should be applied. For instance, canonical structures 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 it-extraposition would result in the ungrammatical pattern *That S seems — in the case of Figure 1, *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...

\(^{13}\)By deterministic we mean that if the conditions for a rule’s application are met, then that rule invariably applies, resulting in exactly one output for any input. While we agree with the view of the Rosetta project (Landsbergen, 1984; Landsbergen, Odijk and Schenk, 1989) that the notion possible translation is theoretically of interest — and we can think of possible scenarios where non-determinism might be useful in principle — such an approach would require, in a practical system, the use of a filtering mechanism. As far as we can see, this would introduce efficiency problems and significantly complicate the design of the system. Since we are interested in building practical MT systems incorporating insights from theoretical linguistics, we have made determinism of the execution grammar (and planner) a design requirement.
"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 the example of GENIE's A-raising rule on page 8, 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).

For example, in (1) above, based on the information that they is the topic (this information comes from transfer), a syntax planning rule (called "A-raise/it-extra"), 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). A-raise/it-extra is invoked lexically by the rule-switch (A-raise/it-extra.yes), which would be part of the lexical information of intransitive verbs like seem and appear that take canonical complements but no canonical subject. Adjectives like likely, as opposed to probable, might also invoke this rule. This planning rule is displayed below. Note that if the complement subject is a topic, then A-raising is chosen over it-extraposition, unless previously determined otherwise.

After completing this work, we learned that Bates and Ingria (1981) also used a mechanism similar to our "rule switches" to control generation within a transformational grammar 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 to evaluate a given base structure via a set of planning rules and set constraints, ensuring the generation of only grammatical sentences.

One might well wonder, as did one referee, how one ensures that the planning rules are consistent. This can be interpreted in at least three ways, which we address briefly. Interpretation 1: What prevents two rules from setting opposing values for some rule-switch? The answer here is simply that the system will not permit any node to have two instances of the same property; therefore, for any given rule-switch, it either fails to appear on a node or else it appears and has precisely one of the two possible values. Interpretation 2: What prevents one planning rule from resetting a (necessary) rule-switch value? The answer here is that nothing in the GENESIS framework would prevent one planning rule from resetting a given rule-switch value (or from deleting a rule-switch name). This is up to the grammar developer. After confronting this possible problem, it seemed to us that the following methodological principle is sound: A rule switch setting can never be changed. To enforce this, one must, of course, always test for a given rule-switch before setting it. Interpretation 3: What prevents two planning rules from setting two different rule-switches that invoke the testing of two incompatible execution rules? The answer here is, "nothing." If this were to occur, then the first rule mentioned in the relevant agenda rule would be tested for application first and, if applicable, would apply. The second rule, although it would pass its rule-switch test, might not apply at all. However, it is possible that only part of the second rule would apply, creating an ill-formed structure. To ensure
architecture is shown in Figure 4.

(DEF-RULE A-raise/it-extra OF predicate; Define rule for predicates.
(IF (AND (A-raise/it-extra is 'yes') ; If rule switch is yes,
    (CMP : SU . topic i e 'true) ; and my complement's subject is a topic,
    (it-extra isnot 'yes) ; and there is no rule-switch setting (it-extra .yes),
    i.e., either it is missing or it is set to no,
    (A-Raise isnot 'no)) ; and there is no rule-switch setting (A-raise .no)...
THEN (AND (SET-VALUE A-raise 'yes) ; then set (A-raise. yes),
    (SET-VALUE it-extra 'no) ) ) ) ; and (it-extra .no)

Planning rules ensure 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 and *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.16 It is also the responsibility of the planner to make sure that 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 ensures that in the case of object-raising sentences, GENIE will generate sentences like Who is easy to please? (via object raising), rather than either *Who is to please easy? or ?Who is it easy to please?.

Execution rules are turned on (or off) either by syntax planning rules or by lexical entries. Determinism in the execution grammar is effected as follows. Execution rules are classified as either unmarked or marked. The first part of an unmarked rule with rule-switch name UR would be (IF (UR is-not ' no) THEN ... ) ; i.e., UR would be tested for application unless its rule-switch is that this will not occur, the grammar developer must set the rule switch of the second rule to no via the planning rule for the first rule (or conceivably in the execution rule itself). In practice, we have found it easy to ensure that there are no undesirable interactions of this sort. But the general answer is that it is up to the grammar developer to understand the effects of the various rules and to make sure the rule-switches are set accordingly.

16In English, verbs taking both direct and indirect objects can differ in their behavior when the (logical) direct object is unspecified. Compare, for example, the patterns available to sing with those available to teach:

i. She taught him math. / She taught him math.
ii. She sang songs to him. / She sang him songs.
iii. *She taught him. / She taught him.
iv. She sang him. / *She sang him.

Japanese does not exhibit this alternation, that is, the correspondents to (iii) and (iv) would have superficial indirect objects, i.e., they would be analogous to the first one in each pair.
turned off by either a lexically specified rule or a planning rule. Examples of unmarked rules include those forming relative clauses and questions. Similarly, the first part of a marked rule MR would be (IF (MR is 'yes) THEN ...); i.e., MR would be tested for application only if its rule-switch has been turned on by either a lexically invoked rule or a planning rule. Examples of marked rules include passive, A-raising, B-raising and object raising. In GENIE, if a rule meets the conditions for applicability, it necessarily applies, resulting in at most one
sentence generated for each input. GENIE will, in fact, generate some output for any input so long as the input structure is a well-formed relational structure. There is, however, no guarantee that a grammatical structure will result, since every well-formed relational structure is not necessarily a bona fide canonical structure for some English sentence. We have intentionally made the notion of relational structure quite weak, since transfer sometimes presents GENIE with input that is defective in one way or another from the viewpoint of English grammar. In such a case, it is not unusual for GENIE to generate perfectly well-formed English on the basis of flawed input from transfer.

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 in turn carries the direct object marker o. Following the arguments given in Postal (1974), we assume that prevent is a subject-to-object or 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 canonical structure for the English sentence (5) is as shown to the right in Figure 5 and the canonical structure of the Japanese sentence (4) is shown to the left.17

GENIE’s rule of B-raising, given just below, maps the English canonical structure into a superficial relational structure, as shown in Figure 6. As was shown in Figure 5, the English and Japanese canonical structures are isomorphic, i.e., there are no structural changes in transfer.

(DEF-RULE B-raising OF transitive-verb; Define rule for transitive verbs.
  (IF (B-raise is 'yes); If the rule switch is yes,
    THEN (ASSIGN DO CMP); then make my direct object my complement,
    (ASSIGN CMP:SU DO); make my complement’s subject my direct object and...
    (ON CMP (CALL make-infinitive)))); on my complement, call the rule that makes infinitives.

To produce sentence (5) from the English canonical structure in Figure 5 (as illustrated in Figure 6) merely requires the following dictionary entry for prevent.

17Postal's main English-internal arguments were based on the existential there, the weather it and idiom chunks, all of which can be direct objects of prevent (cf. She prevented there from being a riot/it from raining/the cat from being let out of the bag).
This lexical entry states that \textit{prevent} is a transitive verb and hence has access to the rules defined for transitive verbs in the category hierarchy, 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 will be flagged with \textit{from} via a general rule; and the feature (cvform . ing), which the rule make-infinitive will use when called by B-raising to determine the verb form \textit{going} in the example. \textit{Prevent} has no rep(acement)-lexical-form, which is used, e.g., to map a single input form such as \textit{look-up} into a verb \textit{look} and a particle \textit{up}; or more generally to map senses into lexical strings. rep-category, 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. The general points to be noted here are (i) it is a simple matter to influence the behavior of the planner and the execution grammar through the rule-switch mechanism, and (ii) the combined use of the syntax planner and the rule-switch mechanism permits the rules in the execution grammar to be deterministic and essentially conditionless, without creating a rigid system.\footnote{\textsc{Genshell} permits one to establish a set of dictionaries and to specify in what order they should}
Figure 6. Example of B-raising to produce “She prevented him from going.”

As depicted in Figure 4, 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). The pre-cycle is used only to make any necessary readjustments to the output of transfer; this is motivated by the practical considerations of modularity and independence of function. 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 is relevant to adjectives, nouns or intransitive verbs. It should be noted that rules called by a particular agenda might be accessed via inheritance. For instance, 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 includes many of the rules needed for English clause structure, including yes/no questions, wh-questions, relative clauses, subordinate clauses of various types, verb-particle combinations, raisings of vari-

---

16 Whether passive is, in fact, relevant to intransitive verbs depends on one’s analysis of so-called “pseudo-passives” such as This bed has been slept in and This bridge has never been flown under. We ignore this issue here; for detailed discussion, see Postal (1986).
ous sorts, passives and extrapositions.

6. CONCLUDING REMARKS

We have developed a generator shell, GEN SHELL, that is independent of natural language and application. GEN SHELL includes a flexible dictionary system, the high-level relational grammar rule-writing system GEAR and a mechanism for category-driven processing; it provides a powerful computational framework for the development of relational-grammar-based generators. Within the GEN SHELL environment, we have also implemented GENIE, a highly articulated English generator which employs a pipeline plan-and-execute design for resolving the many lexico-syntactic and stylistic decisions that inevitably result when generating from highly abstract (often language neutral) canonical structures.

While relational grammar has been recognized for its contributions to theoretical linguistics, its potential for use in computational systems has not been widely explored. Other than the application of relational grammar to analysis described in Bayer (1986) and its use in Maybury's GENNY system (Maybury, 1989), the work described here is the only other natural language processing effort we are aware of that employs relational grammar. 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, relational grammar has practical value in natural language processing.

REFERENCES

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

Bayer, S., C.E. Kalish and L.E. Joseph. 1986. Grammatical relations as the basis for language parsing and text understanding. Proceedings of AAAI-5, Philadelphia, 788-790.

Bateman, J., R. Kasper, J. Schütz and E. Steiner. 1989. Interfacing an English Text Generator with a German MT Analysis. Paper submitted to the European ACL-4, Manchester.

Bates, M., and R. Ingria. 1981. Controlled Transformational Sentence Generation. Proceedings of ACL-19, Stanford.

Blake, B.J. 1990. Relational Grammar. London: Routledge.

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

Danlos, L. 1984. Conceptual and Linguistic Decisions in Generation. Proceedings of COLING-10, Stanford, 501-504.
Dubinsky, S., and C. Rosen (eds.) 1987. A Bibliography on Relational Grammar Through 1987 with Selected Titles on Lexical-Functional Grammar. Indiana University Linguistics Club, Bloomington, IN.

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

Hutchins, W.J. 1986. Machine Translation: Past, Present, Future. Chichester, England: Ellis Horwood.

Hovy, E. 1985. Integrating Text Planning and Production in Generation. Proceedings of IJCAI-9, Los Angeles.

Jacobs, P. S. 1985. A Knowledge-Based Approach to Language Production. Ph.D. dissertation, University of California, Berkeley, Computer Science Division, UCB/CSD 86/254.

Johnson, D.E. 1974. On the Role of Grammatical Relations in Linguistic Theory. Proceedings of the Chicago Linguistics Society 10, 269-283.

Johnson, D.E. 1977a. On Keenan's Definition of Subject Of. Linguistic Inquiry 8: 673-692.

Johnson, D.E. 1977b. On Relational Constraints on Grammars. In P. Cole and J. Saddock (eds.), Syntax and Semantics 8: Grammatical Relations. New York: Academic Press, 151-178.

Johnson, D.E. 1979. Toward a Theory of Relationally-Based Grammar. New York: Garland.

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, Tokyo, 1150-1158.

Johnson, D.E. 1988b. A Relational Grammar Approach to Machine Translation. Proceedings of the Information Processing Society of Japan, Natural Language Special Interest Group, Vol. 88.61, 1-9.

Johnson, D.E., and L.S. Moss. 1991. Foundations of Multistratal Axiomatic Grammar. IBM Research Report RC16444.

Johnson, D.E., and P.M. Postal. 1980. Arc Pair Grammar. Princeton: Princeton University Press.

Johnson, D.E., and H. Watanabe. 1990. GENshell — A Shell for Natural Language Generation Systems. IBM Research Report RT10036.

Kay, M. 1979. Functional Grammar. Proceedings of the 5th Annual Meeting of the Berkeley Linguistics Society, 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. Ph.D. dissertation, Information Science Department, University of Pittsburgh.
Lakoff, G. 1970. *Irregularity in Syntax*. New York: Holt, Rinehart, Winston.

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

Maruyama, H., H. Watanabe and S. Ogino. 1990. An Interactive Japanese Parser for Machine Translation. Proceedings of *COLING-13*, Vol. 2, Helsinki, 257-262.

Maybury, Mark T. 1989. Knowledge Based Text Generation. Rome Air Development Center, Report RADC-TR-89-93.

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

McDonald, D. 1987. Natural Language Generation: Complexities and Techniques. In S. Nirenburg (ed.), *Machine Translation: Theoretical and Methodological Issues*. Cambridge: Cambridge University Press, 192-224.

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

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. Kenschaf and I. Nirenburg. 1988. *DIOGENES-88*. Technical Report, Center for Machine Translation, Carnegie Mellon University.

Perlmutter, D.M. 1980. Relational Grammar. In E. Moravcsik and J. Wirth (eds.), *Syntax and Semantics 13: Current Approaches to Syntax*. New York: Academic Press, 195-230.

Perlmutter, D.M. (ed.) 1983. *Studies in Relational Grammar 1*. Chicago: University of Chicago Press.

Perlmutter, D.M., and C.G. Rosen (eds.) 1984. *Studies in Relational Grammar 2*. Chicago: University of Chicago Press.

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

Postal, P.M. 1986. *Studies of Passive Clauses*. Albany: State University of New York Press.

Postal, P.M., and B.D. Joseph (eds.) 1990. *Studies in Relational Grammar 3*. Chicago: University of Chicago Press.

Rösner, D. 1986. When Mariko Talks to Siegfried — Experiences from a Japanese/German Machine Translation Project. Proceedings of *COLING-11*, Bonn, 652-654.

Schindler, P.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. *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 of IJCAI-7, Vancouver, 25-30.

Zernik, U., and M.G. Dyer. 1987. The Self-Extending Phrasal Lexicon. *Computational Linguistics* 13: 308-325.