[*Proceedings of the National Symposium on Machine Translation*, UCLA February 1960]

Session 10: PROGRAMMING

THE LOGIC OF AUTOMATIC FORMULA SYNTHESIS

Vincent E. Giuliano
Arthur D. Little, Inc., and Harvard University

Researchers in automatic translation have often been asked whether it might be possible to derive translation algorithms automatically--through a machine-programmed comparison of texts in both translated and untranslated versions. Suppose, for example, that parallel bodies of Russian and English scientific text are supplied as simultaneous inputs to a machine; can the machine be somehow instructed to infer or synthesize rules capable of transforming one body into the other? In all fairness, the writer must hasten to comment that this question remains as yet unanswered--because of practical complications connected with translating large bodies of text under carefully controlled conditions. The logic of a simple variety of automatic rule synthesis can, however, indeed be characterized; it is the subject matter of this paper. The logic will be discussed within the setting of a particular application involving English and Russian syntactic patterns, although it may ultimately lend itself to other applications involving parallel texts. The sample application will be characterized only briefly; a more complete discussion may be found in [1] .

We will hypothesize as input to the automatic algorithm-synthesizing process a sizable and representative corpus of Russian scientific text together with a suitable English translation. The Russian text will be presumed to have been subjected to an automatic Russian-English dictionary lookup process, and to a subsequent automatic analysis of Russian syntactic sentence structure. The feasibility of automatic syntactic analysis of Russian is now generally accepted by professionals in the field; and, indeed, experimental computer programs capable of doing such analysis have been described at this Symposium [2, 3, 4] . We will assume that the analysis relates each word in a sentence to an over-all syntactic structure by specifying pertinent dependency relationships such as "subject", "object", etc. , and that concurrently, it removes grammatical ambiguities residual from a simple word-by-word translation.

The parallel English text will be presumed to be prepared from the original in such a manner as to enable automatic cross-identification between the lexical units of the two texts. That is, Russian words must be readily identifiable with their English images. Specialized methods of preparing translations for machine consumption have been described in the literature by Harper, Hays, and Scott [5], Jones [6], Mattingly [7], Giuliano [1] and others; they will not be discussed in detail here. All of these methods are based on specialized postediting of partial machine translations; they all require the posteditor-translator to confine his transformations to ones that can be dealt with automatically. For instance, words must not be moved from one sentence to another. Unfortunately, postediting of the type required involves a significant manual effort; this is perhaps the greatest practical obstacle in the path of automatic formula synthesis.

Finally, we shall suppose that the parallel English text has also been subjected to an automatic process of syntactic analysis--this time of English sentence structure. In the light of the recent successes with Russian syntax, it is plausible that this can be accomplished without undue difficulty.

In the application being described, the automatic algorithm-synthesizing process is to determine the influence of given syntactic variables in the Russian text on producing a known syntactic transformation in the English. Before each machine run, the variables and the transformation must be specified as clues to the algorithm-synthesizer by a human monitor; hopefully, the output will be an algorithm relating the variables to the transformation. Such an algorithm will, of course, be strictly valid only for the given corpus of text. More concretely, the inputs to a formula-synthesizing run might be:

D_r = a determiner formula that indicates to the machine the general type of syntactic structure being investigated. For example, D_r might specify the presence of a genitive noun complement of another noun.

B_r = a structural transformation that might be made by the posteditor in the course of producing the English text from a word-by-word translation. For example, the posteditor might insert "of" before the translation of a given word.

$\phi_1, \phi_2, \ldots, \phi_n$ = a List of binary-valued propositional statements that may conceivably be pertinent in relating the Russian structure defined by D_r to the transformation B_r. Since these are functions of textual position, they will be called "variables". For example:

ϕ_1 = The construction under consideration is within a subordinate clause.

ϕ_2 = The word under consideration (the genitive complement) is modified by an adjective.

ϕ_3 = The word predicting that the word under consideration (the noun accepting the complement) is the object of a preposition.

At any given position in the text, either D_r pertains or it does not. When it pertains, then either B_r pertains or it does not, and each of the $\phi_1, \phi_2 \ldots, \phi_n$ are either true or false. Insofar as the machine is concerned, then, D_r, B_r, $\phi_1, \phi_2, \ldots, \phi_n$ may all be treated as binary-valued variables that are functions of textual position; subroutines must be provided capable of determining the truth value of any of these at any textual position [1].

We are now prepared to discuss the algorithm-synthesizing process itself. The purpose of the logical process is to synthesize a logical formula Φ out of the given $\phi_1, \phi_2, \ldots \phi_n$ that precisely characterizes the conditions when the structure D_r leads to the transformation B_r in the given corpus. A resultant algorithm is then of the form

$$D_r \,.\, \Phi \rightarrow B_r$$

to be read: "Whenever the condition D_r is satisfied and the formula Φ is true, then the transformation B_r is to be performed in the English text".

The first portion of the automatic synthesis process consists of a machine pass through the parallel texts. Both texts are to be scanned simultaneously and in phase with one another, from beginning to end. The scanning is temporarily halted only when the computer senses the presence in the Russian of a syntactic structure satisfying the determiner condition D_r. When a context satisfying D_r is encountered, the computer executes certain testing and incrementing operations before going on. In order to facilitate the discussion of these operations, a brief paragraph will first be devoted to a topic of elementary logic, truth value configurations [8] .

There are 2^n possible configurations of truth values of the variables $\phi_1, \phi_2 \ldots, \phi_n$; these correspond to the rows in the schematic listing of Table 1. A "1" in any position is here taken to mean that the corresponding ϕ_i is true in the given configuration, a "0" that it is false. Thus, in the first configuration all the ϕ_i are false; in the last all the ϕ_i are true. The configurations are uniquely identified by the binary patterns of the 1's and O's; each row in the configuration table corresponds to a binary number k between 0 and $2^n - 1$. The number k can therefore be used as a name for the corresponding configuration of variables.

k	ϕ_1,	ϕ_2, ... ,	ϕ_{n-1},	ϕ_n	Interpretation
0	0	0	0	0	All ϕ_i are false.
1	0	0	0	1	Only ϕ_n is true.
2	0	0	1	0	Only ϕ_{n-1} is true.
.	.	.	.	.	
.	.	.	.	.	
.	.	.	.	.	
2^n-2	1	1	1	0	All ϕ_i are true except ϕ_n
2^n-1	1	1	1	1	All ϕ_i are true.

Table 1

Configurations of Logical Variables

Two sets of index registers $\{X_k\}$ and $\{Y_k\}$ are set up and retained within machine memory during the pass through the parallel texts. The values of k correspond to the configurations of ϕ_i that are actually encountered in the text corpus for contexts that make D_r true. When D_r is true, the appropriate subroutines are used to determine the truth values of each of the ϕ_1 , ϕ_2 , . . . , ϕ_n . The pattern of 1's (trues) and 0's (falses) thus obtained defines a logical configuration k' that characterizes the state of the ϕ_i variables for the instance of sentence structure located at the given textual position. When a given configuration k' is thus encountered for the first time in the corpus, the machine sets aside two index registers, one for $X_{k'}$ and one for $Y_{k'}$, the numbers in both registers being initially set to 0 . Then, and whenever the same k' configuration is encountered in subsequent contexts for which D_r is satisfied, the computer increments the number in the $X_{k'}$ register by 1 .

After an $X_{k'}$ egister is incremented, the computer program ascertains whether the posteditor elected to make the transformation B_r in the corresponding position in the English text. If not, the computer merely continues its scan through the parallel texts, searching for the next instance of Russian sentence structure that satisfies D_r . If, however, the transformation B_r is indeed found in the corresponding position of the English text, the program then increments the $Y_{k'}$ register by 1 before proceeding with the scanning process. The machine goes through the entire corpus of text in this manner, specifying the truth values of $\phi_1, \phi_2, \ldots, \phi_n$ whenever D_r is satisfied, and selectively incrementing the X_k and Y_k registers.

After the text-scanning pass, a second machine program is required to interpret the tally counts in the X_k and Y_k registers. Hopefully, its output will be a logical formula Φ compounded out of the listed ϕ_i variables and the logical connectives "." (and), "v" (or) and "~" (not). At worst, it will be a clear indication that important variables are missing from the ϕ_i list.

The first operation performed by the interpreting program is the computation of a third set of numbers $\{Z_k\}$. For $X_k = 0$, Z_k are undefined; for $X_k \neq 0$, Z_k are defined as $Z_k = Y_k / X_k$. From the counting process, it follows that defined values of Z_k satisfy $0 < Z_k < 1$. The Z_k define the desired formula Φ . It is convenient to discuss the synthesis of formulas in terms of four different types of patterns that can be described by the Z_k :

Pattern Type 1: All Z_k are defined and either 0 or 1.

When a pattern of this type is present, the formula synthesizer has found an algorithm that cannot be improved insofar as the given text corpus is concerned. The vector of binary elements $[Z_1, Z_2, Z_3, \ldots, Z_{2^{n-1}}]$ is itself a representation of the

---

[1] The methods of representing and reducing logical formulas mentioned in this paper are well known in the fields of mathematical logic and algebraic switching theory. Machinable methods for reducing logical formulas to minimal normal forms, for resolving "do not care" conditions, etc. , are treated in [9] , [10] , [11] , and [12] .

desired formula. Since the Z_k are all either 0 or 1, each configuration corresponds to either doing or not doing the transformation B_r, with no equivocation. The formula can be expressed in disjunctive canonical form by taking a sum of the logical products corresponding to the configurations for which $Z_k = 1$. Each product is obtained by conjoining all the n variables, negating just those to which a 0 is assigned in the configuration considered. For example, a simple hypothetical situation is illustrated in Table 2. The formula corresponding to the Z_k is

$$\Phi = \sim \phi_1 . \phi_2 . \sim \phi_3 \vee \sim \phi_1 . \phi_2 . \phi_3 \vee \phi_1 . \sim \phi_2 . \phi_3 .$$

Formulas thus obtained are in a so-called "canonical" disjunctive normal form. They can often be reduced to simpler normal forms by well-known rules of logic [9], [10], [11].

k	ϕ_1 ϕ_2 ϕ_3	X_k	Y_k	Z_k
0	0 0 0	17	0	0
1	0 0 1	4	0	0
2	0 1 0	32	32	1
3	0 1 1	118	118	1
4	1 0 0	2	0	0
5	1 0 1	61	61	1
6	1 1 0	1	0	0
7	1 1 1	75	0	0

Table 2
Hypothetical Pattern of X_k and Y_k
Leading to a Pattern of Type 1

Certain of the variables included in the list $\phi_1, \phi_2, \ldots, \phi_n$ may not be needed in order to construct a valid Φ formula. Such variables will appear in the canonical form of a formula only vacuously. For example, the formula $\sim \phi_1 . \phi_2 . \phi_3 \vee \sim \phi_1 . \phi_2 . \sim \phi_3$ contains the variable ϕ_3 only vacuously, and is reducible to $\sim \phi_1 . \phi_2$. Vacuous variables can be automatically eliminated in the course of reducing a formula to a more minimal normal form.

Pattern Type 2: Defined Z_k are either 0 or 1, but some Z_k are undefined.

A valid algorithm can be synthesized when a pattern of this type is present, but it is not necessarily unique. The undefined Z_k are in one sense like the so-called "do not care" conditions of switching theory [10], [11], [12]. Since configurations corresponding to these Z_k do not occur in the experimental corpus, it might seem that 0's and 1's could be assigned to them in any desirable manner. In fact, machinable procedures exist for assigning values to Z_k for "do not care" configurations in such a way as to simplify the resulting formula [10], [11]. Assigning such values automatically in this somewhat offhand fashion would most likely not, however, be a sound experimental procedure. Different formulas would result from assigning different sets of values to the undefined Z_k. While all such formulas would work equally well for the experimental corpus, they would behave differently in the event that one of the "do not care" conditions actually occurred in another text. If the value 1 were assigned to a $Z_{k'}$ that should actually have the value 0, then the algorithm would erroneously lead to the transformation B_r whenever configuration k' is encountered in another text. To be safe, then, it is probably best to adopt a blanket rule for assigning values automatically; the machine is to assign the value 0 to each of the "do not care" Z_k. A synthesized algorithm will then not lead to the transformation B_r if one of the "do not care" configurations is encountered in a later text. Strategies alternative to this one have, however, been proposed by Lawler [13] in an interesting paper that views automatic algorithm synthesis as a statistical game.

Consideration might well be given to the use of a ternary-valued logic to enable better treatment of the "do not care" conditions. Assigning the value 0 to the undefined Z_k is a "fail-safe" procedure since the resulting algorithm leads to the execution of the action B_r only in textual situations actually examined in the experimental corpus. Nevertheless, the effect of a 0 assigned to an undefined Z_k is the same as that of a 0 computed from a non-vanishing X_k. Certain information is therefore not reflected in the algorithm: in the former case the configuration was not encountered, in the latter case it was encountered and found to have the value 0. It may be possible to keep better track of this information by using a three-valued logic, where one of the values means "unresolved".

Pattern Type 3: Some of the Z_k are proper fractions, $0 < Z_k < 1$, but at least one Z_k is 1 .

A valid algorithm can be obtained when a pattern of this type is present, but this algorithm will be "weak" in the sense that it does not account for all instances of D_r leading to B_r in the experimental corpus. The fractional values of Z_k correspond to configurations that only sometimes lead to the given B_r transformation. Other variables besides those included in $\phi_1, \phi_2, \ldots, \phi_n$ must be taken into account when these configurations are present. The weak algorithm is obtained by simply rounding off each of the fractional Z_k to zero, thus giving a pattern of type 1 or 2 that can be reduced by the methods already discussed. It is important to stress the fact that weak algorithms are also "fail-safe" insofar as the experimental corpus is concerned; a derived algorithm leads to the transformation B_r only for configurations that always lead to the transformation in the experimental corpus.

Pattern Type 4: Some Z_k are fractional and no Z_k is 1.

When a pattern of this type is present, no configuration of the given variables unambiguously leads to the given action, and it is not possible to synthesize a valid basic algorithm from $\phi_1, \phi_2, \ldots, \phi_n$. Pertinent variables are clearly missing from this list and must be identified by the monitor before successful results can be obtained from the automatic process.

Outputs of the logical formula-synthesizing process might consist of the derived algorithm, in both printed and machine-readable format, and an edited list of the pertinent X_k , Y_k , and Z_k counts. The list should facilitate human monitoring and control of the process. The counts give an indication of the relative occurrence frequencies of the various configurations; they should enable evaluation of a derived algorithm in terms of the types and frequencies of the situations encountered. Again, to the extent that the experimental corpus is only approximately representative of what can occur in Russian technical writing, so also will the algorithms synthesized from this data be, at best, only approximately valid. A discussion of the degree of validity to be assigned to a formula obtained from a given corpus is, however, plainly beyond the scope of this paper. A machine-derived algorithm must certainly be subject to human scrutiny and evaluation before it can be finally accepted.

Results of late research in the syntactic problems of Russian-English translation are encouraging--so much so that there is some doubt as to the need in this area for such a relatively exotic tool as automatic algorithm synthesis. Nevertheless, the logical process may someday prove useful in an exploration of the "fine structure" of syntactic transformations. That is, the method might help in the detection and analysis of relatively infrequently occurring phenomena involving complex interrelationships of syntactic variables.[2] Beyond the scope of Russian-English syntax, moreover, the logical techniques might prove to be useful in the study of other language pairs that now remain relatively unexplored.

---

[2] This point, as well as several others germain to the topic of automatic algorithm synthesis, was raised by David Hays in Session 1 of this Symposium.

REFERENCES

[1] Giuliano, V.E. , "A Formula Finder for the Automatic Synthesis of Translation Algorithms", Mathematical Linguistics and Automatic Translation, Report NSF-2, Section IX, Harvard Computation Laboratory, 1959.

[2] A New Approach to the Mechanical Syntactic Analysis of Russian, National Bureau of Standards Report 6595, 1959.

[3] Harper, K. E. , and Hays, D. G. , The Use of Machines in the Construction of a Grammar and Computer Program for Structural Analysis, RAND Corporation Report P-1588, 1959.

[4] Sherry, M.E., "Predictive Syntactic Analysis", presented in Session 3 of this Symposium.

[5] Harper, K. E. , Hays, D. G. , and Scott, B. J. , Manual for Post-editing Russian Text, RAND Corporation Report P-1624, 1959.

[6] Jones, P. E. , "A Feedback System for the Harvard Automatic Translator", Mathematical Linguistics and Automatic Translation, Report NSF-3, Section XIV, Harvard Computation Laboratory, 1959.

[7] Mattingly, I. G. , "Post-editing for Feedback", Ibid., Section I.

[8] Quine, W. V. , Methods of Logic, Henry Holt and Co. , New York, (1953).

[9] Synthesis of Electronic Computing and Control Circuits, Annals of the Computation Laboratory of Harvard University, Vol. 27, Harvard University Press, Cambridge, 1951.

[10] Karnaugh, M., "Synthesis of Combinational Logic Circuits", Communication and Electronics, No. 9, pp. 593-598, November, 1953.

[11] Roth, J.P. , "Algebraic Topological Methods in Synthesis", Proceedings of an International Symposium on the Theory of Switching, Annals of the Computation Laboratory of Harvard University, Vol. 29, Harvard University Press, Cambridge, 1959.

[12] Quine, W. V. , "Two Theorems about Truth Functions", Boletín de la Sociedad Matemática Mexicana, Vol. 10, pp. 64-70, 1953.

[13] Lawler, E. L. , "The Empirical Formulation of Translation Algorithms", Papers Presented at the Seminar on Mathematical Linguistics, Vol. V, Harvard University, 1959.