DAVID G. HAYS

READINGS IN AUTOMATIC LANGUAGE PROCESSING

AMERICAN ELSEVIER

TP3
H425

7963215
5

READINGS IN AUTOMATIC LANGUAGE PROCESSING

EDITED BY

DAVID G. HAYS

The RAND Corporation

NEW YORK

AMERICAN ELSEVIER PUBLISHING COMPANY INC.

1966

SOLE DISTRIBUTORS FOR GREAT BRITAIN
ELSEVIER PUBLISHING COMPANY, LTD
Barking, Essex, England

SOLE DISTRIBUTORS FOR THE CONTINENT OF EUROPE
ELSEVIER PUBLISHING COMPANY
Amsterdam, The Netherlands

Library of Congress Catalog Card Number: 66-19743

MANUFACTURED IN THE UNITED STATES OF AMERICA

Preface

The readings collected in this volume are not intended to substitute for a textbook, but to supplement one. In a university course, with a series of lectures to provide the central framework and a few longer monographic publications to treat key areas in greater detail, this volume might suffice, but that is not its proper role.

It is not intended to introduce a novice to the digital computer, for he can acquire a better introduction in many ways; the elementary statements in the following introduction may lead some readers to go in search of a proper description of the computer and how it is programmed, but that is another matter. A previous acquaintance with linguistics is also advisable.

The papers presented here are not always the historically significant contributions, but if I had taken a historian's view I could not have abridged the individual readings as I did.

I have included these papers because they epitomize, in their various ways, methods, solutions to central problems, or approaches to the use of the computer as a processor of natural language. Other papers will undoubtedly refine, or perhaps supersede, the concepts that the reader can learn from these pages. But I have tried to choose papers in which sound concepts were developed with enough richness of detail to let the reader see how it all works.

I am grateful to the authors who permitted me to reprint their work, and to abridge in order to conserve space. I am likewise grateful to the publishers whose permission to reprint I hereby acknowledge. And I apologize to the authors of equally meritorious papers for which I was not able to find a place. I trust that the reader, after this introduction to the field, will be prepared to seek them out.

D.G.H.

7963215

Table of Contents

1 Introduction

This book takes the name of a field that encompasses every use to which computers can be put in the manipulation of ordinary language—the kind of language spoken and written every day by almost every human being in the course of his private, commercial, or academic routine. The papers in this book do not cover the whole breadth of what computational linguistics will one day span. It would not be possible, as computational linguistics is entering its second decade and the digital computer its third, to predict every application to which linguists will put the computer in the future nor the kind of programs that will make the computer an entirely useful tool for the jobs that have already been invented for it. By 1945 it was obvious to a few engineers and mathematicians that automatic machines could be constructed to perform any numerical calculation; by 1955, a few linguists had realized that the advancement of their science required the computer, and that by means of their knowledge the computer could be put to work in new, useful areas. Perhaps by 1975 applied computational linguistics will be known to everyman, and it will be possible to estimate the true market for automatic language processing. In 1965, university scholars are just beginning to adopt the computer as a research tool, and commercial applications are just beginning to show a profit.

The importance of the automatic digital computer as a laboratory instrument is known to everyone. It serves two major purposes: reduction of data and derivation of conclusions from theoretical premises. Like physics, with its voluminous collections of cloud-chamber photographs and like sociology, with its massive collection of census reports and other demographic data and like biochemistry, with tens of thousands of compounds to test for possibly beneficent physiological activity, linguistics has more data than any previous generation of linguists knew what to do with. Every published book, newspaper, and magazine is a sample of human use of language, and if he prefers to observe language in its spoken rather than its written form, the linguist has only to turn on a tape recorder beside his radio or television set or wherever a conversation takes place. But it does no good to record these data if analysis is impossible. The human scholar can hope to identify only a few of the patterns, regularities, and systematic features that could be found by fully detailed, uniformly thorough analysis of every available specimen of human language. Insofar as the computer can be programmed to inspect the material at hand, and to report what it finds in summary terms, it becomes the linguist's indispensable assistant.

A theory is a collection of fundamental propositions about reality. Taken together, these propositions predict many—often indefinitely many—consequences, sometimes testable and sometimes not. The task of the theoretician is to discover what these consequences are, and it is exceedingly hard work. Computers are often used as working models of theoretical systems; they are allowed to operate randomly, except for the constraints imposed by the theoretical principle being tested, so that the results of their calculations can be ascribed in part to chance, in part to the theory being tested. This technique, first called the Monte Carlo method and now often described as simulation, has given the computer a second major role in science. The intricate phenomena called natural language can be described only by theories of great complexity; computer simulation is being used to test the consistency of linguistic theories and to help isolate the weak points where they require strengthening.

Outside the field of linguistics, and therefore to be counted among applications, several branches of social science need the digital computer in its guise of language processor as a laboratory tool. Psychology, sociology, and anthropology are largely—if not altogether—dependent on the analysis of what the persons studied say and write. Social scientists conduct interviews, observe group discussions, collect letters, diaries, and other private communications, examine the contents of newspapers and other mass media, and so on. Whereas the linguist's interest is in the form of expression, the social scientist's concern is with content, and a special kind of content at that: the internal state that causes a person to utter a particular sentence, in a particular context, at a particular time, and the effect on his audience of what he says. To take the case of group discussions, how can affection, respect, and authority be predicted from what the discussants have said to one another in the course of a few hours of conversation? Or to take an example from history, how can the motivations of a leading personality be estimated from his memoirs, notes, and other papers of a certain period? A computer program based exclusively on the linguist's theory of language could not be expected to serve the social scientist's purposes, yet one might expect that analysis of forms of expression would be, in each application, prerequisite to the analysis of content, and so it has turned out.

The library is the scene of another exceedingly obvious application of computational linguistics. Knowledge is preserved from generation to generation in many forms, in sound recordings, motion pictures, charts and diagrams, models, and many other guises; yet text remains the predominant medium for the preservation of what the earlier generations have learned. Until writing was invented, the upper limit on what a culture could remember of its own past was narrowly limited; everything had to be remembered, and even with specialists in the art of memorization no culture was ever able to preserve a small fraction of what the earliest collections of manuscript brought together. Libraries have been growing for two millenia, and they seem to be reaching a new kind of limit: the upper limit of what can be preserved by a culture without active mechanisms for locating what is stored. Subject classifications and indexes, implemented with card catalogues and similar systems, are scarcely adequate for libraries of twenty million volumes, and the world's most important research libraries are now passing that mark. The computer has been called on to provide a solution to this problem; like the social scientist, the librarian is concerned with the content of text, not with the form. An article in a scientific journal is intended as a contribution to knowledge; the librarian's task is to discover where, in the vast, indefinitely complex body of facts comprising man's knowledge of himself and his universe this contribution fits. What does it repeat in order to demonstrate its proper position, what does it contribute that is novel, what new connections does it make among known facts? Perhaps no librarian has ever made such an analysis of a library, but librarians with computers programmed for automatic language processing may eventually be able to do just that. Such at least is the goal of automatic documentation.

Other important and difficult areas of application for computational linguistics have been noted. Machine translation was probably the first. Here the goal is to translate the content of a document or spoken message from one spoken language into another. Programming digital computers, even when the purpose of the program is simple numerical processing, is to some degree applied computational linguistics. The languages used by the earliest programmers were exceedingly simple, not calling for the same kind of theoretical models for their analysis as do the natural languages. As programmers attempt to do more and more difficult tasks, they require more and more powerful languages. Furthermore, they are attempting to relinquish their original duties to less well-trained users. If a person without detailed training in the design of computer programs is to explain directly to the computer what he wants done in more and more difficult cases, computers must be provided with programs for language processing in order to translate the user's statement into a sequence of operations.

Thus one branch of computer programming has moved very close to linguistics. Other applications of automatic language processing in business and government come to mind as soon as one observes the obvious fact that buyers and sellers, administrators and underlings, like all the rest of humanity, exchange enormous quantities of words every day. To help them in processing their correspondence would be profitable. The applications in teaching could be equally important. Finally, there is a range of applications that calls for much less sophisticated knowledge of natural language, but that has already demonstrated how major changes in industry can be brought about with simple ideas. Typists and typesetters occupy themselves for millions of hours each year with the remaking of text: copying from one sheet onto another, producing "clean copy" by collating corrections, deletions, format instructions, and hyphenation with "a preliminary draft." If the preliminary draft is stored in the memory of a digital computer, the collation can be made automatic. The typist need only transcribe the amendments, leaving the computer to carry forward what was correct on the draft, to put the changes in at the right places, to arrange the format as it ought to be, and to justify lines if that is called for. Many newspapers are now produced in this manner, books are printed with the help of computers, and it seems obvious that in time virtually every typewriter operated by a paid employee will be connected in some manner to a computer. And here the circle closes, for the text put into the computer for the sake of reducing cost and increasing speed in the publication industry becomes a base on which the linguist can perform his research.

Pure computational linguistics and its several applications are unified by their common employment of certain techniques and processes that make up the specific substance of the field. Linguistics as a whole, and abstract linguistic theories, are not directly concerned with the computer. Algorithms appropriate for carrying out the processes suggested by linguistic theory and the facts of natural language are a special problem, one with which the linguist need not concern himself unless he chooses. The computer can aid him, whether he understands its workings or not. Nevertheless, tradition has it that the good craftsman knows his tools and, therefore, every linguist has good reason to choose to know at least a little about the computer.

In comparison with empirical procedures, computer-based methods for handling large files of text, large dictionaries, complicated grammars, and other large files of data are fantastically easy. Still, these files have to be prepared for input to the computer, arranged conveniently in the computer's active storage area, transferred to and from inexpensive media for permanent retention, reorganized, and printed out for inspection. In short, one group of elementary processes required throughout computational linguistics constitutes a system for file maintenance. Textual and similar materials generally go into the computer by way of a keyboard operation. There are typewriters that produce paper tape, punched cards, and magnetic tape; they are manufactured with many different character sets, some very limited and others satisfyingly large. It is also possible to connect one or many typewriters directly to a computer, through electrical wiring. The operating speeds of contemporary computers are so high that a computer can recognize each key stroke as it occurs on each of a hundred typewriters being operated simultaneously by fast typists and still have time between strokes to carry out some simple computations.

Inside the computer, the usual storage medium is an array of magnetic cores, each holding just one bit of information; six or eight bits are commonly used to represent a character. Standard large computers can retain about two hundred thousand characters in storage. Magnetic tape, on which information is recorded in the same way that sound is recorded by home tape recorders, is the usual medium for inexpensive storage. Information can be transferred automatically at high rates between machine storage and tape.

The computer is also equipped with its own printing devices, operating at high speed, with fonts of from 48 to about 250 characters. Type can be set automatically from tapes produced by computers; although this process is slower and more expensive than direct computer printout, it provides for an exceedingly wide variety of characters and very high quality.

The most elementary operations of input, internal arrangement, and output are provided for either in the equipment as constructed or in programs furnished for general use. The linguist working with computers does not have to begin by designing typewriters or working out convenient ways of moving large volumes of information to and from magnetic tape. The lowest level that he has to control for himself is the encoding of his large character set and the features of arrangement on the printed page that he wishes to preserve. A six-bit machine character corresponds to one of sixty-four possibilities; in publishing a book, a typographer can call for characters from a set of several hundred or several thousand. One of the typographer's characters must therefore be represented, in some manner, by several machine characters. This is the problem of text encoding. A good scheme retains all the useful information of the printed book and makes it easy to isolate those aspects of the information about a character that are needed, for example, in alphabetization.

The linguist's files are organized in various ways, from a linguistic point of view, and their organization must be represented within the computer. Text is organized into articles, books, subject collections, and so on. Dictionaries are organized into entries, each with various parts: heading, grammatical description, definition, and so on. A grammar, a bibliography, a concordance, an index verborum, a thesaurus, a computer program—each kind of file has a definite pattern of organization. After the content of the file is first prepared and put into the computer, it is often necessary to add further information, either for a new element—as when new words are added to a dictionary—or of new kinds. Files from different sources, but of the same kind, sometimes have to be merged; thus, two dictionaries of the same language might be assembled into a single, larger dictionary. In merging, provision must be made for putting the elements of the composite file in proper order, detecting and dealing appropriately with overlap, and so on.

Files have to be consulted; in some cases, consultation amounts to looking for one specific item whose place in the file may or may not be known. Thus, if a dictionary is arranged alphabetically by heading, finding the entry for a given word entails only going a certain distance through the alphabet; but if a bibliography is arranged alphabetically by author, and one or more entries on a certain subject are to be found, the subject identification in each entry must be checked.

Sometimes file consultation must be performed on a wholesale basis, which is as much as to say that the content of two files must be compared. For many purposes, the processing of text must begin with dictionary lookup for every word. When the text is small enough to be stored in the high-speed memory of the computer, the dictionary can be read in little by little, and each entry in turn can be compared with all the words in text; when the dictionary is small enough to be stored, the text can be read one word at a time. When both dictionary and text are too large, some process must be found to facilitate all the comparisons; and when both are large enough, the magnitude of the task justifies ingenuity of the highest order.

Syntactic analysis of a text, the determination of sentence structures, can be regarded as another kind of file consultation. Yet it is certainly more complex than dictionary lookup. The units of text that are to be matched with dictionary entries are explicitly, overtly present in the text, ready to be matched. A grammar, when it can be regarded as a list of phrases, consists of phrase descriptions to be matched against text, but the phrases are only implicitly present in the text to begin with. A phrase described in the grammar may consist of two parts, of which one part is potentially itself a phrase. But the constituent phrase has its parts in turn, and only the ultimate constituents are overtly represented in the text after dictionary lookup. Parsing algorithms are devoted to cyclical, or recursive, comparison of text with grammar so that in effect the phrasal construction of the text is made explicit, each phrase appearing in time to be identified as part of a larger phrase.

Grammars are large tables, when they can be treated as tables at all, and unlike dictionaries their entries are interrelated in complex ways. A dictionary can be revised by the simple addition of a new word. When a new kind of phrase is added to a grammar, or when a new distinction among parts of phrases is introduced, many other parts of the grammar can be implicated, and far-reaching alterations may be necessary. How best to organize a grammar for consultation during parsing has been the object of much investigation. Besides tables, computer programs have been suggested as representations of grammar, and systematic accounts of the features that permit constituents to be bound together in phrases have also been recommended.

The inventory of elementary processes out of which programs for automatic language processing can be built is not exhausted by our current knowledge of the structure of language. We are well enough aware of our ignorance to see, dimly, further aspects of language structure, and much of what we can hope to bring within our control during the years ahead is concerned with the representation of meaning. One necessity that seems to face us is the use of further dictionaries. In the first dictionary, a string of letters is associated with grammatical and other properties; in the second dictionary, syntactically connected parts of sentences are tabulated and associated with elements of meaning. Such dictionaries are apparently necessary in information retrieval and all other applications where content, the substance of what is said, must be processed.

No branch of science or technology is ever complete, but books occasionally must be completed so that science and technology may go on. To regret that this volume of readings cannot cover every conceivable branch of its subject would be to deny its readers the hope that by clear thinking and industry they may contribute to further growth. There is room for growth; our ability to process natural language materials automatically, almost nonexistent a decade ago, has reached a level that permits us to look forward to more useful and more intellectually satisfying, and certainly far more complex, elaborations.

2 Specification Languages for Mechanical Languages and their Processors – A Baker's Dozen

Saul Gorn

1. Introduction

Many varieties of mechanical languages and the languages which specify their syntax exist. By many techniques and devices these syntactical languages specify concepts and processes. In these languages it is possible to specify the same object in different ways to obtain clarity from different points of view. The choice of the language depends upon its convenience in specifying or communicating a concept.

The purpose of this paper is to show to what extent the languages illustrated are capable of specifying such mechanical languages or their processors, and to what extent these specifying languages are equivalent in their ease of mechanical translation among themselves.

To dispel some of the confusion as to the power of applicability of these languages, one trivial example is worked over in a dozen different ways. The simplicity of this example provides the link permitting the comparison of the methods of specification. The processor (used by McNaughton as an illustration of logical nets) being specified we will call the *triple sequence alarm.* The corresponding input language of strings of zeros and ones we may call the *triple one sequenced strings.* The processor itself as shown in Fig. 1 may be considered a *data generator* or a *triple one sequenced string recognizer.*

The dozen specifying languages can be given the titles and classifications

Note: The material in this paper comes from the University of Pennsylvania's Office of Computer Research and Education and is an outcome of the work jointly supported by the National Science Foundation Grant G-14096 and the Air Force Office of Scientific Research AF-49(638)951. Reprinted with the permission of author and publisher from *Communications of the Association for Computing Machinery,* Vol. 4, No. 12 (December 1961), pp. 532–542.

of Table 1. The groups not separated by double lines are mechanically translatable into one another from top down. Translation between these groups calls for heuristic methods.

The examples will illustrate some of the terminology presented in the

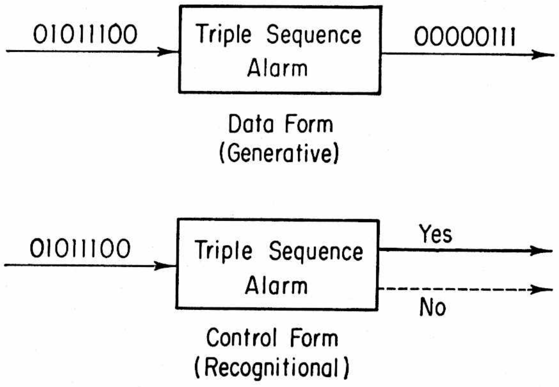

Fig. 1.

author's paper, "Some Basic Terminology Connected with Mechanical Languages and Their Processors" (*Comm. ACM 4* [August 1961], 336). Some of these languages are essentially one-dimensional (linear) and some are essentially two-dimensional (graphic and tabular); some are purely sequential, and some permit simultaneous action; some are more suitable for behavioral (recognitional) specification, and some are better for structural (generative) specification; one is a command language where the others are descriptive; and one is a sublanguage of natural language where the others are mechanical. Finally, some are more suited to specifying languages and some to specifying processors.

For example, one might have a language which is suited to specifying languages structurally but which can be used to specify the *processors* of those languages behaviorally.

2. Natural Language

The "triple sequence alarm" is a device with one input and one output. The input will accept sequences of signals of equal duration chosen from two standard signals which we will designate by the symbols 0 and 1. The output will emit during each of the signal duration intervals one of the two signals, 0 or 1, acceptable by the input. The device is such that a signal 0 will be produced at each signal duration interval until the first occurrence of a signal 1 which had signals 1 as its two immediate predecessors; at

Table 1. Classification of Specifying Languages

Title	*Dimension*	*Timing*	*Mode*	*Structural or Behavioral*	*Processor or Language*
1. Natural Language	1?	Sequential or Simultaneous	Descriptive or Command	Structural or Behavioral	Processor or Language
2. Regular Expression	1	Sequential	Descriptive	Structural --------- Behavioral ---------	Language Processor
3. Backus Normal Form	1	Sequential	Descriptive	Structural	Language
4. Trees	2	Sequential	Descriptive	Structural	Language
5. Prefix	1	Sequential	Descriptive	Structural	Language
6. State Diagram	2	Simultaneous	Descriptive	Behavioral or Structural	Processor
7. Symbol State Diagram	2	Simultaneous	Descriptive	Behavioral or Structural	Processor
8. Incidence-Matrix	2	Simultaneous	Descriptive	Behavioral or Structural	Processor
9. Logical Net	2	Simultaneous	Descriptive	Structural	Processor
10. Propositional Logic with Time Variable	1	Simultaneous	Descriptive	Structural	Processor
11. Logical Equations	1	Simultaneous	Mixed	Structural	Processor
12. Turing Machine	1 or 2	Simultaneous	Descriptive	Behavioral or Structural	Processor
13. Flow Chart	2	Sequential or Simultaneous	Mixed or Command	Behavioral or Structural	Processor

that duration interval and for each successive interval until the end of the input string the output signal will be 1.

What we have just given is a descriptive, behavioral specification of the triple sequence alarm processor in a linear sequential sublanguage of a natural language.

3. Regular Expressions

The language-naming language we now present will have, as basic processors, generators and recognizers for a certain class of infinite classes of finite strings of zeros and ones. One such infinite class, for example, is the class of all finite strings of zeros and ones containing somewhere three successive ones, that is, the class of all input strings for which the triple sequence alarm ends with a signal of 1.

The language is that of Kleene's [15] "regular expressions" as modified by McNaughton.[16] We now specify the "regular expression language-naming language" in natural language (our next example will present the specification in a mechanical language), but in a generative manner known as a "production system" (see Gorn [7, 8, 9]).

The regular expression language is a linear sequential language of strings of characters chosen from the alphabet {0, 1, (,), *, $\vee$}. Each string of characters from this alphabet which is a regular expression will be a partial specification of a linear sequential language of strings of characters chosen from the alphabet {0, 1}; it will be a structural specification when a processor is specified which will generate all these strings. The purely syntactic generative specification of the regular expression language is the following *production system*

SY_a The one-character string, 0, is a regular expression.

SY_b The one-character string, 1, is a regular expression.

SY_c If a string designated by α is a regular expression, then the string composed in left to right order of "(", the characters of α, ")", and "*" is also a regular expression (briefly, if α is a regular expression, then so is "$(\alpha)*$").

SY_d If the strings designated by α and β are regular expressions, then so is the string constructed by taking all the characters from left to right of α and following them immediately on the right by all of the characters from left to right of β (briefly, if α and β are regular expressions, then so is $\alpha\beta$). This basic procedure for all linear sequential languages is called "concatenation".

SY_e If the strings designated by α and β are regular expressions, then so is the string beginning with "(" concatenated on the right by the characters of α, followed on the right by "$\vee$", then concatenated on