自然言語処理 70－6
（1989. 1. 20）

英語生成プログラムGENIEにおける「ルール・ガバメント」

デビッド E. ジョンソン
日本IBM 東京基礎研究所

本論文では、「ルール・ガバメント」と一般に呼ばれる問題の解決策として日英機械翻訳システムJETSの英語生成プログラムGenieの採用した方式について述べる。「ルール・ガバメント」とは、構文生成のうち形態論・統語論・スタイル上の要因によって制御されるものを指す。Genieの主な特徴としては、1）関係文法に基づいている，2）プランと実行の2段階を区別した設計になっている，3）「カテゴリー・ドリブン」な処理方式の原理を基礎としている，ことがある。以上の3つの特徴が「ルール・ガバメント」に関する生成上の問題点の解決に役立っている。

RULE GOVERNMENT IN THE GENIE ENGLISH GENERATOR

David E. Johnson
IBM Tokyo Research Laboratory
5-19 Sanbancho, Chiyoda-ku, Tokyo 102, Japan

This paper discusses the approach taken in **Genie**, the English generator for the **JETS** Japanese/English machine translation system, to problems broadly characterizable as **rule government**. Rule government refers to cases in which the generation of some construction is controlled by lexical, syntactic or stylistic factors. Some of the distinctive aspects of Genie are that it is based on the theory of relational grammar, has a two stage plan-and-execute design, and is based on the principle of category-driven processing. Together, these factors permit the solution to a number of generation problems involving rule government.

1.0 Introduction

This paper discusses the approach taken in **Genie**, the English generator for the **JETS** Japanese/English machine translation system, to problems broadly characterizable as **rule government**. Rule government, as used here, basically refers to cases in which the generation of some construction is controlled by lexical, syntactic or stylistic factors. Simple cases involve absolute lexical requirements, e.g., **want** does not passivize, **explain** does not dativize. Other cases involve complex iteractions of lexical requirements and properties of clause structure. The techniques used in Genie to address a variety of problems in rule government are presented along with examples.

1.1 Plan-and-Execute Design

Genie is based on **relational grammar (RG)**, which means, among other things, that sentences are represented in terms of explicitly marked relations/functions like subject, direct object, indirect object and that grammatical rules directly manipulate such relations (cf. Johnson 1974, Johnson and Postal 1980, Perlmutter and Postal 1974). Generation has two phases: planning and execution. The key idea is that restrictions/requirements on generation are determined during planning (e.g., passivization is required/prohibited), followed by the creation of the required constructions by a simple relational grammar (i.e., execution rules have very few conditions on applicability). The syntax planner, **Gramplan**, controls the execution grammar through a set of so-called **rule switches**, set in accordance with lexical, syntactic and stylistic requirements. Rule switches are simply features that name rules in the execution grammar. Each rule switch may be set either to **Yes** or to **No**, determining whether the named rule is to be tested for application.

Many rule-switch settings come from lexical entries (and so are called **lexical rule switches**). For example, the lexical entires for **resemble** and **want** will have the switch **(Passive = No)**, preventing the generation of ***He was resembled by her**, ***Toys are wanted by the children**, etc. The verbs **explain**, **confess**, **whisper**, etc. will have the switch **(Dative = No)** to prevent the generation of sentences like ***He explained/whispered/confessed her something**. Verbs like **try**, **want**, etc. will have the switch **(O-Equi = Yes)**, where **O-Equi** names the rule that deletes subjects of subordinate clauses that are coreferential to objects in the higher clause. This will insure that sentences such as ***I want/will try for me to go** will not be generated. Note that **I want for her to go** can be generated because **(O-Equi = Yes)** only triggers the *testing* of a rule, it does not mean that the rule *will apply (succeed)*. In contrast, verbs like **believe** will be marked **(O-Equi = No)** to prevent O-Equi from being tested and wrongly generating ***I believe to go** (cf. Lakoff (1970). There are many variant structures that are conveniently controlled through lexical switches (see Johnson 1988a,c for other examples).

Many switch settings are determined by **syntax planning rules**. For example, as discussed in Johnson (1988a), the verb **seem** has a disjunctive requirement: either a sentence of the form **It seem(s) that S** (the so-called **It-Extra** construction) or of the form **NP seems to VP** (the so-called **A-Raising** construction) must be generated (e.g., **It seems that they went to Tokyo, They seem to have gone to Tokyo** but not ***That they went to Tokyo seems**). The best choice depends on a variety of factors, e.g., the presence of modals in the subordinate clause. If there is a modal, then only the It-Extra construction is grammatical (**It seems that they can swim, *They seem to can swim**). On the other hand, certain predicates, e.g., **apt**, **tend**, etc. require the A-Raising construction. In this case, since modals are always finite, some action must be taken to express the modality without a modal. In the case of **can**, for instance, the syntax planner realizes the modal as the phrase **be able to**. So **He is apt to be able to swim fast** would be generated and not ***It is apt that he can swim fast** or ***He is apt to can swim fast**.

Looking at another example, Gramplan also has a planning rule that inspects clauses with direct objects to determine whether the direct object is clausal and hence must be assigned some other relation to produce grammatical or stylistically preferred output (cf. ***He taught how to ride a bike to her, He taught her how to ride a bike, He taught math to her, He taught her math** and ***He explained how to ride a bike to her, *He explained her how to ride a bike, He explained to her how to ride a bike, *He explained her that**). With verbs like **teach** and **tell**, which permit Dative, the planning rule simply sets **(Dative = Yes)**. On the other hand, verbs such as **explain**, **whisper**, **mention**, **say** prohibit Dative even with clausal direct objects (***He explained/whispered/mentioned/said her that he would leave early**). In such cases, the planning rule

for direct objects takes another action -- it sets the switch **(Null-Extra = Yes)**, causing the execution grammar to generate, e.g., **He said to her that he would leave by tomorrow** (cf. ***He said that he would leave by tomorrow to her**). (Note in passing that Null-Extra is inappropriate for **teach**: ***He taught to her how to ride a bike.**) The common result of these two distinct, relation-changing actions (Dative versus Null-Extra) is that the demoted clause is placed at the right-end of the clause by the linearization process. Although the argument that bears the canonical indirect object relation winds up with the same relative position in the clause (Verb Argument Clause), its superficial form will, as a result of the distinction in superficial relations (direct object versus indirect object), be different (noun phrase versus prepositional phrase). See Figure 1 on page 3.

A key point here is that *lexically* determined requirements (here, whether or not Dative is lexically permitted) can interact through rule switches with *syntactically* or *stylistically* determined conditions (here, clausal direct-object-hood) to determine an appropriate superficial form. Further, separating generation into a syntactic planning-phase and a subsequent execution-phase enables one to introduce context-dependent, sometimes heuristic, rules to deal with less well-understood grammatical/stylistic issues without losing the benefit of employing a simple execution-grammar.

1.2 Relational Execution Grammar

The relation changing rules of the execution grammar are classified as either **unmarked** or **marked**. With respect to a given node of the appropriate category, an **unmarked** rule R is *tested* for application *unless* the switch **(R = No)** has been set and a **marked** rule R is *tested* for application *only if* the switch **(R = Yes)** has been set. Examples of marked rules are Passive, Dative, A-Raising, B-Raising and C-Raising. Unmarked rules include those generating Yes/No auxiliary inversion, Wh-questions and relative clauses.

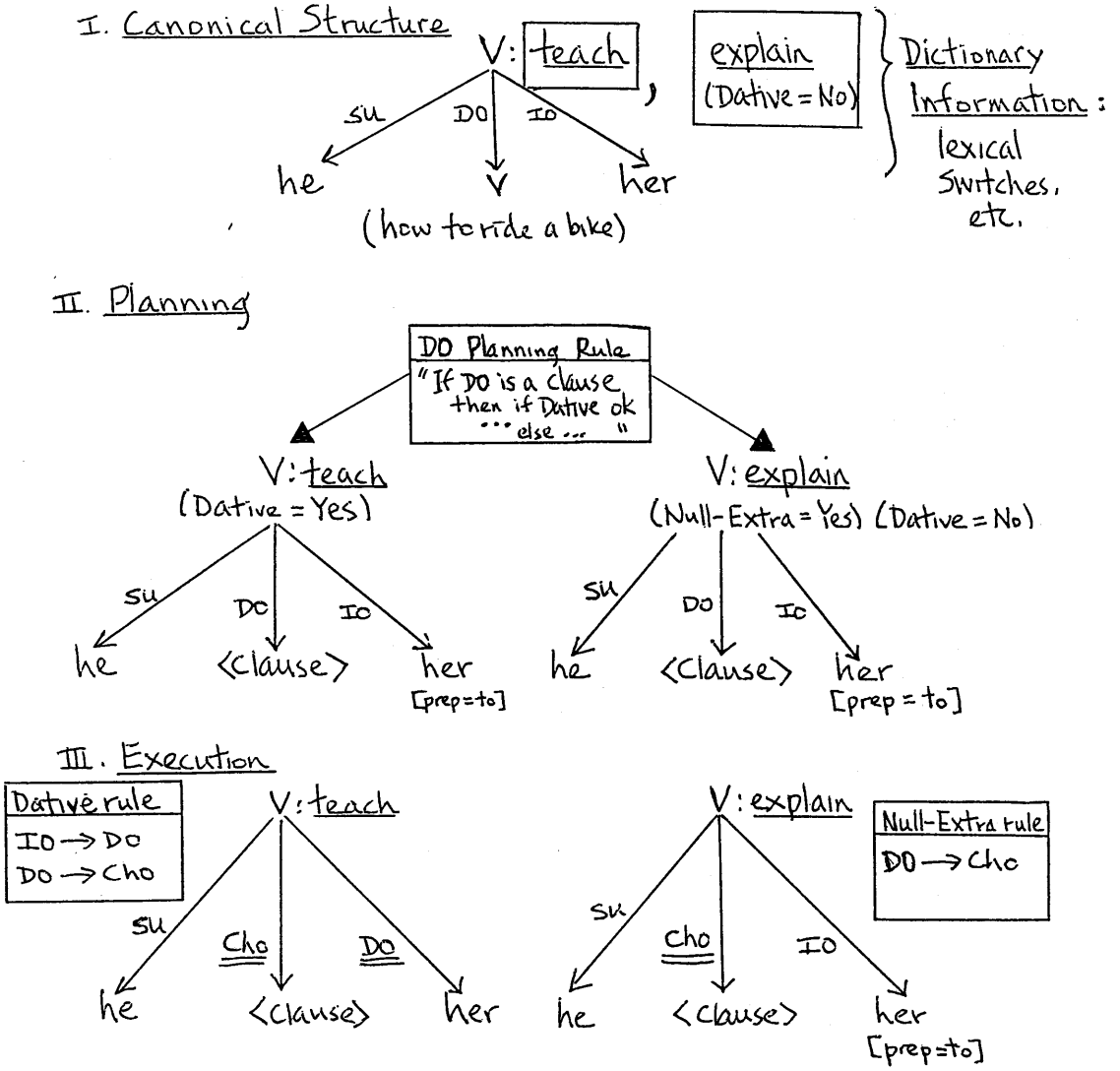
Following the earlier derivational models proposed in Postal and Perlmutter (1974) and Johnson (1974), the relational execution grammar is processed bottom-up with cyclic processing of rules like Passive, Dative, Subject-to-Subject Raising and Subject-to-Object Raising and post-cyclic processing of relational rules that generate Wh-Questions, Relative Clauses, etc. Note that *with the exception of linearization, all rules are relation-changing*; since the relational structures are unordered, there is no notion of a word order (linear precedence) altering rule. Application of the relation changing rules results in an *unordered*, surface relational structure.

Linearization is accomplished by a top-down, recursive pass over the superficial relational-structure. As an example, the linearization rule for verbs states that, given a verb head, V, output the sentence in the order: **Complement-Preposition, Complementizer, Subject, Verb, Sub-Direct-Object, Particle, Direct-Object, Indirect-Object, Direct-Object-Chomeur, Subject-Chomeur Locative, Temporal, Other-PP-Modifiers, Clausal-Complements.**

1.3 Category-Driven Generation and Rule Aliases

Genie has been implemented in **General**, an object-oriented shell for relational grammar-based linguistic processing developed to meet Genie's requirements by Peter Schindler (Schindler 1988). Besides being based on relational grammar (including a special language for writing relational-grammar rules), it is based on **category-driven processing**, makes available a facility for establishing a **part-of-speech category hierarchy with inheritance**, and provides mechanisms for **rule replacement** and **rule addition**, Schindler's novel notion of category-driven processing means that the only rules tested for application are those appropriate to the categories actually present in the input structure. That is, at generation time, for each node in the input structure, the grammar invoked for that node is constructed from the input relational structure, the dictionary entry for that node and the category hierarchy. For example, in the Genie category hierarchy, there is a rule of Passive attached to the transitive-verb node. As a result, all and only those input clauses with *transitive* verbs will result in Passive being tested for application.

Rule replacement has been used in Genie for cases where specific lexical items require special conditions or actions to be associated with a particular kind of relation-changing process. That is, a *rule name* is often used as an *alias* for any number of distinct rules that are in complementary distribution. For instance, consider passivization. Certain verbs, e.g., **look at**, **depend on** and **rely**



IV. Linearization

SU	Verb ...	DO	IO	Cho
he	taught	her		how to ride a bike
he	explained		to her	how to ride a bike

Figure 1. Planning/Execution Example

on, mark what I take to be a canonical direct object with a preposition. This flagging of the direct object is accomplished simply by adding to the verb's lexical entry the property of the form **(DOP . Prep)**, where **prep** is the required preposition. Based on this property, a planning rule will, e.g., flag the direct object of **depend** with **(PREP = on)**, resulting in, e.g., **They depend on him.** But now passivization is a problem. The *regular/default* Passive rule attached to the transitive verb node in the part-of-speech hierarchy would generate, e.g., ***On him can be depended.** Rather than either complicating the regular Passive rule or putting two (crucially ordered) rules of Passive in the grammar, one can simply specify *in the lexical entry* for a verb like **depend** that another rule **(Copy Passive)** replace Passive, i.e, be executed under the alias **Passive**. Copy-Passive leaves a lexically unspecified direct object noun behind to carry the preposition and deletes the preposition on the noun advanced to subjecthood, resulting in **He can be depended on.** Realization of the preposition and linearization are completely general.

Other cases involving rule replacement include linearization of adjectives and indefinite nouns like **someone**. Compare **someone tall** and ***tall someone.** The atypical linearization information **(noun adj)** resides with the small set of nouns that require this exceptional ordering. The rule replacement and inheritance mechanisms insure that these infrequent phrases will be treated properly *without complicating the normal linearization routine.* Two other, similar cases are: **how big a box** versus ***a how big box; too big a box (to use)** versus ***a too big box to use.** Rather than attempt to build one monolithic grammar with rules riddled with exceptional conditions, we have modularized the process in two distinct ways by (1) introducing a syntax planner, (2) using category driven generation.

1.4 Lexical Entries

Lexical entries in Genie consist of a key and six attributes:

1. **Form of Lexical Entries**

   Genie lexical entries consist of a key and six attributes:
   Key

   a. **Input Part-of-Speech (IPOS)**

   b. **Lexeme Feature Value (LEXFV)**

   c. **Alternative Part-of-Speech (APOS)**

   d. **List of Properties**

   e. **List of Rule Bundles**

   f. **List of Word-Specific Rules**

Lexical look up is done on the basis of both the **Key** and the **IPOS** information (so that homonyms can be distinguished). The attribute **LEXFV** was introduced to permit sense-to-word mapping (discussed below), and the **APOS** attribute was established to enable conversion from one part of speech to another. The latter might be used to map from an abstract category of some semantic representation to the English system. Information present in a lexical entry is added to the information residing on the input node with generation information taking precedence over information in the input structure. This means that in the context of MT, the generation dictionary need only contain "exceptional" information and that the absence of information will not generally result in termination of the generation process. The fourth attribute is a list of arbitrary property/value pairs containing standard grammatical information such as case, number, gender, mass/count, as well as the lexically determined rule-switch information discussed above (e.g., **(Dative . No)**, **(B-Raise . Yes)**), etc.

Skipping the fifth attribute for the moment, consider the the sixth, which contains a list of **word-specific rules**. The general notion **word specific rule** has some precedent in the literature and is a key feature of, e.g., the Mu machine translation system. However, the concept used in Genie is from Schindler (1988) and is closely tied to the notion of category-driven parsing. Genie word specific rules are relational rules that (i) can be either planning rules or execution rules and (ii) can

either be added to the grammar or replace a more general grammar rule, and are unique to a single predicate. Generation of idioms like **pull the wool over ... eyes** from the semantic predicate **DECEIVE** use a word-specific rule (see below).

The fifth attribute is a list of **rule bundles**, a concept unique to Genie/General. Each rule bundle is a named set of rules which typically function together and are required by some relatively small group of lexical items for proper processing. Rule bundles are one means for dealing with cross-classification. By specifying the name of a particular bundle in a lexical entry, at the time of lexical look up, the named rule bundle is activated and the rules in the bundle are used in processing the construction headed by the lexical entry. Additionally, rules can be shared by different bundles. Since there are many groups of words with rather special properties, this mechanism is particularly useful.

For example, the so-called **tough**-class of predicates, which are relatively few in number, have special properties and consist both of adjectives like **easy, tough, difficult** and nominals like **a snap a breeze, a bitch,** etc.. These predicates permit, in general, three structures - **Clause be Predicate, It be Predicate Clause** and **NP be Predicate to VP**, as in:
(2a) **Reading this book is difficult.**
(2b) **For me to read this book would be difficult.**
(2c) **It is difficult to read this book.**
(2d) **This book is difficult to read.**
Gramplan must decide which structure is to be generated by the execution grammar. For instance, the last pattern results from applying the rule of so-called **C-Raising** (Non-Subject-to Subject Raising). In general, the clausal subject structure is not preferred and if it contains a nominal requiring extraction, not even grammatical, as discussed by Ross (1967) and codifed in his **Sentential Subject Constraint**. Compare:
(3a) **For me to read this would be difficult.**
(3b) ***What would for me to read be difficult?** (from **For me to read what would be difficult?**)
(3c) **What would be difficult for me to read?**
In Genie there is a rule bundle that evaluates C-Raising predicates and determines which structure is to be generated. For example, when planning the structure of a C-Raising clause, if there is a non-subject **wh**-nominal, then the switch **(C-Raising = Yes)** is set, causing the execution grammar to generate (3c).

The same strategy of preferring raised over non-raised structures is employed quite generally. For example, there is a set of rules that evaluate predicates permitting **A-Raising** (Subject-to-Subject Raising) and, provided there is a **wh**-subject in the sentential subject, the switch **(A-Raise = Yes)** is set, rather than the default switch **(It-Extra = Yes)**, determining (4a) rather than (4b):
(4a) **Who seems to have gone to Tokyo?** (from ***That who went to Tokyo seems?**)
(4b) ***Who does it seem that went to Tokyo?**
and (5a) over both (5b) and (5c):
(5a) **Who is likely to go to Tokyo?** (from **For whom to go to Tokyo is likely?**)
(5b) ***Who is it likely that will go to Tokyo?**
(5c) ***Who is that will go to Tokyo likely?**
(Note in passing that if the complementizer **that** is deleted from (4b) and (5b), the results - **Who does it seem went to Tokyo?** and **Who is it likely will go to Tokyo?** -- are grammatical (but still, I think, awkward relative to (4a) and (5a)). However, this tact requires the formulation of either a **that**-Deletion rule or a more complicated **that**-Realization rule, both of which bring new non-trivial problems.) Since A-Raising and C-Raising are in complementary distribution, it would be senseless to (i) attempt to apply both sets of planning rules to both types of clauses and (ii) to attempt to execute the A-Raising rule in a C-Raising clause and vice versa. If the clause has an adjective like **apt** or **likely** as main predicate, then the C-Raising planning bundle will *not be invoked* and so the switch **(C-Raise = Yes)** will never be set. Since C-Raise is a marked rule, i.e., will only be tested for application if the switch **(C-Raise = Yes)** has been set, the C-Raise execution rule will *not even be tested for application* (mutatis mutandis, similar remarks hold for C-Raising predicates and A-Raising.)

Rule bundles have a number of useful properties. They allow one to capture the notion of complementary distribution of rules and to cross-classify categories *at any level of detail with respect to the rules invoked.* The lexical entries, however, are kept simple, containing merely the name of the rule bundle to be activated. Rule bundle cross-classification of traditional parts of speech is in terms of relational and process-oriented notions like "Clausal Subject", "Clausal Direct Object", "C-

Raising Construction", "A-Raising Construction", rather than structural notions like NP, S or more traditional sub-classes like "proper noun", "count noun", etc.. Finally, they can contain both planning rules and execution rules and are invoked *only if they are called by specific lexical items.*

1.5 Generating Idioms from Semantic Representations

The generator has been set up so that it can accept semantic-case representations as input (this feature is not used in JETS). Since Genie's planning and execution rules are formulated in terms of syntactic GRs, input case-roles must first be mapped onto this recognized set of relations. This mapping is accomplished by a combination of general (default) rules which determine canonical-level grammatical relations and *prepositional feature assignment.* These rules are not unlike those first suggested by Fillmore, except for the important difference that in Genie the assignment is basically a relabeling to a set of *canonical-level, syntactic relations*, while for Fillmore the mapping is onto superficial representations. The default rules are, informally, along the following lines: (1) if there is an agent, make it the subject, (2) if there is a theme and an agent, make the theme the direct object, (3) If there is a recipient, make it the indirect object and flag it with the preposition **to**, etc. Lexical rules are also required here. For instance, the recipient role is typically flagged with **to**, but in some cases the preposition is **on** as in **That dawned on me** and **He blamed that on her.** The default rules are sufficient to generate, e.g., the famous triple: **The window broke, A rock broke the window** and **He broke the window (with a rock).**

Genie's flexibility can be illustrated by looking at the generation of idiomatic expressions. Consider the problem of generating the idiomatic **John pulled the wool over Mary's eyes** from a semantic/case representation along the lines of **(DECEIVE (AGENT JOHN) (THEME MARY))**. In the idiomatic sentence **John pulled the wool over Mary's eyes, John** is the subject of **pull, over ... eyes** is a locative prepositional phrase, **Mary** bears the genitive relation to **eyes**, and **wool** is the direct object. This rather extensive restructuring is achievable by associating the sense **DECEIVE** with a lexical entry whose part-of-speech is *verb*, lexical string is **pull**, and which contains a idiom-specific rule that (1) creates a direct object **wool** and (2) takes the theme, **Mary**, into the genitive of a constructed locative noun **eyes** which has the feature **(Prep = over)**. Since this planning rule is specific to one idiom, it resides in the lexical entry itself, and hence would be called into play only where relevant. This example illustrates the use of lexically triggered rule-addition to achieve special processing.

Once the idiomatic structure is constructed, it is processed like any other structure. Since the case-to-GR rules are invoked at the beginning of the planning phase, they preceed the other planning rules, e.g., the one that determines whether certain clauses must be passivized. Since **pull** in the idiomatic phrase **pull the wool over x's eyes** is a verb syntactically, it will undergo the passivization planning rule just like any other transitive verb. Thus, **The wool was pulled over Mary's eyes** can also be generated. On the other hand, the characteristic defectiveness of idioms with respect to syntactic operations is often handled easily with the switch setting mechanism. For example, generation of the idiomatic phrase **kick the bucket** (in the sense of **DIE**) is straightforward. Roughly, one must associate with the sense **DIE** a lexical entry that specifies that it is a verb spelled **kick**, has the rule switch **(Passive = No)** and a planning rule whose execution by the planner creates a direct object **the bucket**. Since a normal *verb/direct object* structure would be built, **kick** would undergo tense-spelling **(kicked not kick the bucketed)**. However, in contrast to **pull the wool over x's eyes**, since the dictionary entry would also carry the switch **(Passive = No)**, passivization would be prevented.

1.6 Example: Verb-Particle Constructions

Verb-particle constructions provide another area of English grammar that involves numerous idiosyncratic constraints (cf. Fraser 1976). In Genie, creating a verb-particle construction, e.g., **look up**, is simple: one merely specifies the property **(PART . up)** in the lexical entry for the sense "look-up". A Particle planning rule will (1) create a **Particle** relation with **up** as the dependent and **look** as the head and (2) set the switch **(Dative = No)** to prevent generating sentences like ***He gave up her the toy/*he gave her up the toy**, and (3) will set the switch **(2-to-Sub-2 = Yes)** *if the direct object is pronominal.* Later, the execution rule **2-to-Sub-2** will demote the direct object (2) to sub-

direct-object (Sub-2) and linearization will place the Sub-2 next to the verb, insuring the order V NP Particle, e.g., **He threw it out** rather than ***He threw out it.** (cf. the statement for linearization of verb constituents above). Some constructions require that the particle be separated from the verb, e.g., **bring (someone) to** but not ***bring to someone.** This, of course, can be handled with a simple lexical switch **(2-to-Sub-2 = Yes)**.

However, even this collection of conditions is not sufficient to correctly generate verb-particle constructions. Consider, e.g., **put back**, which requires a direct object and optionally takes a locative phrase (**He put it back (on the shelf)**). (Note that **put** requires the locative: ***he put it.**) However, if the locative is *present*, the particle should, if only to avoid ambiguity, be separated from the verb (**??He put back the book on the shelf**). However, the separation is not called for if the locative is absent: **He put back the book.** Further, if the direct object is clausal, then the *non-* separated construction is preferred: **He put back whatever it was he took while I was out** versus **??He put whatever it was he took while I was out back.** This means one does not want to simply require **put back** to take the separated construction, which could be insured by a simple lexical switch. Rather, a better solution would be to add to the entry for **put back** a contextual planning rule that would examine the clause for a locative phrase and if found, would set the switch **(2-to-Sub-2 = Yes)**. Such planning rules pose no problems for Genie.

1.7 Final Remarks

The above examples illustrate the techniques used in Genie to handle problems of rule government, an important area for generation that seems to have been largely overlooked by the computational linguistics community. Genie's plan-and-execute design, in conjunction with the use of rule-switches, rule replacement and rule addition, make it easy to drive the generation process in very specific ways. On the one hand, exceptional behavior can be controlled by adding rule-switch specifications and/or exceptional rules to dictionary entries or through the use of more general planning rules. On the other hand, the execution grammar can be kept quite simple.

1.8 References

Fraser, B. 1976. **The Verb-Particle Combination in English** Academic Press.

Johnson, D. E. 1974. **Toward a Theory of Relationally-Based Grammar.** University of Illinois PhD Thesis. Published by Garland Publishers, Inc., New York, 1979.

Johnson, D. E. 1988a. "The Design of Post-Analysis in the JETS Japanese/English Machine Translation System", in the **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.

Johnson, D. E. 1988c. "Genie: A Transportable English Generator", IBM Tokyo Research Laboratory Report TR87-1023.

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

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

Perlmutter, D. M. and P. M. Postal. 1974. **Lectures on Relational Grammar.** LSA Linguistic Institute, University of Massachusetts, Amherst.

Ross, J. R. 1967. **Constraints on Variables in Syntax**, MIT doctoral dissertation

Schindler, Peter A. 1988. **General: An Object-Oriented System Shell for Relational Grammar-Based Natural Language Processing** M.S. Thesis, Department of Electrical Engineering and Computer Science, MIT.