7

DECIPHERMENT MODELS

The first research models we present may be called MODELS OF LINGUISTIC DECIPHERMENT, for in principle they can have an applied, deciphering function in addition to their primary theoretical function, which is to provide an objective basis for linguistic concepts and assertions. As stated earlier, the primary information for such models is a text about which nothing is known in advance. One knows neither the language in which the text is written — the code used to 'encode' it, nor the genetic relations of this language to other known languages, nor translations of the text into known languages, nor the subject matter of the text. For example, let us suppose that a text in some Martian language has fallen into our hands. Its subject matter is life on Mars, of which we know nothing, and it is not related to any text in any known language. To decipher the text we can use only our ability to distinguish black dots from white ones (which also can be easily 'learned' by an electronic computer). All the other information we need, namely, the elementary units of the text (the letters or sounds, the morphemes, the words, the sentences, and the meanings), the classes of elementary units (the vowels and consonants, the lexical and grammatical morphemes, the parts of speech, the types of sentences, the semantic fields), and the rules governing the combination of units from different classes (the syntactic relations between words in sentences, for instance), must be drawn from the text in an entirely automatic manner.

There could be a sequence of algorithms, each solving one of the problems named above. The input information for the first algorithm would be the text and data on the black and white dots (the ability to distinguish between them), and the output of this algorithm would be information on the alphabet of symbols (letters, for instance) by which the text is represented. Each succeeding algorithm would receive as input the information produced by the preceding one. Thus the second algorithm would find in the alphabet of symbols discovered by the first algorithm vowels and consonants, then classes of vowels and classes of consonants, etc. until the value of every letter is established. The next algorithm would use this information to find syllables, then morphemes, words, morpheme classes, and word classes. Once word classes are determined, it would be possible to tackle syntactic problems. Sentence boundaries would be established and then the relations between words in sentences. Finally, once all the essential features of the grammar are known, it would be possible to explore the meaning of the words and the sentences. Such algorithms would provide data sufficient to translate the text into some known language or to determine the extralinguistic reality it refers to.

If this program, formulated scientifically for the first time by Suxotin ⟨173, 174⟩, turns out to be practical, then the principal linguistic concepts can be reduced to the primitive concepts of black and white dots, a series of substantive hypotheses, and the mathematical functions describing them. Even if it turns out to be impractical, it is interesting to see how far such a description can be carried out successfully.

In the present chapter we describe several simple algorithms which fall into this category.[1] These algorithms developed, by Suxotin, Harris, and others, have a number of features in common. First, they all stem from SIMPLE and GENERAL notions about language, based on UNIVERSAL phenomena. For example: 'a letter is a specific combination of dots'; 'a morpheme is a specific combination of phonemes'; 'a word form is a specific combination of morphemes'; 'in every natural language there are at least two levels — a level of significative units (morphemes, word forms, constructions) and a level of nonsignificative units (phonemes)'; 'in every language there are lexical morphemes, and their distri-

[1] See Andreev ⟨3, 4⟩, Spang-Hanssen ⟨358⟩, and Šrejder ⟨221⟩ for discussions of related issues.

bution in texts in unlike the distribution of grammatical morphemes'; 'syntactic differences have corresponding semantic differences'; 'words that are close in their meaning can be found near each other in texts'; and so on. Secondly, such algorithms make use of information about the DISTRIBUTION of elements and their NUMERICAL PARAMETERS.[2] Thirdly, these algorithms usually operate in the following manner. First the set of POSSIBLE solutions is established, then the BEST solution in this set is found by means of so-called FAVORABILITY FUNCTIONS. Favorability functions are numerical functions which get a certain value (the highest or lowest possible, for instance) when they represent the correct (the best) solutions. Every favorability function formalizes some substantive hypothesis about possible properties of the object under study. Favorability functions were used for the first time, though not in a rigorous fashion, by Hjelmslev ⟨129, *171*⟩ and Harris ⟨295, *63*⟩, and we shall use their observations to elucidate this concept. Consider the following example. A given set of sounds has to be divided into classes corresponding to phonemes. Any arbitrary division of the set of sounds into classes can be considered a possible solution. Hjelmslev and Harris advanced a substantive hypothesis in regard to the properties of phonemes, on the basis of which it is possible to discover phonemes in a set of sounds. They suggested that although the number of phonemes is relatively small, every phoneme occurs in a large number of environments. Therefore the number of phonemes (sound classes) in the given classification and the number of environments for every phoneme can serve as the favorability functions by means of which the correct solution can be isolated from the set of possible solutions. Hjelmslev's and Harris's remarks indicate that the correct solution must be associated with a low value of the former function and a high value of the latter.

The first algorithm we present is Suxotin's ALGORITHM FOR

[2] On the relationship between structural (including distributional) methods and numerical (including statistical) methods see, e.g., ⟨111, 177, 148, 193, 199, 323, 287, 295, 293, 195, 305, 357, 358, 363, 376⟩.

SEPARATING VOWELS FROM CONSONANTS.[3] This algorithm is based on the assumption that the alphabet of letters used to represent the given text is already known. Its task is to provide a key to the rules for reading the text. Without such rules the decipherment cannot be considered complete. Any division of the alphabet into two classes is a possible solution. The algorithm formalizes the following substantive hypothesis about the properties of vowel and consonant letters: vowels and consonants alternate in texts; no text is composed of vowels only, and no text is composed of consonants only; vowels followed by consonants are followed by vowels, cf. *korova* 'cow', *palata* 'ward', *sobaka* 'dog', etc. However, this is not a strictly regular alternation; clusters of vowels and consonants are possible, cf. *teatr* 'theater', *kloaka* 'cesspool', *strax* 'fear', *vstroennyj* 'built in', etc. We can therefore be sure only of the PREVALENCE of vowel-consonant alternations over clusters: vowels are followed by consonants MORE OFTEN than by vowels, and consonants are followed by vowels MORE OFTEN than by consonants. The most frequent letter in any text is a vowel (a phenomenon demonstrated by texts in numerous languages).

Let us draw up a table to represent the frequencies of two-letter combinations (Table 1).

TABLE 1

	a_1	a_2	a_3	a_4	a_5	
a_1	.	.	....	...	..	11 5 -3
a_2	..	.	...	....	..	12 4 -2
a_3	...	....	.	..	...	13 9
a_4	....	...	..	.	....	14
a_5	.	.	....	...	.	10 4 -4

[3] We are speaking here of vowel and consonant letters, not sounds. However, what we say in regard to texts represented in letters applies just as well to texts represented in phonetic transcription.

Let us assign to each line and each column a letter of the alphabet and then write in each square the number of times the combination of the letters a_i and a_j occurs in the given text (that is, we record the number of occurrences of a_1 with a_2, with a_3, etc.). Each combination is represented by a dot in Table 1.

Let us assume that we have already classified the vowels and consonants, and the vowels are now in the upper left corner of the table (Table 2). If our hypothesis is correct, then the table

TABLE 2

	Vowels	Consonants
Vowels	1	2
Consonants	4	3

should reflect the composition of the text as follows: the number of entries in squares 1 and 3 should be small (because vowels rarely combine with vowels and consonants rarely combine with consonants), and the number of entries in squares 2 and 4 should be much larger (because vowels often combine with consonants, and consonants often combine with vowels). Consequently, in order to find in the set of possible solutions the correct one for our problem, we must change the original table into a table such as Table 2, that is, rearrange the lines and columns so that the sums of entries in the diagonal squares 1 and 3 would be the smallest.

This is achieved by the following algorithm:

(1) Find the sum of entries in each line.

(2) Check whether there are any lines with positive sums. If not — give the answer. If yes — execute (3).

(3) Find the line with the largest sum of entries and carry over to the class of vowels the letter at the head of this line.

(4) Find the column headed by the letter carried over in step (3), and deduct from the sum of entries in each line, excluding lines which have already been carried over, twice the number of entries at the intersection of the given line and the column headed by the letter carried over in step (3).

(5) Erase the former sums of entries for the lines and execute (2).

Let us apply this algorithm to Table 1. The first step yields the results 11, 12, 13, 14, and 10, written in the first column to the right of the table. The second step convinces us that all the sums are positive. The line with the largest sum of entries, which we find in executing (3), is the fourth line (the sum of its entries is 14). We carry over to the class of vowels the letter a_4. The results of the execution of (4) are shown in the second column to the right of the table. They were obtained in the following way: $11 - 2 \cdot 3 = 5$ for the first line, $12 - 2 \cdot 4 = 4$ for the second, $13 - 2 \cdot 2 = 9$ for the third, and $10 - 2 \cdot 3 = 4$ for the fifth. We execute (5), erasing the former sums, and return to (2). We establish that all the sums (5, 4, 9, 4) are positive, we find the line with the largest sum (the third), and carry over to the class of vowels the letter a_3. We carry out the remaining steps and find that no positive sums are left (cf. the numbers -3, -2, and -4 in the third column). The letters a_3 and a_4 are thus in the class of vowels, and the letters a_1, a_2, and a_3 in the class of consonants. It is obvious that the table we have obtained (Table 3) is the right one.[4]

[4] This algorithm, as Suxotin has shown, does not find the absolute minimum, as a rule. However, this complication and the methods by which it can be resolved are irrelevant at this point.

TABLE 3

	a_3	a_4	a_1	a_2	a_5
a_3	·	··	···	∷	···
a_4	··	·	∷	···	∷
a_1	∷	···	·	·	··
a_2	···	∷	··	·	··
a_5	∷	···	·	·	·

Suxotin's algorithm was tried out in a number of computer experiments conducted on Russian, English, French, German, and Spanish materials (the text in each language contained 10,000 symbols). The operation of the algorithm on the German text produced three mistakes (*s*, *h*, and *k* were classified as vowels, apparently because of the high frequency of the combinations *sch*, *ch* and *ck*); its operation on the English, French, and Russian materials produced only one insignificant and easily accountable mistake in each case; the Spanish text was processed without any mistakes at all.[5]

Obviously this algorithm does not provide sufficient information for tackling the next level, which is the morphological level. Another algorithm is required, for instance, for the distinction of syllables. In addition to this algorithm, which is different from the one for determining morphemes, another algorithm is required for transcribing syllabic writing systems into alphabetic systems, and many additional algorithms are necessary. Suxotin developed several algorithms of this kind, but we shall not treat them here, as they are quite complex. For the same reason we shall have to

[5] Ševoroškin applied analogous notions and devices in his brilliant decipherment of the Karian language (see <218a>).

forego a discussion Suxotin's most interesting morphological algorithm ⟨174⟩. Instead we shall describe Harris's simpler algorithm for identifying morphemes, or rather morphs ⟨294⟩.

Harris's algorithm does not discover morphs as such, but rather the boundaries between them. To elucidate the conception underlying this algorithm, let us consider the popular game of 'ghost'. In this game one of the players writes down the first letter of a word he chooses (a noun). Each of the following players adds a letter to the right of the letters already written down, trying not to finish the word. The player who finishes the word is the loser. It is inevitable that someone will lose, for there are no infinite words, and as the number of letters increases, the players' options for altering the course of the game, i.e., their options for changing the word in formation, decrease: the part of the word which is already formed determines the remaining part more and more definitely. At the end of every word (after the last letter), there is always an 'upswing' of indefiniteness, for the new word can begin with practically any letter of the alphabet. Word boundaries are thus associated with 'upswings' (peaks) of indefiniteness.[6]

A very similar conception underlies Harris's morphological algorithm, which functions in the following manner. One sentence is picked out of a text in phonemic transcription, e.g., English [hiyzklevər][7]. All the other sentences which begin with the same phoneme [h] are then sought out, and the different phonemes which follow [h] in these sentences are counted up. These phonemes are considered 'successors' to [h]. The number of 'successors' to [h] is nine. Then all the sentences which begin with the same two phonemes as the given sentence, i.e., [hi], are sought out, the 'successors' to this pair of phonemes are also counted up (there are fourteen of them), and so on. The number of 'successors' varies at different points in the sentence: sometimes it climbs up, forming peaks, and sometimes it drops. If phonemic variation

[6] These observations could be presented also in information-theory terminology, as some readers might have realized. It is noteworthy that entropy does indeed figure in some computer experiments, as we shall see.

[7] The transcription is Harris's.

at morpheme junctures is indeed greater than within morphemes, then the peaks must occur at morpheme boundaries; cf.

h	i	y	z	k	l	e	v	ə	r
9	14	29	29	11	7	...	...	...	...

In the given sentence there are two peaks – after the phoneme [y] (29), and after the phoneme [z] (29). These are indeed the points at which morphological boundaries should be drawn. The resulting segmentation (*he/'s/clever*) reflects the actual morphological composition of the sentence. It is interesting that for the sentence *he's quicker*, which seems to be very similar to *he's clever*, the segmentation obtained in this way is different, and also correct: following the phoneme [k] there is another peak, and a third boundary is thus drawn before the comparative morpheme *-er*.

To avoid possible errors several additional rules are required, one of which is the following: what must be counted up is not only the number of 'successors' to phonemes from the beginning of a sentence to its end, but also the number of 'predecessors', from the end of the sentence to its beginning.

In conclusion let us refer to the results of computer experiments ⟨12, *141*⟩ designed to test the method of morphological segmentation proposed by Harris. The experiments were conducted on one hundred English sentences in phonemic transcription, and the results were quite good: in 85 % of the cases the computer's segmentation of the text was correct.

As stated above, an algorithm of this type isolates morphs rather then morphemes, generally speaking, and the latter have to be established on the basis of some other principles. Harris's method of morphological analysis also involves other difficulties, many of which were diagnosed by Harris himself (the method has no procedure for discovering 'discontinuous' morphemes and other nonsegmental morphemes, suprasegmental morphemes, and so on). We would like to point out an additional difficulty, which stems from the fact that there are also various kinds of fused and superposed morphemes.

One of the principal premises of Harris's method and of similar methods is the assumption that morphemes DO NOT INTERSECT in texts, that is, the last part of a morpheme cannot be the beginning of another (as in chain words). This is true in most cases, but the situation on the whole is more complex. Quite instructive in this respect is what happens in Russian, as described by Zemskaja under the general heading of 'superposed morphemes' (an instance of haplology) ⟨67⟩. In the examples analyzed by Zemskaja, morphemes (or rather morphs) are superposed when the last phonemes of a stem coincide with the first phonemes of a following derivational suffix; cf. *derbist* 'jockey' and *taksist* 'taxi driver' versus *futbol-ist* 'football player': in *derbist* and *taksist* the vowel *i* belongs to the stem as well as to the suffix (cf. the 'proper' but inexistent forms *derbi-ist* and *taksi-ist*). Other examples of this type are the words *lilovatyj* 'lilaceous' (versus *čern-ovatyj* 'blackish'), *Merin* 'Mary's' (versus *Kat-in* 'Katja's'), and *sal'erizm* 'Salierism' (versus *vagner-izm* 'Wagnerism').

In these cases and in similar ones, Harris's algorithm will err. However, there are methods of correction to eliminate such errors.

The morphological segmentation of a given text is only the first step, though the most important one, in discovering the grammar of the text. It is also necessary to at least (1) distinguish between lexical morphemes (roots) and grammatical morphemes; (2) establish classes of roots (e.g., stems of nonderivative nouns, stems of nonderivative verbs, stems of nonderivative adjectives, and so on) and classes of grammatical morphemes (paradigms); (3) learn to identify word forms and classes of equivalent word forms; and (4) learn to identify sentence boundaries and the rules governing the syntactic relations between the word forms in sentences. Obviously, every one of these tasks requires a special algorithm.

We shall describe here only one algorithm, developed by Suxotin, for determining the SYNTACTIC RELATIONS between word forms in sentences ⟨174⟩. Before describing this algorithm, we shall indicate in general terms how lexical morphemes can be distinguished from grammatical morphemes. Alphonse Juilland has been particularly interested in this problem ⟨323⟩. He studied two numerical parameters of morphemes: (1) their frequency in texts, and (2) their distribution in texts of different genres. He came to the conclusion that what characterizes grammatical morphemes is a high frequency of occurrence and an even distribution in different texts, whereas lexical morphemes are characterized by a low frequency and an uneven distribution, particularly characteristic of terms. These observations seem fairly obvious: the nominative and accusative case morphemes, for instance, occur at more or less the same frequency, which is quite high, in texts of different genres, while the root morphemes of words such as *integral* 'integral', *logarifm* 'logarithm', or *sinus* 'sinus' are not only considerably less frequent but also unevenly distributed, as they occur chiefly in mathematical texts. The parameters studied by Juilland can thus serve as a basis for favorability functions to solve the given problem.

Another way of solving this problem was indicated by Suxotin. He observed that the behaviour of grammatical and lexical morphemes is in certain respects similar to that of vowels and consonants: they alternate in texts (a group of lexical morphemes is followed by a group of grammatical morphemes, and vice versa), and the most frequent morpheme is always a grammatical morpheme. The class of lexical morphemes and the class of grammatical morphemes in a set of morphemes can thus be discovered by the algorithm described above for distinguishing between vowels and consonants (or by a variant of this algorithm, adapted to the special nature of the morphological material).

Similar ideas were expressed by Knorozov and were applied in the decipherment of unknown writing systems at the USSR Academy of Sciences' All-Union Institute of Scientific and Technical Information. According to Knorozov, the class of lexical morphemes consists of a LARGE number of units, and each one combines with a SMALL number of units from the other class – the class of grammatical morphemes. The latter consists of a SMALL number of units, and each combines with a large number of units from the class of lexical morphemes.

Let us turn now to the algorithm for determining the syntactic relations between word forms in sentences. The algorithm consists of two parts, the first of which is a research model in the true sense of the word: its output is the elementary syntactic rules of the given language. In the second part of the model, which cannot be considered a research model, properly speaking, the syntactic structure of sentences is analyzed by means of these rules and represented in the form of trees.

The algorithm is applied to a text in which all the word forms have been replaced by (recoded into) symbols of word-form classes, on the basis of previously given data. The following symbols are used: (1) N_n – a noun or personal pronoun in the nominative case; (2) N_g, N_d, N_a, N_i, N_p – a noun or personal pronoun in the genitive, dative, accusative, instrumental, and prepositional cases; (3) A_n, A_g, A_d, ... – an adjective or its equivalent in the nominative, genitive, dative, etc.; (4) V – a finite verb; (5) P – a preposition; and (6) C – a conjunction, etc.

Accordingly, the word forms of the sentence *V okna brezžil sinevatyj xolodnyj ↄvet utra*; *pod lavkoj šipel i krjakal proↄnuvšijsja selezen*' 'in the windows glimmered the azure cold light of morning; under the bench hissed and quacked the awakened drake' can be recoded into $PN_aVA_nA_nN_nN_g$; $PN_iVCVA_nN_n$. The algorithm establishes the relationships between the word forms in simple sentences, so that before it can be applied, some other algorithm must break up the complex sentences into simple sentences with no vocatives, parenthetic words, and the like. For every simple sentence we must obtain a diagram like the ones we used to draw up in school. The relationships in the sentence *V okna brezžil sinevatyj xolodnyj svet utra* 'in the windows glimmered the azure cold light of morning', for instance, must be diagrammed as in Figure 5.

The algorithm formalizes the following substantive hypothesis on the nature of syntactic relations: if there is a direct syntactic relation between two word forms, then these word forms belong

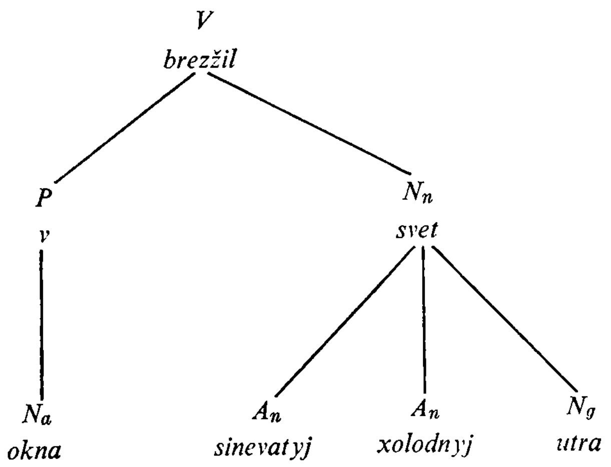

Fig. 5.

to word-form classes the elements of which frequently occur in the same sentence. The order in which the word forms occur in sentences is considered immaterial: word forms which occur one next to the other are not necessarily related to each other syntactically, and word forms which occur at a considerable distance from one another may still be related to each other syntactically.

We shall use the concepts of graph theory (introduced on pp. 127-128) to explain the operation of this algorithm. The sentence diagram in Figure 5 is obviously a tree. The operation of the algorithm must thus produce a tree representation for every sentence – a connected graph with no cycles. The graph must be connected, because every word form in any simple sentence is directly related syntactically to at least one other word form.[8] The requirement that the graph contain no cycles seems less obvious: in sentences with a subjective or objective predicative nominal, the form of the predicative depends both on the verb (by government) and on the subject or the object (by agreement). More specifically, its case is

[8] Sentence adverbials seem to constitute counterexamples to this proposition. However, a sentence adverbial can be regarded as related to the fulcrum of the sentence, i.e., the predicate.

determined by the verb, its gender and number – by the subject or object noun; cf. *Oni stali vzroslymi* 'they became adults (instr. pl.)', *On stal vzroslym* 'he became an adult (instr. masc. sing.)', and *Ona stala vzrosloj* 'she became an adult (instr. fem. sing.)'; or *On ščitaet ee beznravstvennoj* 'he considers her immoral (instr. fem.)' versus *Ona ščitaet ego beznravstvennym* 'she considers him immoral (instr. masc.)'; and so on. A graph representing government as well as agreement relations must inevitably contain cycles, cf. *Oni stali vzroslymi.* However, for several substantial reasons which are not directly related to our topic, agreement relations are not indicated in such cases.

The algorithm operates on a sentence representation in the form of a complete graph (Figure 6). Its task is to remove from the graph

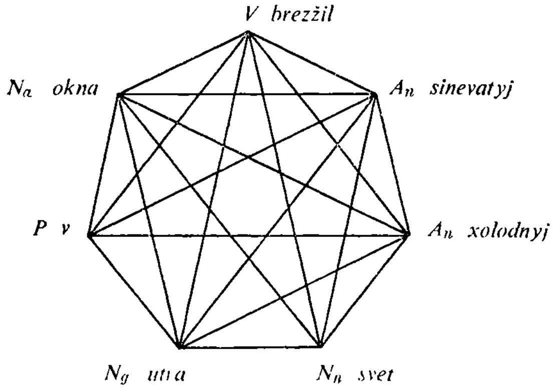

Fig. 6.

all the 'extra' sides, leaving in it only the sides which represent direct syntactic relations between the word forms of the given sentence. In other words, its task is to choose the best solution from the set of possible solutions (the complete graph). The task of the algorithm can be compared to that of a filter which receives 'input' which includes a great amount of unnecessary materials and produces 'output' which contains only desirable materials.

As stated above, the algorithm operates in two stages. In the first stage it produces a 'grammar' for the given text in the form of a square table. The columns and lines in the table represent classes of word forms (see Table 4).

TABLE 4

	N_n	N_g	N_a	A_n	V	P	...
N_n		1	1	1	1	1	...
N_g	1		1	1	1	1	...
N_a	1	1		1	1	1	...
A_n	1	1	1		1	1	...
V	1	1	1	1		1	...
P	1	1	1	1	1		...
...	...	...	...	...	...	...	...

The table is filled out in the following manner (this is a simplified version): the first sentence of the text is taken out and every possible pair of word-forms in it is noted down. In our sentence the pairs are [*v* 'in', *okna* 'windows'], [*v* 'in,' *brezžil* 'glimmered'], [*v* 'in', *sinevatyj* 'azure'], [*v* 'in', *xolodnyj* 'cold'], [*v* 'in', *svet* 'light'), [*v* 'in', *utra* 'of morning'], [*okna* 'windows', *v* 'in'], [*okna* 'windows', *brezžil* 'glimmered'], [*okna* 'windows', *sinevatyj* 'azure'], [*okna* 'windows', *xolodnyj* 'cold'], [*okna* 'windows', *svet* 'light'], [*okna* 'windows', *utra* 'of morning'], and so on, altogether thirty pairs. Then the line which corresponds to the first word form in a given pair and the column which corresponds to the second word form in the same pair are sought out. For the first pair these are line P and column N_a, for the second – the same line and column V, for the third – the same line and column A_n ,and so on. The digit 1 is written in the square where the corresponding line and column intersect, regardless of whether the given pair of word forms occurs once or more than once in the given sentence. After the first sentence is processed by these rules (the results of which are shown in Table 4), the second sentence is processed in an analogous manner, and then the remaining sentences to the end of the text for which a grammar must be produced. When all the sentences have been processed, the squares in the table are full of digits, with some squares containing more digits than others. It is obvious, for instance, that in the square N_nV there will be more digits than in the square A_nV, for when a sentence contains a finite verb, the probability of there being a noun in the nominative case in the same sentence is quite high, whereas it is quite possible that the sentence contains no adjective in the nominative case; cf. *Časy b'jut* 'the clock strikes', *Brezžit den'* 'day is breaking', *Vojut sireny* 'sirens wail', and so on. In the square N_nA_n there will be more entries than in the square N_nN_a, for when a sentence contains an adjective in the nominative case, it almost always contains also a noun in the nominative case, whereas the presence of a noun in the accusative case does not presuppose a noun in the nominative case; cf. *Dlinnyj sostav medlenno polz po ravnine* 'the long train slowly crept along the plain'; *Jarkoe solnce lenivo podymalos' nad lesom* 'a bright sun idly rose over the forest' and *Pročitaj etu knigu* 'read this book'; *pšenicu pobilo gradom* 'the wheat (acc.) was hit (neut.) by hail (instr.)'; *Ego znobilo* 'he (acc.) was shivering (neut)'.

The first stage, then, consists of the DISCOVERY of the grammar of the given language on the basis of the text data: the resulting table contains, in a probabilistic form, all the rules for the relations between word forms in sentences. In the second stage of the operation of the algorithm, we return to the first sentence of the text with this probabilistic grammar, draw a complete graph of this sentence (Figure 6), and write at each side of the graph the figure from the corresponding square in the table (since the table is symmetric in relation to the main diagonal, it makes no difference whether we take the figure from the square XY or YX). Then we begin to remove from the graph the 'extra' sides. The first side to be removed is the side with the lowest figure. At every subsequent step another 'minimal' side is removed, but not if its removal would produce a disconnected graph. The sides are removed until the graph becomes a tree. This is the representation we have been seeking. The principles of the operation of the algorithm can be illustrated by the following hypothetical example (Table 5 and Figure 7).

TABLE 5

	a	*b*	*c*	*d*	*e*	...
a		90	35	50	29	...
b	90		80	30	40	...
c	35	80		78	31	...
d	50	30	78		45	...
e	29	40	31	45		...
...	...	...	...	...	...	...

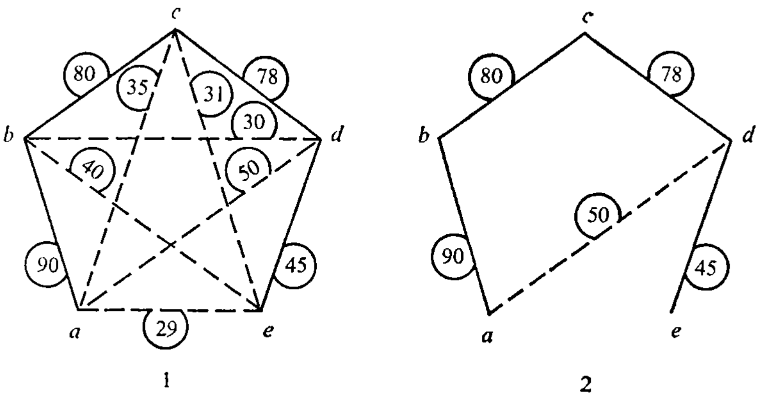

Fig. 7.

First, in accordance with the first rule, we remove side *ae*, which is the 'minimal' one (29), and then, in accordance with the same rule, we remove sides *bd* (30), *ce* (31), *ac* (35), and *be* (40), in the given order. Then (Figure 7, [2]) we remove side *ad* (50) instead of *de* (45), although 50 is greater than 45, because without side *de* the graph would have become disconnected and vertex *e* would have been isolated. The remaining sides, drawn in solid lines, form the desired tree (Figure 7, [2]).

Trees representing the syntactic relations within the remaining sentences in the given text are obtained in the same manner, and the operation of the algorithm then terminates.

Data on the direct syntactic relations between word forms, assuming that a morphological analysis of the text has already been carried out, can supply important information on the lexical aspect of the given language and can deepen our understanding of its syntax. Mel'čuk, for instance, has shown how so-called set phrases can be recognized in such conditions ⟨112⟩, and Suxotin has proposed a very simple method for converting information on the direct syntactic relations between word forms into information on dependency relations, or subordination (a method for orienting the graphs of sentences), and also methods for obtaining from the information on dependency relations information on the different types of dependency relations (agreement, government, and parataxis). Moreover, with no additional data we can learn from the information provided by Suxotin's algorithm described above interesting facts on the meanings of words in various languages. This last question will be discussed in greater detail now, and we shall also present some of the results of our study of the meanings of Russian verbs on the basis of their syntactic properties.[9]

The logical structure of models of the type we are concerned with and their operational aspects should be fairly clear by now. We shall therefore concentrate on the substance of the models in the following discussion. As for their formal aspects, we shall describe in a more or less detailed fashion only one algorithm, and of the formal concepts we shall mention only those which

[9] Generally speaking, our model belongs to the class of experimental rather than decipherment models. However, within a certain framework and for certain materials it can also function as a decipherment model (cf. ⟨8⟩). An interesting attempt to describe the meaning of words on the basis of their grammatical and combinatorial properties was made by Šajkevič ⟨203⟩.

are absolutely necessary, in the first place the concept of government and the concept of the compatibility of subordinate forms.

The concept of GOVERNMENT can be developed by an algorithm operating with data on the words of some given language, data on their syntactic interdependencies within sentences, and data on the stems and the grammatical forms of the words. Government can then be regarded, as it sometimes is, as the power of the stem of a head word, usually a verb, to determine the form (or forms) of a dependent word (or words); cf. *bojat'sja čego* 'to fear something (gen.)', *upvraljat' čem* 'to manage something (instr.)', *zabotit'-sja o kom* 'to take care of someone (prep.)', *darit'komu čto* 'to give to someone (dat.) something (acc.)', *trebovat' čego ot kogo* 'to demand something (gen.) from someone (gen.)', *obespečivat' kogo čem* 'to provide someone (acc.) with something (instr.)', *vmenjat' čto komu vo čto* 'to charge someone (dat.) with something (acc.) for something (gen.)'.

In principle verbs can govern more than one form, but when they do, they do not govern the different forms with the same strength. The STRENGTH of government can be defined rigorously as the conditional probability that a certain form or set of forms will occur in a text given the occurrence of a certain verb in the same text ⟨74, 7⟩. On the basis of various formal criteria ⟨7⟩ it is possible to establish in the set of forms governed by a given verb the subset of forms which are governed fairly strongly. The latter belong with the major syntactic properties of the given verb.

There is a close relationship between the syntactic properties of a verb and its semantic features. This is demonstrated by the fact that when verbs are alike in their syntactic properties, they are also alike, as a rule, in their semantic features, and when verbs differ in their syntactic properties, they also differ in their semantic features. Such a relationship develops without fail in every standardized language which avoids form duplication and utilizes its resources in an increasingly consistent and uniform manner, as a result of the operation of analogy, which leads to the levelling of the syntactic properties of words that are semantically close to each other. Such analogy accounts for the present-day government of verbs such as, for instance, *nabljudat'* 'to watch' (*za kem-libo* 'after someone [instr.]'), *rukovodit'* 'to guide' (*kem-libo* 'someone [instr.]'), and *trepetat'* 'to tremble' (*pered kem-libo* 'before someone [instr.]'), which in the nineteenth century and even in the early part of the twentieth century could govern the forms *nad* 'over' N_i, N_a, and N_g, respectively; cf. *nabljudat' nad raznymi rabotami* 'to watch over various projects (instr.)' (Aksakov), *rukovodit' čitatelej* 'to guide the readers (acc.)' (L. Tolstoy), *trepetat' muža* 'to tremble (before one's) husband (gen.)' (Chekhov). These changes in government have come about through the pressure of the indisputably dominant types *gljadet'* 'to look' (*nadzirat'* 'to supervise', *sledit'* 'to follow', *smotret'* 'to look') *za kem-libo* 'after someone (instr.)'; *komandovat'* 'to command' (*pravit'* 'to govern', *rasporjažat'sja* 'to manage', *upravljat'* 'to run') *kem-libo* 'someone (instr.)'; and *drožat'* 'to tremble' (*trjastis'* 'to shake') *pered kem-libo* 'before someone (instr.)'. The fact that the syntactic resources of languages are utilized in a consistent and uniform manner makes it possible to determine unerringly the semantic class of Russian verbs which can govern, for instance, the forms *kogo čem po čemu* 'someone (acc.) with something (instr.) on something (dat.)' (*bit'* 'to beat', *kolotit'* 'to hit', *udarjat'* 'to strike'); *komu čto o čem* 'to someone (dat.) something (acc.) about something (prep.)' (*govorit'* 'to tell', *rasskazyvat'* 'to tell'); *komu ot kogo za čto* 'to someone (dat.) from someone (gen.) for something (acc.)' (*vletelo* 'it came', *dostalos'* 'it came', *popalo* 'it came'); and so on.

Observations on similarities and differences in the meanings of verbs on the basis of similarities and differences in their syntactic properties can apply either to the set of uses of a single verb or to the set of uses of all the verbs in a given language. In the first case one is concerned with distinguishing between the different meanings of a given verb, in the second with establishing semantic classes of verb meanings (hereafter verb meanings will be referred to simply as verbs). Both of these tasks can be accomplished by the same method; only the material to which the method applies will be different.

The chief method for establishing semantic similarities and differences in the set of uses of a given verb is by applying the principle of COMPATIBILITY, which can be defined as follows: two subordinate forms are compatible with regard to a given verb if they are realized with this verb simultaneously in at least one sentence. By this definition the forms N_d, *na* 'on' N_a, and N_i are compatible with regard to the verb *otvečat'* 'answer', cf. *On otvetil drugu na predloženie* 'he answered (his) friend (dat.) concerning the proposal (acc.)', *On otvetil drugu soglasiem* 'he answered (his) friend (dat.) with consent (instr.)', *On otvetil na predloženie soglasiem* 'he answered the proposal (acc.) with consent (instr.)', and even *On otvetil drugu na predloženie soglasiem* 'he answered (his) friend (dat.) concerning the proposal (acc.) with consent (instr.)'.

The compatibility of subordinate forms with regard to a given verb is evidence of the same meaning, whereas their incompatibility indicates a semantic difference; the exceptions to this rule are few and can be handled in a formal fashion. This principle makes it possible to distinguish between the meanings of the verb *otvečat'* 'answer' in the sentences *On otvečaet drugu* 'he answers (his) friend (dat.)' and *Rabota otvečaet vsem trebovanijam* 'the work answers all the requirements (dat.)', since in the second sentence the form in the dative case is incompatible with the forms *na* 'on' $N_a N_i$. The different meanings of the verbs *bit'*, *bit'sja*, *gladit'*, *mešat'*, *otkryvat'sja*, *razvodit'*, *risovat'sja*, *trjasti*, and numerous others can be distinguished in the same way; cf. *bit' lošad' knutom po krupu* 'to beat a horse (acc.) with a whip (instr.) on (his) crupper (dat.)' and *bit' zorju* 'to sound reveille (acc.)'; *bit'sja golovoj o stenu* 'to hit one's head (instr.) against the wall (acc.)', *bitsja s bratom na kulakax* 'to fight with one's brother (instr.) with fists (prep.)', and *bit'sja nad zadačej* 'to struggle over a problem (instr.)'; *gladit' dočeri lob ladon'ju* 'to stroke the forehead (acc.) of one's daughter (dat.) with one's hand (instr.)' and *gladit' bel'e utjugom* 'to iron linen (acc.) with an iron (instr.)'; *mešat' vodu s vinom* 'to mix water (acc.) with wine (instr.)' and *mešat' komu-libo rabotat'* (*v rabote*) 'to disturb someone (dat.) working (infinitive) (at [his] work [prep.])'; *otkryvat'sja drugu* (*pered drugom*) *vo vsem* 'to reveal to one's friend (dat.) (before one's friend [instr.]) everything (prep.)' and *otkryvat'sja na ulicu* 'to open to the street (acc.)' (of a door, for instance); *razvodit' muža s ženoj* 'to separate between husband (acc.) and wife (instr.)' and *razvodit' rukami* 'to spread one's hands (instr.)'; *risovat'sja pered vsemi svoej vyderžkoj* 'to show off one's self-control (instr.) in front of everyone (instr.)'. *risovat'sja komu-libo v voobraženii* 'to be pictured in someone's (dat.) imagination (prep.)', and *risovat'-sja na blednom nebosklone* 'to be silhouetted on the pale horizon (prep.)'; *trjasti mal'čika rukoj za plečo* 'to shake a boy (acc.) with one's hand (instr.) by (his) shoulder (acc.)' and *trjasti golovoj* 'to shake one's head (instr.)'.

Let us extend the application of the compatibility principle now to the vocabulary as a whole and treat compatible subordinate forms, even if they are compatible with regard to just one verb, as distinctive syntactic features marking different verb classes. Verbs with the same syntactic features will be assigned to the same class. On this basis we can get classes such as, for instance: *bit'* 'beat' (*kolotit'* 'hit', *molotit'* 'thresh', *udarjat'* 'strike') *kogo čem po čemu* 'someone (acc.) with something (instr.) on something (dat.)'; *bit'sja* 'hit' (*kolotit'sja* 'bang', *stukat'sja* 'knock', *udarjat'sja* 'strike') *čem obo čto* 'with something (instr.) against something (acc.)'; *gladit'* 'stroke' (*trepat'* 'pat') *komu čto čem* someone (dat.) something (acc.) with something (instr.)'; *mešat'* 'disturb' (*pomogat'* 'help') *komu rabotat'* (*v rabote*) 'someone (dat.) working (infinitive) (at [his] work [prep.])'; *mešat'* 'mix' (*putat'* 'mix up', *razvodit'* 'separate', *razlučat'* 'separate', *soedinjat'* 'unite', *sravnivat'* 'compare') *čto s čem* 'something (acc.) with something (instr.)'; *otkryvat'sja* 'reveal' (*vinit'sja* 'blame oneself', *ispovedovat'sja* 'confess', *kajat'sja* 'repent', *priznavat'sja* 'admit', *soznavat'sja* 'admit') *komu* (*pered kem*) *v čem* 'to someone (dat.) (before someone [instr.]) something (prep.)'; *risovat'sja* 'show off' (*gordit'sja* 'be proud of', *kičit'sja* 'boast', *xvastat'sja* 'boast', *ščegoljat'* 'show off') *čem pered kem* 'of something (instr.) before someone (instr.)'; *trjasti* 'shake' (*dergat'* 'pull', *trogat'* 'touch', *tormošit'* 'tug', *xvatat'* 'seize') *kogo čem za čto* 'someone (acc.) with something (instr.) by something (acc.)'; and others. The semantic homogeneity of these classes is so obvious that no additional comments are necessary.

It was stated above that each class in this classification must consist of verbs having the same syntactic features. However, the number of verbs with perfectly matching features is rather small, and if we were to follow the stated principle to the letter, many of the classes would have consisted of no more than one element. In practice, verbs are assigned to the same class on the basis of relative rather than absolute identity in their syntactic features. But this weaker condition poses a much more serious problem: using absolute identity as a criterion is much easier than using relative identity. In order to determine the latter one must be able to measure it. Semantic similarities between verbs can be thought of in terms of the distance between them in a semantic space, where the verbs which are relatively close to each other are in the same class (and are most similar in their meaning).

The problem confronting us can thus be formulated as a mathematical problem and can be solved in a rigorous fashion. First the distances between the elements in the set of verbs must be measured and tabulated (this can be done on the basis of the data on their syntactic features), then a procedure must be found to establish verb classes on the basis of the tabulated distances. Consequently, the algorithm for solving this problem has two parts. In the first part the data on the government of the verbs (obtained by the methods described above) are converted into data on the DISTANCES between them. In the second part the data on the distances between the verbs are converted into data on their CLASSES.

In mathematics distance (metrical distance) is a positive number ρ associated with two elements a and b which belong to the same set and have the following properties: (1) the distance of each of the elements from itself is null: $\rho\ (a, a) = 0$; (2) the distance of element a from element b is equal to the distance of element b from element a: $\rho\ (a, b) = \rho\ (b, a)$; (3) the distance of a from b plus the distance of b from a third element c is greater than or equal to the distance of a from c: $\rho\ (a, b) + \rho\ (b, c) \geqslant \rho\ (a, c)$. When such a number can be associated with any two elements in a given set, the set is called METRICAL SPACE.

Data on the distances between elements in a given set are easily obtainable when there is a system of n coordinate lines (features) on which every element has a certain value. If $f_i\ (a)$ is the numerical value of element a on the coordinate line i, then the distance between any two elements a and b can be calculated by the formula:

$$\rho\ (a, b) = \sum_{i=1}^{n} |f_i(a) - f_i(b)|,$$

where Σ symbolizes sum and the vertical bars denote the absolute value of the enclosed numbers. The formula can be read as follows: the distance between two elements equals the sum of absolute values of the differences between the numerical values of these elements on all of the coordinate lines, from the first one ($i = 1$) to the nth (the letter over the sum symbol).

Let us return to our problem. The elements in our set are verbs, of course, and the coordinate lines are subordinate forms. Each verb has a certain numerical value on every one of the coordinate lines. The numerical value represents the probability of occurrence of the given form in a text, given the occurrence of the given verb. Let us present the situation in the form of a table (Table 6; the data in the table are drawn from actual texts, but should be regarded as no more than illustrative; see ⟨8⟩ for a more comprehensive and precise treatment of the problem).

By means of the formula given above, the table can be converted into a table of distances (metrical space) such as Table 7.

The squares below the main diagonal need not be filled out, since the table is symmetric. To assist the reader in mastering the simple mathematical computation of the distances on the basis of data as in Table 6, we shall compute here the distance between the verbs *besedovat'* 'talk' and *bojat'sja* 'fear':

$$\rho\ (besedovat', \ bojat'sja) = \sum_{i=1}^{8} |f_i(besedovat') - f_i(bojat'sja)|$$

TABLE 6

	N_g	N_a	*c* 'with' N_i	*o* 'about' N_p	N_g *ot* 'from' N_g	N_a *dlja* 'for' N_g	N_a *iz* 'from' N_g	N_a *v* 'in' N_p
1. *besedovat'* 'talk'	0	0	0,82	0,18	0	0	0	0
2. *bojat'sja* 'fear'	0,29	0	0	0	0	0	0	0
3. *ždat'* 'wait'	0,24	0,36	0	0	0,02	0	0	0
4. *izmenit'* 'alter'	0	0,86	0	0	0	0	0	0,14
5. *ispugat'sja* 'be frightened'	0,10	0	0	0	0	0	0	0
6. *naladit'* 'fix'	0	0,80	0	0	0	0	0	0,10
7. *ožidat'* 'expect'	0,27	0,48	0	0	0,03	0	0	0
8. *postroit'* 'build'	0	0,88	0	0	0	0,07	0	0
9. *proizvodit'* 'produce'	0	0,88	0	0	0	0,04	0	0
10. *razgovarivat'* 'talk'	0	0	0,54	0,08	0	0	0	0
11. *sozdavat'* 'create'	0	0,90	0	0	0	0	0,03	0

TABLE 7

	besedovat'	*bojat'sja*	*ždat'*	*izmenit'*	*ispugat'sja*	*naladit'*	*ožidat'*	*postroit'*	*proizvodit'*	*razgovarivat'*	*sozdavat'*
1. *besedovat'* 'talk'	0	1,29	1,62	2,00	1,10	1,90	1,78	1,95	1,92	0,38	1,93
2. *bojat'sja* 'fear'		0	0,43	1,29	0,19	1,19	0,53	1,24	1,21	0,91	1,22
3. *ždat'* 'wait'			0	0,90	0,52	0,80	0,16	0,85	0,82	1,24	0,83
4. *izmenit'* 'alter'				0	1,10	0,10	0,82	0,23	0,20	1,62	0,21
5. *ispugat'sja* 'be frightened'					0	1,00	0,68	1,05	1,02	0,72	1,03
6. *naladit'* 'fix'						0	0,72	0,25	0,22	1,52	0,23
7. *ožidat'* 'expect'							0	0,77	0,74	1,40	0,75
8. *postroit'* 'build'								0	0,03	1,57	0,12
9. *proizvodit'* 'produce'									0	1,54	0,09
10. *razgovarivat'* 'talk'										0	1,55
11. *sozdavat'* 'create'											0

$$= |0 - 0.29| + |0 - 0| + |0.82 - 0| + |0.18 - 0| + |0 - 0| + |0 - 0| + |0 - 0| + |0 - 0| = 0.29 + 0.82 + 0.18 = 1.29.$$

The first part of the operation of the algorithm ends with the completion of the table of distances. In the second part classes are formed on the basis of this table, through a formalization of the following substantive hypothesis: a class is a set of dots located at a short distance from one another (forming a cluster in the metrical space). The algorithm observes the following conditions in its operation:

(1) Given any three elements *a*, *b*, and *c*, if *a* and *b* belong to the class K_i and *c* does not belong to this class, then ρ (*a*, *b*) must be smaller than ρ (*a*, *c*). It can be demonstrated that when this condition is observed, the given set is partitioned into nonintersecting classes (excluding cases where a class is totally included in another class).

(2) The number of classes in the partition must be as small as possible, but no less than two. It can be demonstrated that if this second condition is also observed, the resulting partition is a unique one (no other partition possesses the two properties in conjunction). This is the partition we have been seeking.

The substance of the algorithm is as follows:

1. In the first step we find the two elements most distant from each other in Table 7. They cannot belong to the same class by our conditions, and we therefore establish two different classes: {*besedovat'* 'talk', ...} and {*izmenit'* 'alter', ...}.

2. We add to each class one element at a time, always choosing from the table the verb closest to an element already assigned to a class. Whenever a new element is assigned to one of the classes, we have to check whether this does not violate the first condition. If the condition is not violated, then we repeat the second step. It it is violated for some element in one of the classes, then the violating element is withdrawn from the class and a new class is formed. According to these rules, we add to the second class the verbs *naladit'* 'to fix', *proizvodit'* 'to produce', *sozdavat'* 'to create', and *postroit'* 'to build', and then add to the first class the verb

razgovarivat' 'to talk' (in this order). Let us try to assign to the class {*besedovat'* 'to talk', *razgovarivat'* 'to talk'} the verb *ispugat'-sja* 'to be frightened', for this verb, as yet unclassified, is the closest (0.72) to an element (the verb *razgovarivat'* 'to talk') which has already been assigned to a class. We find that this cannot be done, as ρ (*ispugat'sja* 'to be frightened', *besedovat'* 'to talk') $= 1.10 >$ ρ (*ispugat'sja* 'to be frightened', *proizvodit'* 'to produce') $= 1.02$. The first condition is violated, and the verb *ispugat'sja* 'to be frightened' must therefore be assigned to a new class {*ispugat'sja* 'to be frightened', ...}.

3. If the formation of a new class does not result in a violation of the first condition for any of the elements already classified, then the second step is repeated (this is what happens in our case: the third class absorbs all of the remaining verbs, namely, the verbs *ispugat'sja* 'to be frightened', *bojat'sja* 'to fear', *ždat'* 'to wait', and *ožidat'* 'to expect'). If there is a violation, then the violating elements are withdrawn one by one from the classes already set up and are returned to the pool of unclassified elements. This goes on until there is no violation, and the second step is then repeated.

This algorithm calls for two comments. First, it is noteworthy that its application indeed produces the desired partition, that is, a partition of the set of verbs into classes satisfying both the first and the second condition. This can be demonstrated in a formal manner. Secondly, the resulting classes are quite homogeneous semantically. The three classes set up in our example, for instance, undoubtedly possess this property; cf. (1) *besedovat'* 'to talk', *razgovarivat'* 'to talk'; (2) *izmenit'* 'to alter', *naladit'* 'to fix', *proizvodit'* 'to produce', *sozdavat'* 'to create', *postroit'* 'to build'; (3) *ispugat'sja* 'to be frightened', *bojat'sja* 'to fear', *ždat'* 'to wait', *ožidat'* 'to expect'. The syntactic features of the verbs in the last two classes differ not only in their number but also in their substance. By applying the algorithm once more to the material in each of these classes we can therefore obtain an even finer and more accurate partition of the set: (2a) *izmenit'* 'to alter', *naladit'* 'to fix'; (2b) *proizvodit'* 'to produce', *sozdavat'* 'to create', *postroit'* 'to build'; (3a) *ispugat'sja* 'to be frightened', *bojat'sja* 'to fear'; (3b) *ždat'* 'to wait', *ožidat'* 'to expect'.

The results the author of the present book obtained from experiments with more extensive materials ⟨8⟩ were quite good. One can therefore expect semantically significant classifications on the basis of syntactic data. If there are also words which can be associated with objects and phenomena in the real world, then it is possible, in principle, not only to establish similarities and differences between words but also to assign definite semantic features to the words.

In order to see how this can be realized, let us examine Ščerba's classical sentence *Glokaja kuzdra šteko budlanula bokra i kudrjačit bokrënka*, which we have already cited in another context (p. 114). This sentence is usually used to illustrate the fact that one can analyze the formal, grammatical structure of a sentence without knowing the lexical meaning of the words in it. It seems to us that an even stronger claim can be made on the basis of this sentence: if one knows the language of the sentence, one can not only analyze its grammatical structure, but also define fairly accurately the meaning of the words it contains.

(1) *budlanut'*-a verb signifying a FORCEFUL, ENERGETIC action on some object, something like *udarit'* 'strike'. This verb is indeed transitive, as it has a direct object *(bokra)*, which is an animate noun (cf. the ending *-a* in the accusative case), and transitive, semelfactive verbs in *-anut'* have this meaning in Russian; cf. *davanut'*, *dolbanut'*, *zvezdanut'*, *mazanut'*, *rezanut'*, *rubanut'*, *sadanut'*, *steganut'*, *tolkanut'*, *trepanut'*, *trjaxanut'*, *xlestanut'*, *šibanut'*, *ščelkanut'*, *ščipanut'* (cf. Vinogradov ⟨30, *529-530*⟩). (The verb *skazanut'* is an exception, though it cannot be identified semantically with *skazat'* 'to say', because it takes an animate noun as direct object, and one can only say SOMETHING, not SOMEONE.)

(2) *Kudrjačit'*-a verb with the same meaning of forceful action on an object, or with the opposite meaning of *laskat'* 'caress' (cf. ⟨152, *67*⟩). This assertion hinges on the following facts: *kudrjačit'* and *budlanut'* have the same syntactic properties: they are both transitive verbs and both have an animate noun as their

direct object. Furthermore, in the given sentence they are coordinated, linked by the coordinative conjunction *i* 'and'. Coordinated constituents which have the same syntactic properties and are linked by the conjunction *i* 'and' (but not necessarily by other conjunctions) tend to have the same semantic features; cf. the ill-formed sentences *On trjaxanul menja i vidit moego brata* 'he gave me a jolt and sees my brother' and *On trjaxanul menja i blagodarit moego brata* 'he gave me a jolt and thanks my brother', where the predicates apparently do not have any semantic features in common besides transitivity.

(3) *Šteko* – an adverb the meaning of which includes the feature INTENSITY (something like *krepko* 'firmly, soundly', or *kak sleduet* 'properly', although the opposite meaning on the same scale, *slegka* 'lightly', is also possible). This assertion hinges first of all on the fact that *šteko* derives from the adjective *štekij* (cf. *dikij* 'wild' – *diko*, *krepkij* 'strong, firm' – *krepko*, *šibkij* 'fast' – *šibko*, etc.), and therefore it cannot be an adverb of place, time, purpose, cause, and so on. Secondly, it modifies a verb which signifies an intense, forceful action on an object, and adverbs of manner which qualify verbs of this semantic class must indeed express intensity; cf. the unacceptability of a sentence such as *Voznica prekrasno (analogično) xlestanul lošad'* 'the coachman lashed the horse excellently (analogously)'.

(4) *Bokr* – 'an animal, a male'; *bokrënok* – its baby. This assertion hinges on the fact that *bokr* is a masculine animate noun and *bokrënok* is a masculine animate noun containing the same root and the suffix *-ënok*. In Russian such formal features are characteristic of word pairs of which the first denotes a grown-up male animal and the second – its baby; cf. *bobr* 'beaver' – *bobrënok*, *golub'* 'pigeon' – *golubënok*, *žerebec* 'stallion' – *žerebënok*, *zver'* 'beast' – *zverënok*, *kozel* 'billy-goat' – *kozlënok*, *kot* 'tomcat' – *kotënok*, *lev* 'lion' – *l'vënok*, *olen'* 'stag' – *olenënok*, *slon* 'elephant bull' – *slonënok*, *som* 'sheat-fish' – *somënok*, *tigr* 'tiger' – *tigrënok*, *tjulen'* 'seal' – *tjulenënok*, *ugor'* 'eel' – *ugrënok*.

(5) *Kuzdra* – an animate being, as only animate beings can execute deliberate actions such as *budlanut'*.

This analysis explains why Russian speakers with no training in linguistics who were asked by the author of the present book to interpret Ščerba's sentence gave the same answer in the great majority of cases: a female animal has struck some male animal forcibly and is dealing blows to its baby (see also ⟨187, *316*⟩).

Various models of decipherment have been described in this chapter – from letter models to semantic models. Let us now take a general look at these models. When we consider the 'output' of certain algorithms, it is not easy to escape the impression that the results obtained by the operation of these algorithms are somewhat trivial. It seems to us that we do not need an algorithm to find out which letters are vowels and which are consonants; we know which of the word forms in a sentence are syntactically related and which are not, and we know that *besedovat'* 'to talk' is semantically closer to *razgovarivat'* 'to talk' than to *ispugat'sja* 'to be frightened' or *izmenit'* 'to alter', and so on.

However, this way of evaluating the models does not do them justice. Models of this type should be evaluated not by the absolute quantity of the results but by their quantity in relation to the quantity of initial information. From this point of view one must admit that the fact that the results are slight and sometimes imperfect is insignificant in comparison with the remarkable fact that so much valuable information about a language and such sound confirmation for intuitive notions can be obtained from such meager initial information.

Research models of this type have also been questioned on other grounds (Hockett ⟨310, *45-46*⟩, Halliday ⟨287, *281*⟩, Chomsky ⟨199, *466*⟩). The criticisms stem essentially from the belief that the units on each level in language do not merely consist of units from lower levels. Thus, an important component of morphemes is their meaning, yet phonemes have no meaning. The meaning of a sentence is not only a total of the meanings of the morphemes it contains; it also includes the meaning of the syntactic structure underlying it. This is demonstrated by the fact that sentences with the same morphological composition can be interpreted in different ways, depending on the syntactic structure with which they are associated; cf. the classical Latin example *amor patris*, where the genitive can be interpreted as either subjective (the father's love) or objective (love for the father).

We shall not dwell here on these criticisms, which are indeed perhaps partly justified. Let us only note that many of the difficulties which seem insurmountable within the framework of decipherment models are resolved by the more flexible experimental models, which can be regarded as decipherment models for texts of infinite length. These models will be discussed in Chapter 8.