[Mechanical Translation, vol. 8, No. 1, August 1964]

Connectability Calculations, Syntactic Functions, and Russian Syntax

by David G. Hays, Stagiaire qualifié, Common Research Center, EURATOM, Ispra*

A *program for sentence-structure determination is part of a system for linguistic computations such as machine translation or automatic documentation. The program can be divided into routines for analysis of word order and for testing the grammatical connectability of pairs of sentence members. The present paper describes a connectability-test routine that uses the technique called* code matching. *This technique requires elaborate descriptions of individual items, say the words in a dictionary, but it avoids the use of large tables or complicated programs for testing connectability. Development of the technique also leads to a certain clarification of the linguistic concepts of* function, exocentrism, *and* homography.

In the present paper, a format for the description of Russian forms and a program for testing the connectability of pairs of Russian items is presented. It recognizes nine functions: subjective; first, second, and third complementary; first, second, and third auxiliary; modifying; and predicative. The program is so far limited to these dominative *functions; another program, for the coordinative functions (coordination, apposition, etc.) remains to be written.*

1. Introduction

The subject of this paper is a certain kind of routine for testing the connectability of pairs of occurrences in text. A *connectability-test* (CT) routine is one part of a program for sentence-structure determination; the other part is a *parsing-logic* (PL) routine. Operating alternately, in a manner to be described in Sec. 1.1, these two routines identify syntactic relations among all the unit occurrences within a sentence. This is the second stage in syntactic recognition of text and follows dictionary lookup, in which the unit occurrences are identified. The kind of CT routine to be considered here has been called "code matching" in the literature; the general properties of this class of CT routines are introduced in Sec. 1.2. Special assumptions about the syntactic relations sought (Sec. 1.3) and the nature of the unit occurrences (Sec. 1.4) have to be introduced. The concepts of syntactic function, exocentrism and homography are discussed in Sec. 2, and a list of functions for Russian is proposed. The notational scheme and symbolic operations needed for realization of a code-matching CT routine in a computer are described in Sec. 3. Sections 4 and 5 apply the concepts of the previous sections to Russian; in Sec. 4 a format for encoding Russian syntactic properties is presented, and in Sec. 5 a CT routine for a part of Russian syntax is given. In Sec. 6, some programming problems involved in the storage and manipulation of large, numerous syntactic descriptions during sentence-structure determination are examined. Finally, in Sec. 7, the relationships between morphology and syntax are introduced as the proper subject of a much larger treatment.

1.1 SENTENCE STRUCTURE DETERMINATION

After dictionary lookup, a text is represented by a string of syntactic descriptions of unit occurrences. The purpose of sentence-structure determination is to establish syntactic relations over combinations of these occurrences. A PL routine[1] is a mechanism for selecting possible combinations; it uses only "word order", i.e. position in the string, as a characterization of each unit occurrence or previously established composite occurrence. Its logic is that of continuity in the general sense: the rule that constituents must be continuous, in phrase-structure theory, or the rule of projectivity, in dependency theory.[2] Besides position, a PL routine can be designed to use other properties of occurrences, but in that case it is specialized.[3] In its general form, the PL routine leads to the identification of *every* possible set of syntactic relations over occurrence spans of all lengths in the text. When one or more sets of syntactic relations bind together all occurrences within a span bounded by appropriate punctuation, the span is recognized as a sentence, unambiguous if it has a unique structure (set of relations binding all its occurrences), ambiguous otherwise.

When a PL routine selects a possible combination of occurrences, it transfers the combination, with descriptions of their syntactic properties, to a CT routine. This routine, using a concrete grammar of the language in which the text is written, determines whether the properties of the occurrences and the general rules of the

* On leave from The RAND Corporation, 1962-63. The work reported here was accomplished in part at RAND and completed at EURATOM.

grammar permit the combination. The CT routine returns a yes-or-no answer; or, if such concepts are used by the grammarian, a measure of the probability, value, or utility of the combination.[4] In its most general form, a CT routine is capable of supplying more than one positive answer for a single combination. Different dependency directions (cf. Sec. 1.3) or different functions (cf. Sec. 2) may have to be distinguished. As a byproduct of the connectability test, the CT routine furnishes, for every positive answer, a description of the syntactic properties of the new composite.

New composites are added to the list of occurrences available to the PL routine. Sentence-structure determination therefore consists of a sequence of selections by the PL routine, each followed by an application of the CT routine.

Both PL and CT routines can be designed in many ways, given the same linguistic theories and facts. The CT routine to be presented here is to be used with a general PL routine; the combination, given a grammar and text, will find every grammatically allowable structure for the text (but whether any of those structures is valid or intuitively acceptable depends on the content of the grammar). For use with a PL routine intended to produce the most "probable" structure of an input string, the CT routine would have to be modified, but only slightly, and in fact the designs of the two parts of a sentence-structure determination program are almost independent.

1.2. CODE MATCHING CT ROUTINES

The classic format for a grammar is a construction list. Each entry has three or more parts, naming the construction and each of its members. The connectability-test routine required is a table-lookup routine; the descriptions of two or more occurrences are looked up in the list, and if the combination is found the name of the construction it forms is found with it. This format is somewhat inconvenient in practice for two reasons. First, if the name of a construction is a concatenation of its syntactic properties, then it often resembles the name of one of its members (the governor). Space in the table is therefore wasted by repetition within each of many entries. Second, the linguist faces a dilemma. If just one symbol is assigned to each distinct unit, the number of rules is increased because many classes of units can participate in unique sets of constructions. If many symbols are attached to each distinct unit, the list can be greatly shortened, but the number of references to be made during sentence-structure determination is increased.

Code-matching CT routines as a class are distinguished by the fact that they require no list of constructions.[5] The syntactic description stored with each occurrence is in a format and notation that permits direct calculation of connectability and of the properties of the combination if one is permitted. In principle, the latter calculation can require the storage of considerable information that is not usable until the combination is formed. Code matching CT routines are related to the formal systems known as categorial grammars,[6] which are known to have essentially the same power as context-free phrase-structure grammars,[7] hence of dependency grammars.[8] In a categorial grammar, each syntactic description is a string of symbols containing one special mark. The string to the right of that mark is matched with the entire string characterizing a following unit, and the two units are connectable if and only if the two strings match exactly. In important papers on the subject, these strings are constructed with two primitive symbols (s = sentence, n = noun), parentheses, and the special mark. As a result of these restrictions, on the matching process and on the alphabet of symbols, the syntactic descriptions needed for natural language are formidable, and the number of different strings assigned to each distinct occurrence is large. Linguistically, it seems more convenient to use both a more elaborate matching process and an enlarged alphabet. In the Russian example given below, the size of each syntactic description is large but limited, not subject to indefinite growth, and most Russian items can apparently be characterized syntactically with a single description.

The principle used here is the isolation of *syntactic functions* and *agreement variables*. On the order of a dozen functions are proposed for Russian; every syntactic relation between a pair of occurrences in a Russian text is to be regarded as an instance of exactly one function. An occurrence is characterized by the functions it can enter and by values of the agreement variables. Each function entails agreement with respect to certain variables. The CT routine therefore seeks a function common to a pair of items and then tests their agreement with respect to the variables material to that function. (In this paper, *material* will be used in this sense; a variable is material to a function if the function entails agreement with respect to it.)

1.3. DEPENDENCY AND PROJECTIVITY

The theory of categorial grammars imposes an asymmetry on every construction. Let / be the special mark, and let s/n be the description of a transitive verb. Then when a noun (description n) follows a transitive verb, the matching operation (symbolized by a dot) gives s/n.n = s. Part of the symbol of the verb remains, whereas the symbol of the noun has entirely disappeared. In general, a code-matching system can be devised to retain parts of both symbols, but a rule of *pars major* can be invoked to maintain the asymmetry. Moreover, the special mark can be regarded as dividing each grammatical symbol into a part to be matched with a dependent and a part to be matched with a governor. Thus the articulation of dependency theory with code matching is natural. In particular, any function

must be regarded as asymmetrical, *served by* one occurrence, *governed by* another, even if phrase structure theory is adopted.

The theory of dependency will be assumed here, and with it the continuity rule of projectivity. The PL routine is therefore supposed to furnish combinations of occurrences consisting of adjacent unit occurrences or adjacent composites whose heads (principal members, from which all other unit occurrences depend directly or indirectly) are to be joined directly by dependency. If the heads of two composites (or two unit occurrences, or a unit occurrence and a composite) are identified as X and Y, the CT routine tests whether X can depend on Y and gives a yes-or-no answer; it also tests whether Y can depend on X, and gives a separate answer to that question.

1.4. UNIT OCCURRENCES

It is assumed here that the units identified during dictionary lookup are *forms,* simultaneously the largest units constructable by morphological rules and the smallest units to which syntactic descriptions can be assigned. This separation of morphology and syntax is justified, linguistically, on three grounds; the argument applies to Russian and presumably to certain other languages, but certainly not to all natural languages. First, the categories and construction rules of Russian morphology and syntax are separable with virtually no overlap (i.e., morphological rules are exocentric). Note here that the categories needed in morphological rules and the categories established by morphological properties are not necessarily identical; many syntactic properties of Russian forms are established by their morphological constitution. Second, an absolutely strict size-level distinction can be made between morphology and syntax, so that dictionary lookup of forms can be completed before sentence structure determination, using only syntactic rules, begins. Third, the continuity rules for morphological and syntactic constructions are somewhat different and much simpler if separated. Specifically, the continuity rule for morphological constructions is that the immediate constituents of each construction are continuous (with some notable exceptions), whereas the rule for syntax is projectivity. Projectivity does not seem to hold in Russian if the syntactic unit is taken to be the morph or morpheme. Incidentally, forms are bounded by spaces or marks of punctuation in printed Russian text and only a limited number of forms or morphological construction types contain either spaces or marks of punctuation. Those containing spaces are strictly limited, and those containing spaces are strictly limited, and those containing punctuation—mainly the hyphen—are of limited types although not limited in number. The same is true of many other printed languages. Another separation satisfying these three criteria (separability of rules, separability by size level, and simplification of continuity rules), or even the first two, does not appear to exist in Russian but might well appear in English, for example.

2. Functions

The code matching plan to be described here can be used with any set of functions, or varieties of grammatical relationships. Let us assume that the functions of a language have been determined; then each unit, elementary or composite, is characterized by two lists of functions: those it can govern and those it can serve as dependent. A description of the structure of a sentence will specify, for each elementary unit, what function it serves in the sentence and what occurrence governs it. For example, in "John ate breakfast" the unit occurrences are "John," "ate," and "breakfast." Here "John" serves subjective function, governed by "ate;" "breakfast" serves objective or complementary function, also governed by "ate;" and "ate" itself serves predicative function, with no governor.

The functions of a language can be classified as optional or singular. An optional function is one that can be served by any number of dependents of a given occurrence; for example, the function of adjectival modifiers of nouns in English may be optional. A singular function is one that can be served by at most one dependent of a given occurrence, such as the subjective function in various languages. (If two conjoined nouns, or two nouns in apposition, serve as subject of a Russian or English verb, the function is nevertheless served only once, by the conjoint or apposite group.) A singular function is said to be obligatory if it must be served by a dependent of every occurrence of a given unit.

Ignorance of empirical fact could lead an investigator to classify two singular functions together as one optional function. This error is corrigible, however, since an occurrence capable of governing both of the singular functions can govern only one dependent with each of them, a fact that can be revealed by study of texts and interrogation of informants. The differentiation of adjective order classes in English, for example, may lead to identification of several singular adjectival functions in place of the optional function now hypothesized. Any two singular functions can be reduced to one if no occurrence in the language is capable of governing both, but cannot be if some occurrences govern one dependent with each function. On the other hand, all of the optional functions of a language can be taken as a single function, since—by definition—governing one dependent with an optional function does not prevent an occurrence from governing others.[9]

Statements about functions governed and functions served determine the major form classes of a language. These necessarily supersede all other part-of-speech classes, which would be irrelevant for syntactic operations. A syntactic unit, elementary or composite, is primarily characterized by three lists of functions: those it can govern, those it must govern, and those it can serve

as dependent. This set of three lists is called the *function triple* of the item. A major form class consists of all forms bearing identical function triples. Within a form class, the agreement variables that are material for any of the functions mentioned differentiate the class members.

Agreement variables are *material* for a function if two units connected with that function agree with respect to that variable. The notion of agreement to be understood here is very broad; it covers the agreement of Russian adjectives with the nouns they modify, and also the agreement between a verb that requires an accusative object and the accusative noun depending on it. The agreement requirements of a function are *homogeneous* if the same agreement variables are material for every combination of units connected with the function. If the modifying function in Russian is a single, optional function, its requirements are heterogeneous, but it can be analyzed into two subfunctions with homogeneous requirements: adjectival and adverbial. The complementary functions in Russian are heterogeneous; many Russian forms can govern as complement either a noun or a noun clause, with different agreement requirements in the two situations (the noun must be in a certain case, the noun clause must be introduced by a certain conjunction). On the other hand, if a unit can *serve* complementary function, the material variables are always the same for it; hence minor form classes can be identified.

Under certain circumstances it is necessary to assign two or more function triples to a single unit which therefore belongs to two or more major form classes and can be called *homographic.* Let $F_1, F_2, \ldots, F_n$ denote the functions of a language. There are four cases.

(i) If unit X can serve some F_i only in occurrences in which it also governs some F_j, and if F_j is not obligatory for X, then X is homographic. For example, finite forms of the Russian *byt′* = *be* can serve predicative function, but only if they govern complements. Otherwise they serve only auxiliary functions, and do not govern complements. One function triple allows X to serve F_i and makes F_j obligatory; another does not allow X to serve F_i and either omits government of F_j or makes it singular.

(ii) If X can govern F_i only if it simultaneously governs F_j, then X is homographic. Its two function triples are similar to those described under (i), *mutatis mutandis.* Any Russian infinitive can be regarded as homographic for this reason; it can govern a subject only if it governs an auxiliary. (But this can be taken as an example of exocentrism; see below.)

(iii) If X cannot govern F_i and F_j simultaneously, even though in general they can be governed together, then X is homographic. (If the two kinds of dependents could not be governed together by any unit in the language, they would be identified as the same function.)

(iv) If the value for X of some agreement variable material to function F_i, which X can serve or govern, varies according to the nature of the dependent that serves function F, for X, then X is homographic; likewise, of course, if the mere presence of a dependent with function F_j is influential. For example, the presence of a negative modifier as dependent of an ordinary transitive verb influences the properties of the direct object permitted in Russian. With the negative modifier, a verb that normally governs the accusative can instead govern the genitive.

As a rule, the functions of a governor are not modified by the attachment of a dependent; when modification takes place, we can speak of *exocentrism.* Exocentrism and homography are to some degree interchangeable. Economy helps to determine which facets of linguistic structure will be handled by one device, which by the other. Consider case (i), as described in connection with homography. Since, in a projective language, it is always possible to attach all dependents to a unit before attaching the unit to its governor, the conditioning dependent can always be attached before the conditioned. Case (ii) is different, since projectivity does not guarantee that the conditioning dependent is attached first; that depends on the grammar of each language individually. If the class of units that can serve the conditioning function is small, and the class of homographic units would be large—as with infinitives (a large conditioned class) and auxiliaries (a small conditioning class)—it is more economical to mark the conditioning units and revise the function triple of the governor when the dependent is attached, provided that the order of attachment can always put the alteration ahead of the pertinent test.

Functions can be classified as coordinative and dominative. The agreement requirements of coordinative functions are symmetric in the sense that the same agreement variables are tested for both members of the pair of associated units. In general, two units can be coordinated if there is some function that the two can serve jointly, but the details are complicated and cannot yet be discussed clearly. Dominative functions are all the others. In Russian, there appear to be at least two coordinative functions, conjunction and apposition, with more than one kind of conjunction possible. The rest of this section treats the dominative functions of Russian.[10]

The dominative functions currently hypothesized for Russian are subjective, complementary (three functions), auxiliary (three functions), modifying and predicative. The following illustrations are archetypal:
Subjective function: nominative noun depending on finite verb.
First complementary function: accusative noun depending on finite verb, *or* genitive noun depending on noun.
Second complementary function: dative noun depending on verb.
Third complementary function: prepositional phrase depending on verb.

First auxiliary function: Finite verb (small category) depending on infinitive verb, *or* finite form of *byt'* depending on short-form adjective.
Second auxiliary function: Negative particle *ne* depending on verb.
Third auxiliary function: Comparative marker depending on adjective.
Modifying function: Adjective depending on noun, *or* adverb depending on verb.
Predicative function: Finite verb depending on relative adverb.
The subjective, complementary, auxiliary, and predicative functions are singular. For the present, the modifying function is optional, and it remains to be seen whether an economical classification of modifiers would lead to a set of singular or obligatory functions to replace this one.

3. Design of a Code Matching CT System

To simplify the exposition of the agreement variables, the general plan of the CT system in which they are to serve is presented first. According to this plan, a grammar-code symbol is assigned to each form in the dictionary and attached to each form occurrence in text during dictionary lookup. Each symbol consists of a string of binary digits (1's and 0's) of fixed length. The nth digit has a certain linguistic significance, and the format of the grammar code symbols is a statement, for each position, of its significance. Each position represents one value of a variable with respect to some operation in the CT routine. For example, if grammatical case is a variable, a noun can be characterized with respect to case in more than one way: its own case, as determined by its ending; the case it governs (usually genitive); and so on. A set of positions representing all the values of one variable will be called a *frame,* A frame, filled with digits characterizing a form with respect to a definite operation, occupies a certain set of positions in the grammar-code symbol, and that set of positions will be called a *segment.* One frame is needed for the set of syntactic functions named above. It has nine positions, for which abbreviations will be used: subjective (s), first complementary (c^1), second complementary (c^2), third complementary (c^3), first, second, and third auxiliary (x^1, x^2, and x^3, respectively), modifying (m), and predicative (p). This frame applies to three segments of the grammar-code symbol: functions governed (F_g), functions served as dependent (F_d), and functions governed obligatorily (F_o). To refer to a segment of the grammar-code symbol of an occurrence, we will use the name of the segment and the location of the occurrence. Thus $F_g(A)$ is the functions-governed segment of the symbol attached to the occurrence at location A in a text. When it is necessary to refer to a single binary position in the grammar-code format, we will use abbreviations for variable values as superscripts: F^s_g, for example, refers to the subjective-function position of the functions-governed segment, and F^{x3}_o to the third-auxiliary position of the obligatory-functions segment.

The first step in the comparison of two grammar-code symbols is to determine whether there is any function that one can serve for the other. Call the two occurrences D and G, and assume that the test is restricted to determining whether occurrence D can serve any function for occurrence G. If F^i_g (G) = 1, then occurrence G can govern a dependent with function *i* (here *i* stands for any function). Likewise, if $F^i_d(D) = 1$, occurrence D can serve function *i.* If there is some function *i* for which $F^i_d(D) = F^i_g(G) = 1$, then occurrence D can serve function *i* for occurrence G, provided that the agreement requirements of function *i* are satisfied. The Boolean product, $F_g(G)$ & $F_d(D) = F$, is constructed by setting $F^i = 1$ if $F_g(G) = F_d(D) = 1$, and writing $F^i = 0$ otherwise. This product can be obtained easily and very quickly by most modern computers, for long strings of 1's and 0's.

Boolean products, also called logical products, will be used throughout this CT routine. In several instances below, it is sufficient to characterize the product as equal to zero or not. If the product F defined above equals zero, occurrences G and D cannot be connected with G as governor; otherwise, their connection is subject to further tests. For functions, and also in several instances below, it is necessary to determine the locations of all 1's in the product. Thus, for functions, each function has its own agreement requirements, and the further tests to be performed follow those requirements. The exact form of the *junctions test* is:

Test $F_g(G)$ & $F_d(D) = F$.
If F = 0, stop.
Otherwise, if $F^i = 1$, test agreement with respect to function *i.*

The tests for the separate functions will be described below. This statement of the test can be encoded for operation on a computer, given the length of F and the fact that F can contain any combination of 1's and 0's.

Another operation, the Boolean or logical sum, will be needed. The sum of X and Y, X v Y = Z, is defined by: $Z^1 = 1$ if $X^1 = 1$ or $Y^1 = 1$, and $Z^1 = 0$ otherwise. Thus $Z^1 = 0$ if $X^1 = Y^1 = 0$. The sum of two segments therefore marks the properties possessed by either of two items.

4. Grammar-Code Symbol Format

The format used here for Russian grammar-code symbols has 38 segments using 11 frames. One frame, for syntactic functions, has been described. The others are substantive type (T), nominal properties (N), clause type (K), prepositional phrase type (H), first auxiliary type (X_1), modifier type (M), preceding adverbial type (D_1), following adverbial type (D_2), location (L), global type (G), and global nominal properties (J).

4.1. SUBSTANTIVE TYPE

Four syntactic functions are served by substantives (the subjective and complementary functions). The units that can serve these functions are diverse, and any governor of a substantive function imposes certain limits on the variety of units that it accepts. Classifying these units according to further agreement requirements, they are nominals (n), infinitives (i), clauses (k), prepositional phrases (h), and adjectivals (a).

Nominals are nouns (morphologically defined) and items that can replace nouns in all contexts: substantivized adjectives, pronouns, relative pronouns, cardinal numbers, etc. These units must satisfy agreement requirements with respect to case, number, gender, person, and animation—the *nominal properties* described in Sec. 4.3.

Infinitives, syntactically, are the same items as morphologically.

Clauses are sentences marked by conjunctions, relative pronouns, or relative adverbs and capable of serving substantive functions. Of course, not every Russian clause is substantival.

Prepositional phrases consist of prepositions with their complements and occurrences that derive from the complements, but only those that serve complementary functions are marked in the substantive-type frame.

Adjectivals are long-form instrumental adjectives, a few genitive nouns, and certain other items that replace long-form instrumental adjectives in copula sentences.

The grammar-code symbol of a form includes five segments to which this frame applies. One describes the unit coded (T_d), one indicates the type of subject governed by the unit coded (T_{gs}), and three describe the types of complements governed by the unit coded, one each for first, second, and third complements (T_{gc1}, T_{gc2}, T_{gc3}).

When the connectability of two items is tested, if the functions test shows that occurrence D can serve a substantive function (say first complementary) and that occurrence G can govern it, the substantive types of G and D are compared: $T_{gcl}(G)$ & $T_d(D)$. It follows that if $F^{c1}{}_g = 0$ for some item, then the content of T_{gc1} for that item is linguistically immaterial, and can have no influence on any connectability test involving the item.

Similar statements can be made about all other segments of the grammar-code symbol; each is material for an item only if definite preconditions are satisfied.

The segments indicating type of substantive governed can contain any possible pattern of 1's and 0's, since, for example, a verb may exist that governs, as second complement, any subset of the set of substantive types. On the other hand, T_d never contains more than a single 1; no Russian item is ambiguously either a prepositional phrase or a subordinate clause. Hence the product of T_d with one of the T_g's never contains more than a single 1. In this the substantive-type test differs from the functions test, and the difference is large from the programming viewpoint.

4.2. CLAUSE TYPE

Several types of Russian substantive clauses must be differentiated because they can serve particular functions for different classes of governors. That is to say, the class of verbs that can govern *chto*-clauses is not identical with the class that can govern *chtoby*-clauses in, for example, the first complementary function. The categories necessary for this purpose have not been established, but it appears that *chto, chtoby, li,* and other introductory words mark syntactically distinct categories of clauses, and will apply to five segments: $\mathbf{K}_d$, $\mathbf{K}_{gs}$, $\mathbf{K}_{gc1}$, $\mathbf{K}_{gc2}$, and $\mathbf{K}_{gc3}$, indicating, respectively, type as dependent, type of subject governed, and type of first, second, and third complement governed. Much of what was said about substantive type, *mutatis mutandis,* can also be said about clause type.

4.3. NOMINAL PROPERTIES

The variables ordinarily discussed in Russian grammars as characterizing Russian nominals are person, number, gender, case, and animation. The subject of a Russian verb ordinarily must agree with respect to all of these except animation (and Harper[11] shows that verbs taking animate and inanimate subjects can be differentiated.) The complement of a verb, noun, or preposition must be in a certain case, or possibly in one of a few selected cases. A noun and any adjective modifying it must agree in number, gender, case, and animation.

The patterns of ambiguity generated by Russian morphology make these variables interdependent. Thus case and number are tied together by such forms as *linii,* which is genitive singular, nominative plural, or accusative plural. This form cannot be characterized simply as nominative, genitive, or accusative, as singular or plural, since that would imply that it can be genitive plural. Either two separate descriptions—two grammar-code symbols—must be assigned to the item or case and number must be combined and treated as a syntactic variable with twelve values, three true for the example. The latter course is preferable, because it accelerates sentence-structure determination with only a small increase in storage requirements (or even, perhaps, with a saving). All five nominal properties are interdependent in this sense.

Taking the simplest view, the complex nominal properties variable would have 216 values. For number has two values, gender three, case six, person three, and animation two: 2 x 3 x 6 x 3 x 2 = 216. Note, however, that gender is neutralized in the plural, that

person (material only for the subjective function) is neutralized except in the nominative case, and that animation (disregarding Harper's finding for the moment) is material only in the accusative case. Combining number and gender into a variable with four values —masculine (m), feminine (f), neuter (n), and plural (p)—and combining case, person, and animation into a variable with nine values—nominative first person (n_1), nominative second (n_2), nominative third (n_3), genitive (g), dative (d), accusative animate (a_a), accusative inanimate (a_n), instrumental (i), and prepositional (p), the complex variable has 36 values: masculine nominative first person (mn_1) masculine nominative second person (mn_2), and so on, through plural prepositional (pp). The fact that nominal properties can be represented with a 36-valued variable is obviously related to the fact that certain computers use a 36-position storage cell. If larger cells were available, the nominative third person could well be differentiated into animate and inanimate, adding four values to the complex variable.

The nominal properties frame N, with 36 positions, applies to five segments of the grammar-code symbol: N_d, N_{gs}, N_{gc1}, N_{gc2}, and N_{gc3}, for description of the item itself, of the subject governed by the item, and of the first, second, and third complements governed by the item. These segments are used in tests for subjective and complementary functions if the dependent is a nominal-type substantive. In the test of modifying function, if the modifier is adjectival type, $N_d(G)$ & $N_d(D)$ is examined; this is the outstanding exception to the rule that *different* segments of the grammar-code symbols of governor and dependent are involved in each connectability test.

4.4. PREPOSITIONAL PHRASE TYPE

When a complementary dependent is found to be a prepositional phrase (as a result of the substantive type test), it is necessary to determine whether it is the kind of phrase acceptable to the potential governor. The syntactic categories of prepositional phrases that can serve complementary functions (in other words, be strongly governed) can presently be described only by naming the preposition and the case of its complement. The list that follows is given[12] by Iordanskaya; *chem* has been added:

> *v* (a), *v* (p), *dlya* (g), *do* (g), *za* (a), *za* (i), *iz* (g), *k* (d), *mezhdu* (i), *na* (a), *na* (p), *nad* (i), *o* (a), *o* (p), *ot* (g), *pered* (i), *po* (d), *pod* (a), *pod* (i), *pri* (p), *protiv* (g), *s* (g), *s* (i), *u* (g), *chem* (n), *cherez* (a).

The prepositional phrase type frame has, for the present, 26 positions. It applies to four segments: H_d, H_{gc1}, H_{gc2}, and H_{gc3}. Prepositional phrases never serve subjective function in Russian.

4.5. FIRST AUXILIARY TYPE

The first auxiliary function is served by modal and tensal dependents of infinitives, short-form adjectives and particles, and syntactically equivalent forms. Two types of auxiliaries must be distinguished: those that depend on infinitives are called *finitive* (f), those that depend on short-form adjectives are *tensal* (t). Finitive auxiliaries form verb phrases; with the auxiliary, the infinitive can govern a subject. Tensal auxiliaries mark tense and sometimes restrict the person of the subject governed. The nonpast-tense forms of *byt'* are marked for both types. The first auxiliary type frame X_1 has just two positions and applies to two segments: X_{1g} for type of first auxiliary governed, X_{1d} to describe the item itself as dependent.

4.6. MODIFIER TYPE

Two kinds of agreement requirements must be differentiated for modifying dependents. If the requirements concern nominal properties, the dependent is adjectival (a); otherwise it is adverbial (d). The modifier type frame has two positions and applies to two segments: type of modifier governed, M_g, and type of modifier as dependent M_d.

4.7. ADVERBIAL TYPE

The classification of adverbs is perennially difficult, and little can be said for the moment about the agreement of adverbial modifiers with their governors in Russian. It is proposed to establish two frames, one for modifiers that precede their governors and one for those that follow (D_1 and D_2 respectively), and to assign positions as syntactic categories are discovered. Each frame will apply to two segments of the grammar code symbol, D_{1g} and D_{2g}, to describe the adverbs governable by the item, and to two others, D_{1d} and D_{2d}, to describe the syntactic categories of the item itself as adverbial modifier.

4.8. LOCATION

A frame with two positions is used to specify the relative location in text of governor and dependent. The first position is for dependent before governor, the second for governor before dependent. The frame is denoted L, its positions L^1 and L^2. In grammar code symbols, this frame indicates restrictions on order. If a governor can have either a preceding or a following dependent, 1's appear in both positions, but if the governor must follow, there is a 1 in the first position only. The frame applies to six segments in the grammar code symbol: L_{gs}, L_{gc1}, L_{gc2}, L_{gc3}, L_{da}, and L_{dx}. The first refers to the subject governed by the coded item, the second, third, and fourth to the complements it governs, the fifth to its own location as adjectival de-

pendent, and the last to its own location as auxiliary. The frame also applies to a segment not in the dictionary but constructed when two occurrences are to be tested for connectability. This segment, always containing a single 1, indicates whether the occurrence being considered as potential governor lies before or after the other. It is denoted L_t.

4.9. GLOBAL PROPERTIES

Global properties are those that belong to any phrase, up to a certain syntactic type, that contains an item bearing the property. For the present, two such properties are known. The word *li* anywhere in a sentence makes the whole sentence interrogative; a sentence containing *li* can serve as a subordinate clause with substantive function. The word *kotoryj* anywhere in a sentence marks it as an adjectival subordinate clause. The two positions of G, the global properties frame, are denoted G^l (li-clause) and G^a (adjectival clause). Only one segment is needed for global properties, showing the global properties of the entire construction headed by the occurrence coded. In the dictionary, G is blank for every form except *li* and the forms of *kotoryj*.

4.10. GLOBAL NOMINAL PROPERTIES

A Russian adjectival clause must agree with the noun it modifies with respect to only two variables: gender and number. Some forms of *kotoryj* are ambiguous with respect to these variables, and since these variables are interdependent with case, the ambiguity can sometimes be resolved when *kotoryj* is attached to a governor in the subordinate clause. The global nominal properties of a subordinate clause, or of any construction within a subordinate clause that contains *kotoryj* are the gender and number of the antecedent expected. The frame has four positions (masculine, feminine, neuter, and plural) and applies to one segment, J, which is always blank in the dictionary and filled out when the governor of *kotoryj* is found.

5. The Connectability Test Routine

From a strictly formal point of view, it is possible to construct an algorithm for testing connectability in any language with context-free phrase-structure grammar. The simplest version of the algorithm supposes that each grammar code symbol is divided into two parts, one showing what "functions" the item can govern, the other what "functions" it can serve as dependent. To test a pair of items, the algorithm merely matches the government code symbol of one with the dependency code symbol of the other. Even with isolation of syntactic functions and agreement variables, as proposed here, a universal algorithm is possible. It would require, for each language, a reference table entered with the name of a function and containing an indication of the segments of the two grammar code symbols to be matched. One line of the table, for Russian, would be

$$F^{x1} : X_{lg}\,(G)\ \&\ X_{ld}\,(D)$$

where the left-hand symbol, denoting a function, labels the entry, and the right-hand part, the entry proper, shows what parts of the grammar code symbols are to be tested. The tests for several Russian functions are more complex, however. Given the modifying function, the first step is to test type; then, if adjectival, to test nominal properties and, if adverbial, to test adverbial type. Such processes can be described in table entries, but they are more readily presented in the form of a program. Since the universal program is absolutely trivial, the only complexity is in the concrete detail of a particular grammar, and it seems convenient to surrender universality for the sake of having a more powerful tool for the description of individual grammars.

The general form of the routine is universal. First, there is a test for possible functions. For each possible function, there is a subroutine. If the agreement requirements for the function are homogeneous, the material segments are tested by taking a logical product which is zero or nonzero. If zero, the items cannot be connected with that function; if nonzero, they can be. If the agreement requirements for the function are heterogeneous, a test to determine type of agreement requirement intervenes and can give one of several answers: no connection possible, or else a certain type of agreement to be tested, implying certain segments as before. In principle, a sequence of type, subtype, subsubtype, etc., tests could be required before specification of agreement variables, but the sequences found in Russian are short. Besides tests of segments of grammar code symbols, tests of relative location are included in the present routine, and tests of punctuation could be added.

Before the CT routine is applied to a pair of occurrences, the parsing logic routine has selected them in accordance with its design and their place in the sentence, has designated one of them as potential governor, and has produced $L_t(GD)$, a location segment showing whether the governor or dependent lies ahead of the other in test. The steps in the routine are named for convenience and numbered for reference.

1. *Function selector*

Test $F_g(G)\ \&F_d(D) = F.$

If F = 0, stop.
If F^s = 1, test subjective function (2).
If F^{c1} = 1, test *i*-th complementary function (3').
If F^{x1} = 1, test first auxiliary function (4).
If F^{x2} = 1, test second auxiliary function (5).
If F^{x3} = 1, test third auxiliary function (6).
If F^m = 1, test modifier function (7).
If F^p = 1, test predicative function (8).

The test produces the logical product of $F_g(G)$ and $F_d(D)$ and examines it. If all positions are zero, the routine is stopped and the PL routine seeks another pair; this is the meaning of "stop" throughout the CT routine. Otherwise, all of the nonzero positions are noted and for each some operation is performed. These operations cannot be performed in parallel, but it is best to imagine them as simultaneous. Each uses the grammar-code symbols supplied for occurrences D and G by the parsing-logic routine and each does or does not produce an output independently of all the others. When one of these routines produces an output, it alters certain portions of the grammar-code symbol of G, but these alterations do not affect either the original symbol on which the other routines are working or the symbols that they will produce as output. It would be possible, in principle, for the CT routine to yield nine separate outputs, and it will not be rare for it to produce two.

2. *Subjective function*

Test $L_{gs}(G)$ & $L_t(GD) = L$.

If $L = 0$, stop.
If $L \neq 0$, test subjective substantive type (2.1).

This test controls relative location of G and D. In a nominal sentence, where the predicate is headed by a noun in the nominative case, either the first nominative noun in the sentence or the second could be regarded as the subject. If $L_{gs} = 10$ for every noun that can govern a subject, the first will always be taken as subject, eliminating an ambiguity that seems universal and pointless.

2.1. *Subjective substantive type*

Test $T_{gs}(G)$ & $T_d(D) = T$.

If $T = 0$, stop.
If $T^n = 1$, test subjective nominal properties (2.2).
If $T^k = 1$, test subjective substantive clause type (2.3).
If $T^l = 1$, prepare output for subjective function (2.5).

The subject of a Russian sentence is a nominal, a clause, or an infinitive. Since T_d contains at most a single 1, this test leads either to a stop or to exactly one branch. If the possible subject being tested is nominal or an infinitive, further tests must be performed, but no further agreement requirements are known for infinitive subjects.

2.2. *Subjective nominal properties*

Test $N_{gs}(G)$ & $N_d(D) = N$.

If $N = 0$, stop.
If $N \neq 0$, replace $N_{gs}(G)$ with N and prepare output for subjective function (2.5).

There may be several 1's in N, but they have no functional significance. The remaining ambiguity in the nominal properties of the subject are irresoluble syntactically, since the subject already has all of its own dependents. The nominal properties of the subject, were their ambiguities resolved one way or another, would not influence the connectability of any other occurrence with the governor of the subject. Hence it is not necessary to produce multiple outputs, one for each possible resolution of the ambiguities remaining. (In this the agreement variables contrast with syntactic functions.)

2.3. *Subjective substantive-clause type*

Test $K_{gs}(G)$ & $K_d(D) = K$.

If $K = 0$, stop.
If $K \neq 0$, test clause-subject location (2.4).

This test determines whether the substantive clause proposed as subject is of a type that can be accepted by the proposed governor. Remaining ambiguity is immaterial, hence there is no branching on type of clause. If it should prove to be the case, however, that different types of clauses have different location rules, then a branching would be necessary.

2.4. *Clause-subject location*

Test $L_{ds}(D)$ & $L_t(GD) = L$.

If $L = 0$, stop.
If $L \neq 0$, replace $K_{gs}(G)$ with K, from (2.3), and prepare output for subjective function (2.5).

In 2 above, a test for location requirements of the governor was made. Here the location requirements of the dependent are examined.

2.5. *Output for subjective function*

Set $F^s_g(G) = 0$.
$F_d(G) = 000\ 000\ 001$.
$T_{gs}(G) = T$
$D_{lg}(G) = D_{lg}(G) \vee D_{lg}(Pred)$.
$D_{2g}(G) = D_{2g}(G) \vee D_{2g}(Pred)$.
Do global properties routine (9).

The governor, since it has a subject, cannot have another; the function is singular. The governor, since it has a subject, cannot serve any function but the predicative. Altering $T_{gs}(G)$ here completes the marking of G to show exactly what type of subject it governs; if the subject is nominal, $N_{gs}(G)$ was altered in 2.2, and if it is clausal, $K_{gs}(G)$ was altered in 2.4. Since G must serve predicative function, it can govern any adverbial modifier that modifies all predicate heads (such as the sentence modifiers that sometimes introduce Russian sentences). The predicate modifiers are described by

D_{lg}(Pred) and D_{2g}(Pred), which are stored as part of the CT routine and incorporated in the adverbial-type government segments of G by logical summation. The complete output, to be finished by the parsing-logic routine, will include the occurrence numbers of G and D, note that G is governor, and that D serves subjective function.

3'. *Complementary function test (i-th complement)*

Test $L_{gci}(G)$ & $L_t(GD) = L$.

If L = 0, stop.
If L ≠ 0, do complementary substantive type test (3'.1).

This test permits governors to be classified according to location of i-th complement. Thus, nouns generally require their complements to follow.

3'.1. *Complementary substantive type*

Test $T_{gci}(G)$ & $T_d(D) = T$.

If T = 0, stop.
If T^n = 1, test complementary nominal properties (3'.2).
If T^k = 1, test complementary substantive clause type (3'.3).
If T^l = 1, prepare output for complementary function (3'.6).
If T^h = 1, test complementary prepositional-phrase type (3'.5).
If T^a = 1, prepare output for complementary function (3'.6).

In Russian, complementary functions can be served by nominals, clauses, infinitives, prepositional phrases, and adjectivals. Since T_d contains at most a single 1, this test leads to a stop or to exactly one branch. If the possible complement being tested is nominal, and infinitive, or a prepositional phrase, further tests must be performed, but no further agreement tests are known for infinitive complements and the requirements for adjectivals are set aside for the time being.

3'.2. *Complementary nominal properties*

Test $N_{gCi}(G)$ & Nd(D) = N.

If N = 0, stop.
If N 7^ 0, replace Ng_{Ci} (G) with N and test prepositional governor (3'.2.1).

If the complement is nominal, agreement in case (and possibly other nominal properties) must be determined. Using the full nominal-properties frame for these segments tends to waste space, but N_d is involved both with government of the item as complement and with modification by an adjectival; hence it is convenient to keep it as a single segment.

3'.2.1. *Prepositional governor*

Test $T^h_d(G) = 1$.

If yes, replace $H_d(G)$ with $H_d(G)$ & $H_d(D)$ and prepare output for complementary function (3'.6).
If no, prepare output for complementary function (3'.6).

This operation, simply a part of output preparation, establishes the type of prepositional phrase headed by G, (supposing, of course, that G is a preposition). The type of phrase is defined by the identity of the preposition and the case of its complement (see Sec. 4.4). $H_d(D)$ indicates the case of D, $H_d(G)$ indicates the identity of G. The product, therefore, identifies the phrase. Note that H_d is stored with nominals even though it is never used in testing their agreement with any other kind of item.

3'.3. *Complementary substantive-clause type*

Test $K_{gcl}(G)$ & $K_d(D) = K$.

If K = 0, stop.
If K ≠ 0, test clause-complement location (3'.4).

This test determines whether the substantive clause proposed as *i*-th complement is of a type that can be accepted by the proposed governor.

3'.4. *Clause-complement location*

Test $L_{dc}(D)$ & $L_t(GD) = L$.

If L = 0, stop.
If L ≠ 0, replace $K_{gci}(G)$ with K and prepare output for complementary function (3'.6).

In 3' above, a test for location requirements imposed by the governor was made. Here the location requirements of the dependent are examined.

3'.5. *Complementary prepositional-phrase type*

Test $H_{gci}(G)$ & $H_d(D) = H$.

If H = 0, stop.
If H ≠ 0, replace $H_{gci}(G)$ with H and prepare output for complementary function (3'.6).

The prepositional phrase proposed as *i*-th complement is checked, controlling identity of the preposition and case of the object, against the requirements that the proposed governor imposes on its *i*-th complement.

3'.6. *Output for complementary function*

Set $F^{ci}_g(G) = 0$.
$T_{gci}(G) = T$.
Do global properties routine (9).

The governor, since it has an i-th complement, cannot have another; the function is singular. Altering $T_{gci}(G)$

here completes the marking of G to show exactly what type of *i*-th complement it governs; if the *i*-th complement is nominal, $N_{gci}(G)$ was altered in 3.2, if clausal, $K_{gci}(G)$ was altered in 3.3, and if prepositional, $H_{gci}(G)$ was altered in 3.5. The complete output, to be finished by the parsing-logic routine, will include the occurrence numbers of G and D, note that G is governor, and that D serves *i*-th complementary function.

4. *First auxiliary function*

Test $L_{dx}(D)$ & $L_t(GD) = L$.

If $L = 0$, stop.
If $L \neq 0$, test first auxiliary type (4.1).

The auxiliary is allowed to control the location of its potential governor.

4.1. *First auxiliary type*

Test $X_{lg}(G)$ & $X_{ld}(D) = X$.

If $X = 0$, stop.
If $X^f = 1$, prepare output for finitive auxiliary function (4.2).
If $X^t = 1$, test tensal auxiliary function (4.3).

The operations required for finitive auxiliaries of infinitives and for tensal auxiliaries of short-form adjectivals are somewhat different, since the infinitive cannot have obtained a subjective dependent before atr tachment of the auxiliary, but the adjectival can.

4.2. *Output for finitive auxiliary function*

Set $F_d(G) = 000\ 000\ 001$.
$F^s_g(G) = 1$.
$F^{xl}_g(G) = 0$.
$T_{gs}(G) = T_{gs}(G)$ & $T_{gs}(D)$.
$N_{gs}(G) = N_{gs}(D)$.
$D_{1g}(G) = D_{1g}(G)$ v $D_{lg}(Pred)$.
$D_{2g}(G) = D_{2g}(G)$ v $D_{2g}(Pred)$.

Release restriction on order of acquisition of dependents by G.
Do global properties routine (9).

Since G governs a finitive auxiliary, it can serve no function but the predicative. It can govern a subject; the type of that subject is jointly controlled by the properties of G and of D, and the nominal properties of that subject are controlled by the properties of D. The governor cannot govern another first auxiliary. Any item that can serve finitive auxiliary function must therefore have information in T_{gs} and N_{gs}, even though it cannot, itself, govern a subject. Since G must serve predicative function, it can govern any adverbial modifier that modifies all predicate heads (see remarks under 2.5—Output for subjective function). The restriction on order of acquisition of dependents prevents the parsing-logic routine from constructing the same structure by two different sequences of tests.[13] It is released here because the infinitive can govern a new kind of dependent; the infinitive with auxiliary is so different in quality from the infinitive without that it must be regarded as a new object.

4.3. *Tensal auxiliary function*

Test $F^s_g(G) = 0$.

If yes, test nominal properties of subject present (4.4).
If no, test subjective substantive type (4.5).

A positive response to this test means that G already governs a subject, with which the proposed auxiliary must agree. A negative response means that the subject of the short-form adjectival has not yet been attached, hence that its properties can be controlled in part by the auxiliary.

4.4. *Nominal properties of subject present*

Test $N_{gs}(G)$ & $N_{gs}(D) = N$.

If $N = 0$, stop.
If $N \neq 0$, set $F^{xl}_g(G) = 0$ and do global properties routine (9).

The subject already attached to G has the properties shown by $N_{gs}(G)$, by virtue of the alterations performed in 2.2. These properties do or do not agree with those allowed to its governor by the tensal auxiliary D. If they do, the output preparation required will be performed by the parsing logic routine: G will be noted as the governor of D, and D will be marked as serving first auxiliary function.

4.5. *Subjective substantive type*

Test $T_{gs}(G)$ & $T_{gs}(D) = T$.

If $T = 0$, stop.
If $T \neq 0$, test subjective nominal properties (4.6).

The substantive types of the subjects allowed by the possible auxiliary and required by the short-form adjectival, must overlap. There is no need to branch on type here, since it is assumed that every possible governor of a tensal auxiliary allows a substantive subject for which nominal properties would have to be tested.

4.6. *Subjective nominal properties*

Test $N_{gs}(G)$ & $N_{gs}(D) = N$.

If $N = 0$, stop.
If $N \neq 0$, prepare output for tensal auxiliary function (4.7).

Overlap in nominal properties between short-form adjective and tensal auxiliary is required.

4.7. *Output for tensal auxiliary function*

Set $T_{gs}(G)$ = T.
$N_{gs}(G)$ = N
$F^{x1}{}_{g}(G)$ = 0.
$F_{d}(G)$ = 000 000 001.
$D_{1g}(G)$ = $D_{lg}(G)$ vD_{lg}(Pred).
$D_{2g}(G)$ = $D_{2g}(G)$ v D_{2g}(Pred).

Do global properties routine (9).

The type and nominal properties of any subject subsequently accepted by G must be acceptable to it and to D. With an auxiliary, the governor can only serve predicative function, and its capacity to govern modifiers is therefore increased. It is not necessary to release the restriction on order of acquisition of dependents by G, since if G had previously been tested for connectability with an appropriate subject the connection would have been made.

5. *Second auxiliary function*

Test $L_{dx}(D)$ & $L_{t}(GD)$ = L.

If L = 0, stop.
If L ≠ 0, test complementation of governor (5.1).

Second auxiliary function is served for verbs by the negative particle *ne.* It must precede its governor. If the governor can govern a first complement, the character allowed that complement is altered by the presence of *ne.*

5.1. *Complementation of governor*

Test $F^{c1}{}_{g}$ (G) = 0.

If yes, prepare output for second auxiliary function (5.6).
If no, test substantive complementation (5.2).

A positive answer means that no further first complement can be attached to the governor, hence that the attachment of *ne* cannot influence it.

5.2. *Substantive complementation*

Test T^{a}_{gc1} (G) = 0.

If yes, test clausal complementation (5.4).
If no, test accusative complementation (5.3).

A positive answer means that no nominal complement can be attached, hence that the presence of *ne* cannot influence it.

5.3. *Accusative complementation*

Test $N_{gc1}(G)$ & N(Acc) = N.

If N = 0, test clausal complementation (5.4).
If N ≠ 0, replace $N_{gc1}(G)$ with $N_{gc1}(G)$ v N(Gen), release order of acquisition of dependents restriction on G, and prepare output for second auxiliary function (5.6).

If the governor of *ne* can take an accusative nominal first complement, it may, in the presence of *ne,* take a genitive instead. N(Acc) and N(Gen), stored with the CT routine, are nominal properties segments containing 1's for the accusative and genitive cases, repectively.

5.4. *Clausal complementation*

Test $T^{k}_{gc1}(G)$ = 0.

If yes, prepare output for second auxiliary function (5.6).
If no, test *chto*-clause complementation (5.5).

If the governor cannot take a clausal first complement, the presence of *ne* cannot influence its character.

5.5. *Chto-clause complementation*

Test K^{c}_{gc1} (G) = 0.

If yes, prepare output for second auxiliary function (5.6).
If no, set K^{y}_{gc1} (G) = 1, release order of acquisition of dependents restriction on G, and prepare output for second auxiliary function (5.6).

Here the superscript y refers to the position of the K frame marking government of a *chtoby* clause, c to the position for government of a *chto* clause. If the governor of *ne* can govern a *chto* clause as first complement, it can alternatively, in the presence of *ne,* govern a *chtoby* clause.

5.6. *Output for second auxiliary function*

Set $F^{x3}{}_{g}$ (G) = 0.

Do global properties routine (9).

The parsing-logic routine will mark G as governor, D as dependent with second auxiliary function.

6. *Output for third auxiliary function*

Set $F^{c2}{}_{g}$ (G) = 1.
$T_{gc2}(G)$ = 10010.
$N_{gc2}(G)$ = N(Gen).
$H^{chem}{}_{gc2}(G)$ = 1
$F^{x3}{}_{g}(G)$ = 0

Do global properties routine (9).

Third auxiliaries depend on adjectivals and mark them as of comparative degree. It is assumed here that comparative adjectivals thus marked can govern genitive nominals or *chem*-phrases. It is also assumed that ad-

jectivals cannot govern second complements under any other circumstances except morphological marking of comparative degree, and then the possibilities of second complementation are identical with those indicated here. The parsing-logic routine will mark G as governor, D as dependent with third auxiliary function.

7. *Modifier function*

Test $M_g(G)$ &$M_d(D) = M$.

If $M = 0$, stop.
If $M^a = 1$, test nominal properties (7.1).
If $M^d = 1$, test adverbial location (7.3).

Modifiers are adjectival or adverbial. Adjectival modifiers agree with their governors in nominal properties; adverbial modifiers agree in type.

7.1. *Nominal properties*

Test $N_d(G)$ & $N_d(D) = N$.

If $N = 0$, stop.
If $N \neq 0$, test adjectival location (7.2).

If there is overlap in case, number, gender, animation, and person, then the adjective can modify the nominal governor. Ambiguity of the governor may be reduced by attachment of the adjective, but any remaining ambiguity can be retained; it is not necessary to generate a separate output for each value of the nominal-properties variable in which there is agreement. Note that $N_d(G)$ is used here; this function can be called strongly endocentric, in the sense that properties of the governor material for its function as dependent are also material for its relation to the adjective it governs.

7.2. *Adjectival location*

Test $L_t(GD)$ & $L_{da}(D) = L$.

If $L = 0$, stop.
If $L \neq 0$, replace $N_d(G)$ with N and prepare output for modifier function (7.6).

The reduction in ambiguity is relevant to the subsequent connectability of the governor, whether with another adjectival dependent or with a governor. The location test requires that adjectives precede or follow their governors in accordance with their type.

7.3. *Adverbial location*

Test $L^1_t(GD) = 1$.

If yes, test preceding adverbial type (7.4).
If no, test following adverbial type (7.5).

A positive result indicates that the potential dependent precedes the potential governor.

7.4. *Preceding adverbial type*

Test $D_{lg}(G)$ & $D_{ld}(D) = D$.

If $D = 0$, stop.
If $D \neq 0$, prepare output for modifying function (7.6).

7.5. *Following adverbial type*

Test $D_{2g}(G)$ & $D_{2d}(D) = D$.

If $D = 0$, stop.
If $D \neq 0$, prepare output for modifying function (7.6).

These two tests, entirely parallel, are separated because the lists of types of adverbial dependents that can, respectively, precede and follow their governors are not identical.

7.6. *Output for modifying function*

Do global properties routine (9).

Since the modifying function, whether adjectival or adverbial, is optional, it is not possible to set $F^m_g(G) = 0$, nor can $D_{1g}(G)$ or $D_{2g}(G)$ be replaced with D to indicate the type of modifier present—another type of modifier may be found later. It is not even possible to put a zero in the position of $D_{1g}(G)$ or $D_{2g}(G)$ where a match was found, since in principle another modifier of the same type might be found; the opposite principle would imply, what one might suspect, that the modifier types are in fact singular functions. For the present, it must be assumed that the parsing-logic routine will mark G as governor, D as dependent with *a certain type* of modifying function, since its type may be necessary to subsequent (postsyntactic) operations.

8. *Predicative function*

Test $L^2_t(GD) = 0$.

If yes, stop.
If no, prepare output for predicative function (8.1).

Predicative function is served by the principal occurrence of a sentence. The governor is a subordinate conjunction. Since one occurrence in a full sentence is independent, that occurrence will also be said to serve predicative function without a governor, but this test applies only when, as in subordinate clauses, a governor is actually present. Such a governor must always, as the test requires, precede the predicative dependent.

8.1. *Output for predicative function*

Set $F^p_g(G) = 0$.

The predicative function is singular. Reference to the global properties routine is omitted because marking with a subordinate conjunction and marking with

kotoryj or *li* are mutually exclusive alternatives. The parsing-logic routine will mark G as governor, D as dependent with predicative function.

9. *Global properties*

Test G(D) = G.

If G = 0, stop.
If $G^a = 1$, do global nominal-properties routine (9.1).
If $G^1 = 1$, prepare output (9.3).

A zero result means that the dependent, whether intrinsically or as a result of previous attachment of some deeper dependent, has no global properties. If $G^a = 1$, either the dependent in the newly-formed connection is a form of *kotoryj,* or it governs, directly or indirectly, some form of *kotoryj.* Likewise, $G^1 = 1$ indicates the presence of *li.*

9.1. *Global nominal properties*

Test J(D) = 0.

If yes, do determination of global nominal properties (9.2).
If no, prepare output (9.3).

For every entry in the dictionary, including *kotoryj,* J is blank. This segment is filled out only by the application of 9.2. Hence if $G^a(D) = 1$, and J(D) = 0, then D is an occurrence of *kotoryj.*

9.2. *Determination of global nominal properties*

Test $N_d(D)$ & N(Masc) = N.

If N = 0, do (9.2.1).
If $N \neq 0$, set $J^m(G) = 1$ and do (9.2.1).

9.2.1. *Feminine*

Test $N_d(D)$ & N(Fem) = N.

If N = 0, do (9.2.2).
If $N \neq 0$, set $J^f(G) = 1$ and do (9.2.2).

9.2.2. *Neuter*

Test $N_d(D)$ & N(Neut) = N.

If N = 0, do (9.2.3).
If $N \neq 0$, set $J^n(G) = 1$ and do (9.2.3).

9.2.3. *Plural*

Test $N_d(D)$ & N(Plu) = N.

If N = 0, prepare output (9.3).
If $N \neq 0$, set $J^p(G) = 1$ and prepare output (9.3),

These four tests are used to reduce the 36-position segment $N_d(D)$ to the 4-position segment J(G). N(Masc), N(Fem), N(Neut), and N(Plu) are four nominal-properties segments stored with the CT routine and containing 1's in their masculine-singular, feminine-singular, neuter-singular, and plural positions, respectively.

9.3 *Output for global properties*

Set G(G) = G(D).

With this step, carrying forward the global properties of the dependent as the global properties of the new governor, the global-properties routine and the CT routine are complete and the parsing-logic routine can begin its search for a new pair of possibly connectable occurrences.

Punctuation, not discussed here, is used to facilitate or prevent connections. It is also used to mark occurrences or connected sequences of occurrences that can serve as appositives or as adjectival dependents of preceding governors. And, in addition, punctuation is used to close off sentences and clauses. When a connected sequence, surrounded by appropriate punctuation, is found to be headed by an occurrence that can serve predicative function, the sequence is regarded as an independent sentence. With different boundaries and a head occurrence that can serve predicative function and is marked with global properties, a connected sequence is regarded as a subordinate clause and given a new grammatical description permitting it to serve as adjectival-modifying dependent or as clausal-substantive dependent.

6. Remarks on Programming

The programming of a CT routine such as the one described in Sec. 5 is fairly straightforward on any computer with large enough storage cells, Boolean operations, indexing, and indirect addressing. The flow-chart in Fig. 1 shows the structural simplicity of the whole routine, and inspection of the instructions used in Sec. 5 proves that only a few basic patterns of testing and alteration of grammar-code symbols are needed. The programmer must remember, however, that as many as 10,000 connectability tests may be required in the processing of one long sentence, and attempt, in every way possible, to reduce the average time consumed per test.

One somewhat delicate matter, given the importance of speed, is the handling of the functions test, which has 2^9 possible outcomes (any combination of the nine functions may have to be tested). On some machines (such as the I.B.M. 7090) there is an instruction that simultaneously modifies the contents of an index register and independently transfers control. (The 7090 instruction is TXI[14]). For example, let T denote an index register, and consider a language with just three functions. The program suggested first forms $F_g(G)$ & $F_d(D)$

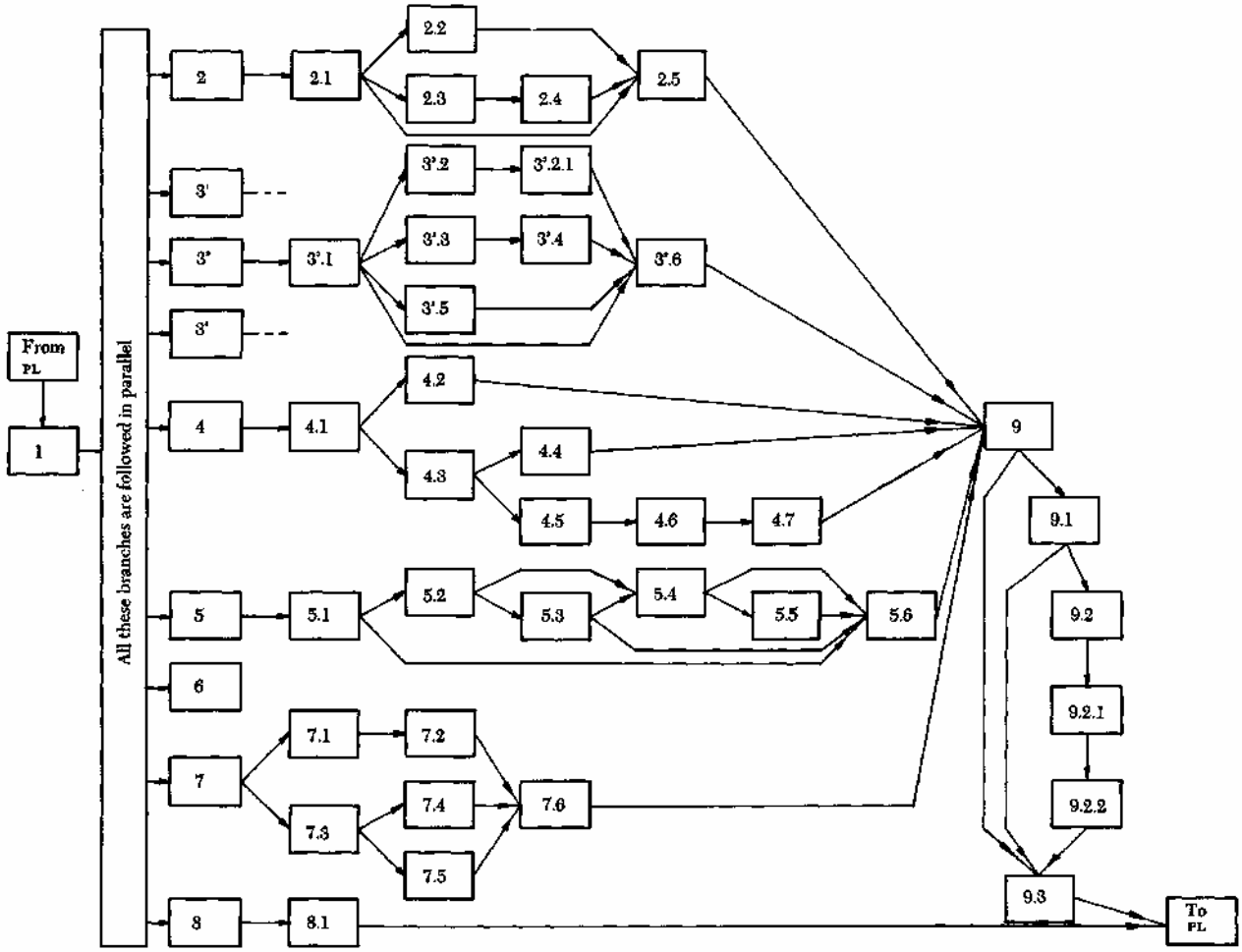


FIGURE 1. CONTROL FLOW FOR CT ROUTINE.

and takes the complement of the product (or, equivalently, takes the union of the complements of $F_g(G)$ and $F_d(D)$). The result contains a zero for each function to be tested: 000 means three functions to test, 110 means the third function (say F_3) alone, etc. A table with 2^n cells for n functions is stored with the CT routine, occupying cells Z, Z – 1, . . ., Z – 2^n + 1. The complemented product of the function segments is stored in T and control is passed to cell Z – con(T) by an indexed transfer. (Here con(X) means the number stored in the cell with address X.) If the computer is the 7090, this cell contains a *decrement,* the address of the index register T, and another address Y. When control passes to Z – con(T), the decrement in that cell is added to T and control passes thereafter to Y. In each cell of the table (see Fig. 2), the decrement is a string of zeros with a 1 in the position representing a certain function, and Y is the address of the first cell of a subroutine for that function. With three functions,

FIGURE 2. CONTROL TABLE FOR FUNCTIONS TEST

	Cell contents				*Functions remaining*
Cell address	*Instruction*	*Decrement*	*Index register*	*Transfer address*	*to be tested*
θ — 0	TXI	1	T	F_s	1,2,3
θ —1	TXI	2	T	F_2	1,2
θ - 2	TXI	1	T	F_3	1,3
θ — 3	TXI	4	T	F_1	1
θ — 4	TXI	1	T	F_3	2,3
θ — 5	TXI	2	T	F_2	2
θ — 6	TXI	1	T	F_3	3
θ — 7	TXI	—	—	Out to PL	—

there are three subroutines to be referenced. The decrement for function 1 is 000 . . . 100 = 4; for function 2, 000 . . 010 = 2; and for function 3, 000 . . . 001 = 1 (the numbers on the left are binary, those on the right octal). Each function-test subroutine ends with a

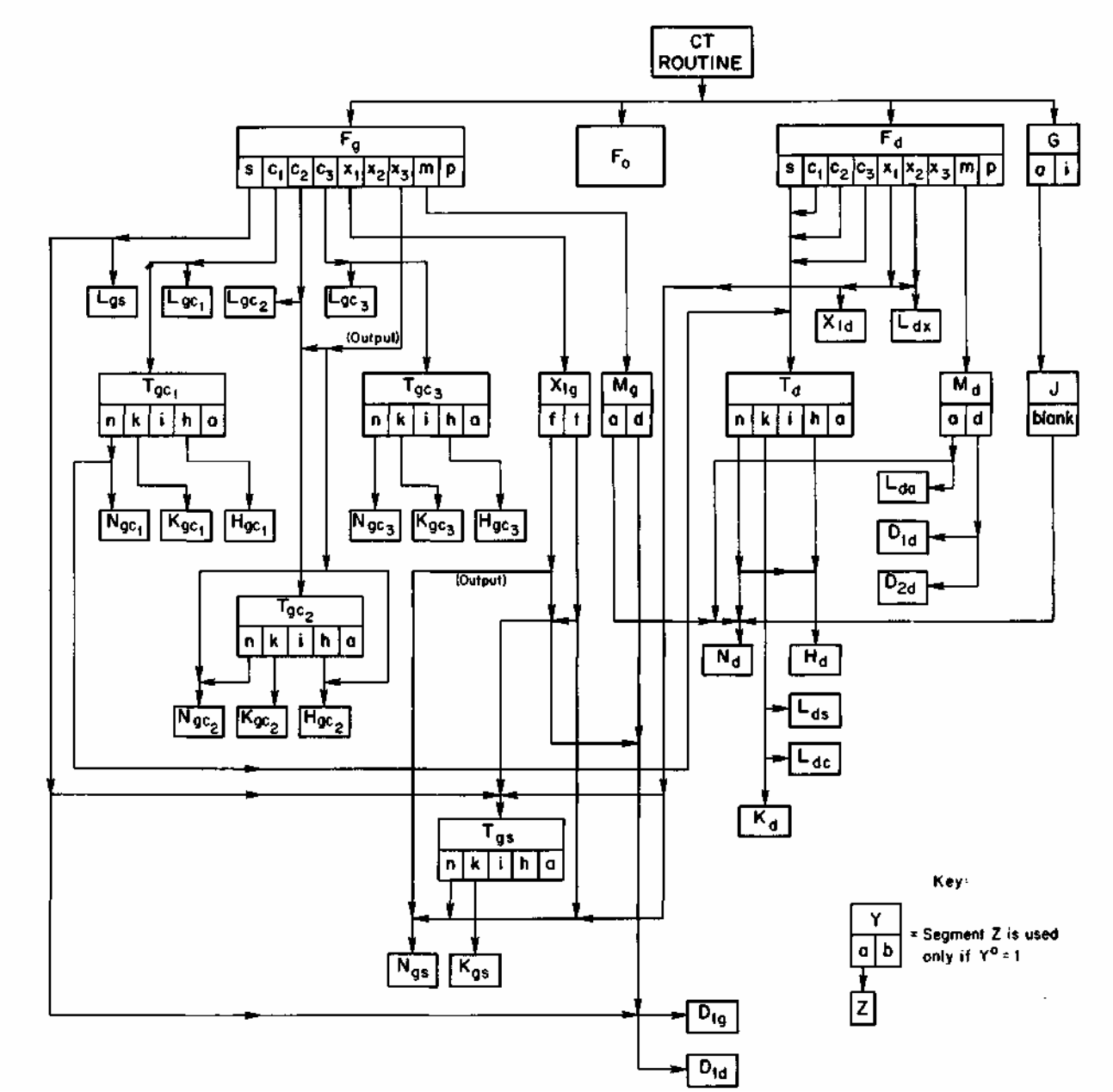


FIGURE 3.

TXI instruction transferring control to Z – con(T); when that transfer occurs, con(T) has been altered by the insertion of a 1 for the function just tested, so that either a new function is tested or the CT routine ends[15]). Another problem, less easily solved, is that of storage. During sentence-structure determination hundreds of intermediate units have to be held in high-speed storage. If each unit is represented by a grammar-code symbol of 400-500 bits, half of a 32,000-cell memory can easily be filled during the processing of a long sentence, and in some cases the whole memory may not suffice. Since each segment of a grammar-code symbol is needed only conditionally, according to functions governed or served and type of agreement required (see Fig. 3), packing is not difficult; unpacking, of course, is time consuming, and a plan for rapid access to individual segments is important if the high intrinsic speed of the code-matching system is to be retained. A possible system is offered as an illustration of what can be done.

A storage cell is a string of bit positions; in the I.B.M. 7090, a cell has 36 positions designated S, 1, 2, . . ., 35, and divided into left and right half cells. Each *frame* is assigned a definite set of positions (as in

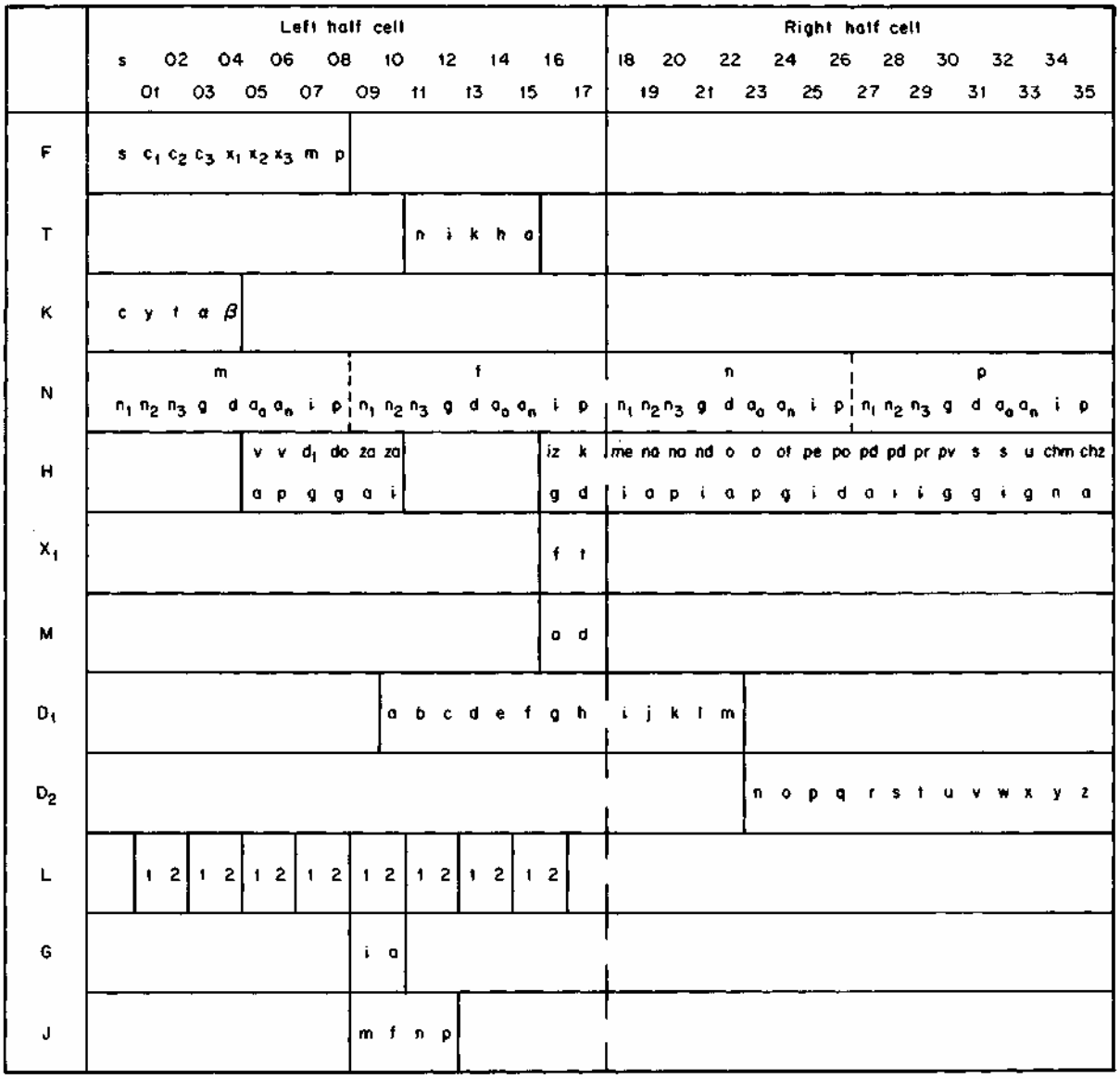
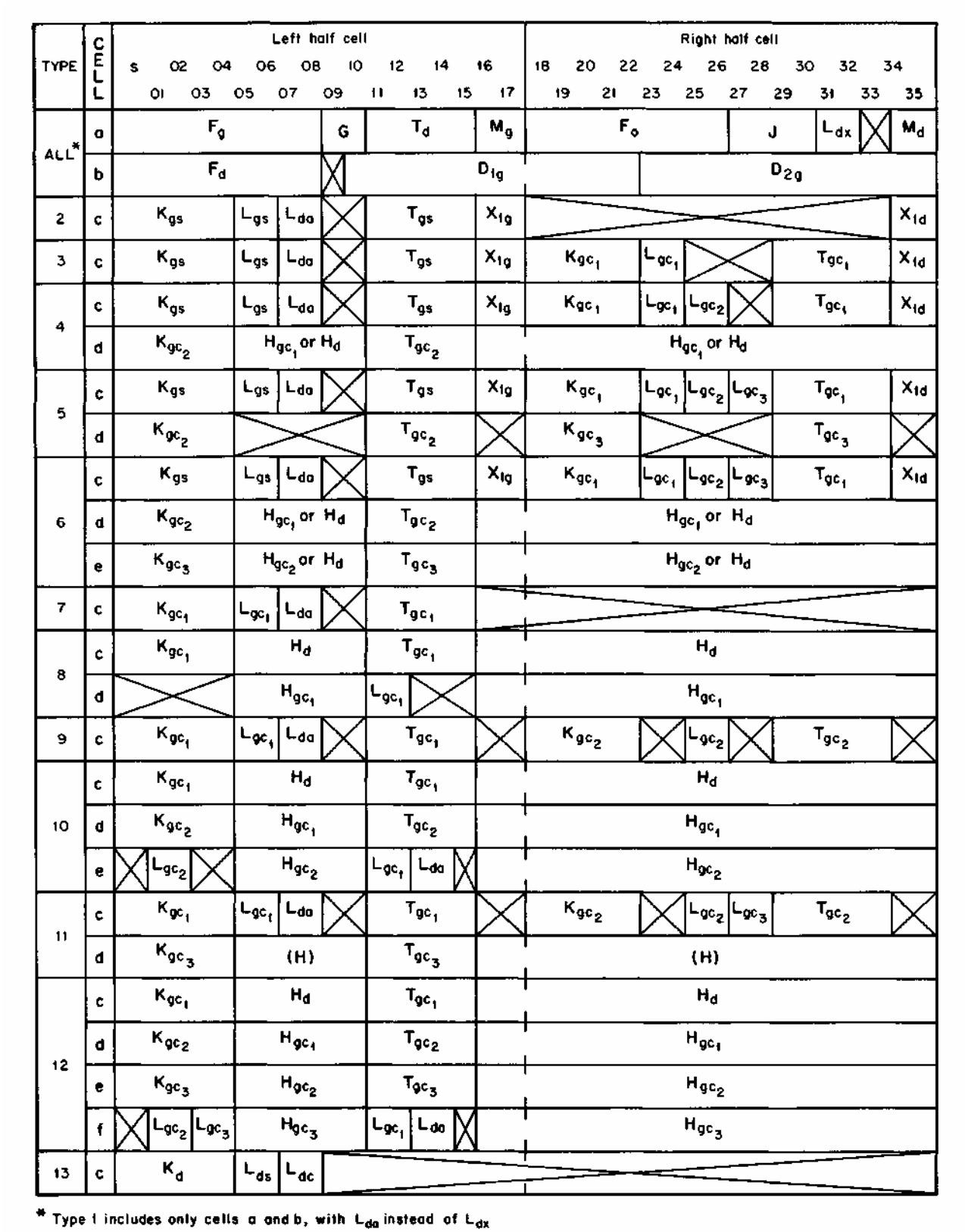


FIGURE 4.

Fig. 4); if the frame is less than 18 bits, its location is relative to the limits of a half cell. Thus, in the illustration, the functions frame occupies the first nine positions of either the left half cell or the right. In what follows, it will be impossible to keep the *location* frames in fixed positions; several alternatives are allowed.

The frames are also grouped in two storage categories, long and short. The long frames are N, H, and D; all the others are short. Thirteen types of grammar-code symbols, numbered 1-13, are defined according to the combination of short segments that they contain. Every type contains F_g, F_d, F_o, T_d, M_g, M_d, D_{1g}, D_{2g}, G, and J, and all except No. 13 contain L_{da}. These segments are stored in the first two cells assigned to any grammar-code symbol; these cells are designated a and b (see Fig. 5). The other short segments are stored, in various combinations as needed, in cells c, d, e, and f. The long segments are stored in additional cells, beginning after the last cell of short segments (in several types of packed symbols, H segments are stored together with short segments; see the figure).

In order to reach a particular segment of the grammar-code symbol of a unit, it is necessary to know, first, the location of cell a for that unit; this information is supplied to the CT routine initially. Next, the relative address (a, b, c, etc.) of the cell containing the segment must be obtained. For many segments, this relative address is invariant and can be put in the CT routine as a constant, but for others it depends on the type of symbol. Hence each grammar-code symbol must be



* Type 1 includes only cells a and b, with L_{da} instead of L_{dx}

FIGURE 5.

accompanied by a cell containing an indication of its type. With type number and segment name, the relative address can be obtained from a table and put in an index register. An indirect-address, indexed instruction puts the cell or half-cell wanted into an operating register. Since most half cells contain two or more segments, the operating register must now be masked, i.e., the logical product of its content and a string of 1's and 0's must be taken. The mask string has 1's in the positions occupied by the segment and zeros elsewhere. Since either left or right half cells can be moved to the right half cell of an operating register, the position of the segment is now invariant and no shifting across the register (a slow operation) is needed. This, of course, with the exception of the location segments, but their use in the CT routine is such that it may be more economical to use them where they are than to shift them.

The long segments are needed in so many different combinations that naming a separate type of grammar-code symbol for each combination would be awkward. Instead, a cell can be reserved for their relative addresses; in this cell, fixed positions contain the relative address of N_d, or zeros if N_d is not available; other positions contain the relative address of N_{gs}, N_{gcl}, etc. It is possible, by using different origins for different types of grammar-code symbols and storing the long segments always in the same order, to limit the number of distinct relative addresses needed for any H segment to less than 7 and for any N or D to less than 15. Hence 3-bit addresses for the H's and 4-bit addresses for D_{1d} and D_{2d} (always stored together in a cell) are adequate. There are four H's, 5 N's, and 1 D—making 36 bits of relative addresses!

Techniques for packing information are endless, and the plan just described is likely not to be the best compromise between volume and speed of access. Quite a substantial saving of space is effected, but several successive operations are needed to reach a single segment. Another plan would be to store N's and H's whenever the functions of a unit can call for them; the 13 types of grammar-code symbols would be reduced to nine, each with a fixed set of H and N segments. This plan would increase the average size of grammar-code symbols but shorten access time a little.

Another programming question is that of summary encoding. The N frame, with 36 positions, can contain any one of 2^{36} arrays of 1's and 0's; since only a few dozen different arrays will appear in the grammar-code symbols of Russian forms, those that do appear can be given abbreviated symbols and a list of array-symbol pairs stored. The dictionary can furnish the abbreviated symbol; the full array is needed only at the moment of code matching. Since the conversion from array to abbreviated symbol requires binary search, whereas the opposite conversion can be performed by indexed addressing, speed of operation requires that the conversion go in one direction only. Hence it does not seem economically reasonable to store abbreviated symbols during the sentence-structure determination process, and the only question is *when* to decode the dictionary entries. The decoding can be done at the time of dictionary lookup, and then needs to be done only once for each distinct form encountered in text—but the expanded grammar-code symbols have to be moved in and out of storage. It can be done for the units in a short span of text at the beginning of sentence-structure determination over that span, but then, although no movement in and out of storage is required, the decoding has to be done for each occurrence. This question has not been settled as yet, and depends on relative speeds of decoding and data transmission.

7. Morphology and Syntax

Terse summaries of morphology and of syntax, each taken separately, tend to be quite short. The brevity of this paper, although it is quite incomplete, is at least suggestive. The statement of Russian syntax included here consists, in fact, of the format in Sec. 4 and the CT routine in Sec. 5. To be added are routines (more than one will be needed) for coordinative functions and, very likely, additional steps in the routine of Sec. 5 for tense sequence, inter-complementary agreements, and so on. Even with these additions, the whole statement of Russian syntax would be extremely short, and corresponding statements of morphology (i.e., of construction rules that apply within the form) are of similar length. Whether statements about higher strata (transformational or sememic statements) can be equally short is unknown, but they may well be. The ease with which natural languages are learned makes their simplicity almost certain—at least their simplicity in certain senses.

There remains the fact, however, that standard treatises on the grammars of modern languages are large and dense with detail. This detail seems mostly to concern interstratal relationships, and that fact is worth noting as a guide to future research. The syntactic behavior of morphologically defined categories is studied, and morphologically unusual items are analyzed, syntactically, one by one. Since not all syntactic properties can be derived from morphological properties, sememically defined categories are also considered. This plan of presentation, although often somewhat confusing because the morphologico-syntactic correlations are often confounded with the sememo-syntactic correlations, has merit.

Suppose that the complete description of a language, beyond the phonological or graphic stratum, consists of formats and CT routines for morphological, syntactic, and sememic levels (not strata, since morphology and syntax belong to one stratum), together with a dictionary and rules for interlevel conversion. Suppose, furthermore, that the CT routines and formats are all simple. The conversions may not be. One conversion was mentioned at the end of Sec. 6, in the guise of a storage

problem: Syntactic grammar-code symbols for forms have to be obtained as the end product of a dictionary-lookup operation that may involve a morpheme list and a CT routine; syntactic properties then have to be ascribed to stem morphemes, affix morphemes, and their constructions. Design of a good routine for this purpose calls for exactly the kind of information supplied, with more or less precision and accuracy, in large grammars. The syntactic-to-sememic conversion, since it crosses a stratal boundary, calls for another dictionary lookup, but again for information about inter-level relationships. Despite the grammars, the amount of such information still to be collected and systematized can hardly be exaggerated.

8. Acknowledgments

The author is indebted to K. E. Harper, C. F. Hockett, M. J. Kay, Y. Lecerf, S. L. Marks, B. Vauquois, D. S. Worth and T. W. Ziehe for the benefit of their criticism and suggestions.

1 A routine invented by John Cocke is described by D. G. Hays in "Automatic Language-data Processing", Chapter 17 of *Computer Applications in the Behavioral Sciences,* Prentice-Hall, 1962. Several others are reported in *1961 International Conference on Machine Translation of Languages and Applied Language Analysis,* H. M. Stationery Office, 1962.

2 Standard references on dependency theory include L. Tesnière, *Elements de Syntaxe Structurale,* Klincksieck, 1959; Y. Lecerf, "Programme des Conflits, Modèle des Conflits," *La Traduction Automatique,* vol. 1, no. 4 (October, 1960), pp. 11-20, and vol. 1, no. 5 (December, 1960), pp. 17-36; and D. G. Hays, "Grouping and Dependency Theories", in H. P. Edmundson, ed., *Proceedings of the National Symposium on Machine Translation,* Prentice-Hall, 1961, pp. 258-266.

3 Three examples of specialized routines, each intended to find one or a few structures for any sentence, but not all: D. G. Hays and T. W. Ziehe, *Studies in Machine Translation—10: Russian Sentence-structure Determination,* RM-2538, The RAND Corporation, 1960; Ida Rhodes, "A New Approach to the Mechanical Syntactic Analysis of Russian," *Mechanical Translation,* vol. 6; and a system being constructed by E. D. Pendergraft at the Linguistics Research Center of the University of Texas, thus far described only in a series of quarterly progress reports to the U.S. Army Signal Corps and the National Science Foundation.

4 As in the work of Rhodes and Pendergraft cited above.

5 Code-matching techniques were suggested by A. F. Parker-Rhodes, "An Algebraic Thesaurus", presented at an International Conference on Mechanical Translation, Cambridge, Mass., Oct. 15-20, 1956. Ariadne Lukjanow, then at Georgetown University, used the term to apply to a method that she proposed, and Paul Garvin (of Bunker-Ramo, Inc.) has developed a system, somewhat different from that proposed here, for Russian syntax.

6 J. Lambek, "The Mathematics of Sentence Structure," *American Mathematical Monthly,* vol. 65, no. 3 (1958), pp. 154-170.

7 Y. Bar-Hillel, C. Gaifman, and E. Shamir, "On Categorial and Phrase-structure Grammars," *Bulletin of the Research Council of Israel, Section F,* vol. 9, no. 1 (1960), pp. 1-16.

8 H. Gaifman, "Dependency Systems and Phrase Structure Systems," P-2315, The RAND Corporation, 1961.

9 Cf. the discussion of strong and weak government in L. N. Iordanskaya, *Two Operators for Processing Word Combinations with "Strong Government" (for Automatic Syntactic Analysis),* Moscow, 1961. Translated in JPRS 12441, U.S. Joint Publications Research Service, 1962.

10 Cf. Hockett's discussion of "Construction types;" C. F. Hockett, *A Course in Modern Linguistics,* Macmillan 1958, pp. 183-208.

11 K. E. Harper, *Procedures for the Determination of Distributional Classes,* RM-2713-AFOSR, The RAND Corporation, 1961.

12 Iordanskaya, op. cit. fn. 9.

13 This restriction, suggested by Y. Lecerf, is discussed in D. G. Hays, *Research Procedures in Machine Translation,* RM-2916, The RAND Corporation, 1961.

14 International Business Machines Corporation, *Reference Manual, IBM 7090 Data Processing System,* revised March 1962, p. 39.

15 This plan was devised in a discussion with M. J. Kay, S. L. Marks, and T. W. Ziehe.