

KEDMA – Linguistic Tools for Retrieval Systems

R. ATTAR

Bar-Ilan University, Ramat Gan, Israel, and The Weizmann Institute of Science, Rehovot, Israel

Y. CHOUEKA

Bar-Ilan University, Ramat Gan, Israel

AND

N. DERSHOWITZ AND A. S. FRAENKEL

The Weizmann Institute of Science, Rehovot, Israel

ABSTRACT In a full-text natural-language retrieval system, frequent need for automatic linguistic analysis arises, e.g. for keyword expansion in a search process, content analysis, or automatic construction of concordances. The availability of sophisticated linguistic tools, which is highly desirable for languages such as English, is quite imperative for, say, Semitic languages, whose complex morphological structure renders simple-minded and approximate solutions such as suffix stripping totally useless. Sophisticated tools were designed and constructed via the fusion of grammatical analysis and grammatical synthesis, resulting in a set of global files which provide in some sense a complete grammatical and lexical description of the language. These files induce a set of local files which adapt to the database at hand and permit flexible on-line morphological analysis.

KEY WORDS AND PHRASES: computational linguistics, grammatical synthesis, grammatical analysis, grammatical synthesis-analysis, *linguistic files, global files, local files, information retrieval, full-text, automatic text processing*

CR CATEGORIES: 3 42, 3.70, 3 71, 3 75

1. Introduction

KEDMA is an experimental project in computational linguistics whose purpose is to provide the conceptual framework and working tools needed to satisfy various linguistic requirements in automatic text processing systems, and in particular those related to information retrieval systems. It has been implemented in a full-text retrieval system with a Hebrew database; hence the title KEDMA, an acronym of the Hebrew phrase "Grammatical Files in Retrieval Systems." The problems discussed, the suggested

This work was done within the Responsa Retrieval Project, developed initially at The Weizmann Institute of Science and Bar-Ilan University, now located at the Institute for Information Retrieval and Computational Linguistics (IRCOL), Bar-Ilan University, Ramat Gan, Israel. All authors are partially affiliated with IRCOL, Bar-Ilan University. The Responsa Retrieval Project is partially supported by the U.S. National Endowment for the Humanities through a grant to Bar-Ilan University. This work was also supported, in part, by the Commission of Basic Research of the Israel National Academy of Sciences and Humanities, through a grant to the second author.

Authors' addresses: R. Attar and Y. Choueka, Department of Mathematics and Computer Science, Bar-Ilan University, Ramat Gan, Israel; N. Dershowitz, Artificial Intelligence Laboratory, Stanford University, Stanford, California 94305; A. S. Fraenkel, Department of Applied Mathematics, The Weizmann Institute of Science, Rehovot, Israel

solutions, and the set of files provided by KEDMA and by its implementation in information retrieval systems are of general applicability. Although specially tailored for Hebrew, it is applicable to similar Semitic languages (e.g. Arabic), and we believe the general ideas to be valid also for other languages with a rich and complex morphological structure, such as Russian or German.

KEDMA was established in 1970 as a subproject of the Responsa Retrieval Project – a full-text information retrieval system for Hebrew case law, spanning ten centuries. The database of this system comprises currently 102 volumes of Responsa and the main working file is a concordance (inverted file), which contains the *coordinate* (author code, volume, document number, paragraph number, sentence number, and word number within sentence) of every single word in the text. The user defines his query by a combination of keywords and metrical and Boolean operators. For fuller details see [6, 11].

KEDMA was initiated to meet the linguistic requirements of a study investigating methods for improving retrieval in full-text systems through feedback processes based on automatic content analysis (see [1, 2]). Subsequently, KEDMA was expanded into a more general purpose project, aimed at providing a variety of linguistic applications. These include automatic construction of concordances (a far-from-trivial task for texts written in highly inflected languages – see [12]), mechanical resolution of morphological ambiguities [7], statistical, stylistic, and syntactical analysis of Hebrew texts, etc. Last but not least, KEDMA enables us to develop an on-line interactive mode for the Responsa retrieval system, which is to date still a batch system.

KEDMA is based on a unified *synthesis-analysis* method—a combination of partial *global* grammatical “synthesis” (inflection, declension, conjugation, etc.) and partial *local* grammatical “analysis” (prefix stripping), by which a complete automatic processing of the morphological aspects of any given Hebrew form—a legitimate string of characters—is achieved. The global files, together with an appropriate algorithm for adjoining or stripping of prefixes, allow for the expansion of a “dictionary form” s into the set $F(s)$ of all its derivative forms, and the stemming of a form f into the set $S(f)$ of all dictionary forms from which f may be derived (Section 6).

Applying the “synthesis-analysis” method to an actual database (in our case the Responsa retrieval system), we construct a set of “local files.” The local files are similar to the global files, but are local to the actual “dictionary” (i.e. list of all different forms) of the database. In particular they contain forms with prepositional prefixes (unlike the global files).

In Section 2 we introduce the concept of the *vocabulary tree*, which provides the general framework and appropriate setting for our approach. Some of its possible applications are listed in Section 3. These applications constitute the main motivation for this work. A short description of Hebrew morphology and the problems it poses for automatic processes is given in Section 4. In Section 5 the local synthesis and analysis approaches and their shortcomings are described, followed by a discussion of the interesting possibilities opened up by their fusion. Section 6 contains the details of the KEDMA approach and a description of the “global files” and “local files,” which are the main products of KEDMA. Some concluding remarks are given in the final Section 7.

2. The Vocabulary Tree

Semantic analysis is a desirable component of any natural-language information retrieval process. Unfortunately, the semantic theory of natural languages has not yet reached the stage where it can be fruitfully applied to “real-life” retrieval systems. In particular, the concept of semantic relations between words is not sufficiently well defined to allow full treatment. In addition to the “obvious” relation between synonyms, the relation between *searchonyms*—words which play the same semantic role in a specific context or search—have also to be considered. Thus “airplane” and “tank” are searchonyms in

the context of “weapons,” but are certainly not related in the realm of “flying objects.” In the latter context “airplane” and “birds” are searchonymous, while “bird” and “tank” are not normally related at all.

Because of difficulties of this sort, we restrict ourselves to the more modest but better defined goal of studying and implementing those semantic relations which are reflected in the grammatical and morphological properties of the language. To this end we introduce a “vocabulary tree,” which will be used to model the semantic facets of the grammatical relations of a language vocabulary. Before going into the details of the vocabulary tree, we define some of the terms and notions which will be used in the sequel.

A *form* is any finite string of characters which does not include spaces and punctuation marks, but is preceded and followed by a space or punctuation mark.

An *l-form* (“language form”) is any form which either appears in a standard dictionary of a given language L or is a valid grammatical variant of such a form.

A *t-form* (“text form”) is any form found in a given text T . Thus a *t-form* may be an *l-form*, but, in addition, it may be a proper name, acronym, abbreviation, or some other form peculiar to a given text which does not appear in a dictionary of the language. It may even be a misprint. For any given finite text T and language, there are forms which are neither an *l-form* nor a *t-form* of T . In the sequel we occasionally use “form” for “*l-form*” or “*t-form*” when the meaning is clear from the context.

A *word* is an occurrence of a *t-form* in the text. Thus, the occurrences of the *t-forms* of a text, counting multiplicities, are all the words of the text. For example, in the text “The computer we use is the IBM 370/168 computer,” there are 5 *l-forms*, 7 *t-forms*, and 9 words.

A *d-form* (“standard dictionary form”) s of an *l-form* f is an entry in a standard dictionary of the language, from which f is grammatically derivable: It is, in some sense, its “normal form.” Thus the *d-form* of “computing” is “compute,” of “went” is “go,” of “children” is “child,” etc. Note that a *d-form* is also an *l-form*.

A *root* is a form representing the name of a class of several *d-forms* which are morphologically and semantically related. Usually it comprises a few basic letters common to all the *d-forms* of the class. Thus “computer,” “compute,” “computerize,” “recompute,” etc., are *d-forms* which can be grouped in a class under the root “comput.” It is important to emphasize, however, that unlike the case of *d-forms*, the exact spelling of the root is unimportant; in fact, it can even be just a serial number, since it is only a name for a “basket” into which we put some *d-forms* which are grammatically and semantically related.

Incidentally, note that our terms “*d-form*” and “*root*” constitute a subdivision of the widely used term “*stem*.” The latter may comprise several—but not necessarily all—*d-forms* derivable from a root. In contrast to the *d-form*, it may be a form which is not an *l-form*. For example, the stem “comput,” which is not an *l-form*, comprises “compute,” “computerize,” etc., but not “recompute.”

The *vocabulary tree*, as depicted in Figure 1, displays the vocabulary in three levels: roots, *d-forms*, and *l-forms*. Clearly the tree can be subdivided into finer levels, either by detailing the grammatical derivations from roots to *d-forms* to *l-forms* or by elaborating on semantic relations within and between the levels of the vocabulary tree. An example of the first kind will be given later for the Hebrew language (Section 4, Figure 2). As to the second possibility, we shall not elaborate further on the semantic relations in this paper, but rather concentrate from now on on the vocabulary tree of Figure 1.

The tree defines two functions:

(i) *Top-to-bottom expansion*: The mapping from a node onto its set of sons, e.g. in Figure 1, $s_1 \rightarrow \{f_1, f_2, f_3, f_4\}$.

(ii) *Bottom-to-top stemming*: The “inverse” mapping, from a node onto its father(s), e.g. $f_1 \rightarrow s_1$. Since two (or more) *l-form* nodes may be labeled identically (when the

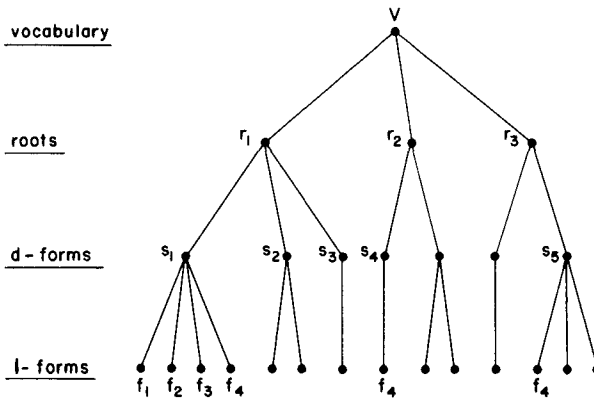


FIG 1 The vocabulary tree exhibiting the relations between d(dictionary)-forms, their fathers (roots), and their sons (l-forms)

labels represent homographs), a given string of letters may have several different fathers, and thus stemming is also a multivalued function, as for example $f_4 \rightarrow \{s_1, s_4, s_5\}$ in Figure 1. The grammatical tree relations can be constructed by either of two morphological processes:

(a) *Synthesis or expansion* ($s \rightarrow F(s)$): Generating the set $F(s)$ of all l-forms derivable from a given d-form s (by declension, inflection, etc.).

(b) *Analysis or stemming* ($f \rightarrow S(f)$): The “inverse” process of extracting from a given l-form f the set $S(f)$ of all possible d-forms from which it can be derived.

We also require from both processes that they yield the pertinent grammatical attributes relating to the derivation. Thus, for example, the attributes of an l-form include its gender (masculine, feminine), its number (singular, plural, “dual”), prepositions that are prefixed to it, etc.

3. Linguistic Applications

The realization of the grammatical relations embedded into the vocabulary tree and the dual processes of synthesis, $s \rightarrow F(s)$, and analysis, $f \rightarrow S(f)$, enable many and varied applications in linguistic research and information processing. Although in this paper we shall concentrate on information retrieval systems, we first mention several other possibilities.

Some obvious applications present themselves naturally in the context of automatic syntactical, semantical, statistical, and stylistic analysis of natural language texts. In stylistic analysis, for example, when evaluating the “richness” of a style, one is more interested in statistics on d-forms (or even roots) than on actual t-forms; also statistics on the distribution of adjectives and adverbial forms (or of nouns versus verbs) may give significant clues to authorship identification [14]. For this type of work the mapping $f \rightarrow S(f)$ is required.

Another example is the automatic compilation of concordances. Usually one would like to see in the concordance all words that are morphologically related (i.e. belonging to the same d-form or perhaps to the same root) clustered together. For highly inflected languages this is a nontrivial problem: The number of grammatical variants of any t-form is usually large, and due to mutations, prefixing, etc., they tend to spread all over any alphabetical list of the text. Therefore it is difficult to gather and collect them together. A solution to this problem based on the $f \rightarrow S(f)$ mapping (analysis) can be employed. For details see [12]

Mechanical translation is another instance in which both stemming and expansion are necessary, since the translation function is presumably given only for the d-forms of the source language. Thus any automatic translation process will involve the stemming of

an l-form f into its d-forms $S(f)$ and finding the set of equivalent d-forms $S' = \{s'_1, s'_2, \dots, s'_k\}$ in the target language, and then "expanding" the appropriate d-form $s'_1 \in S'$ into the corresponding l-form $f' \in F(s'_1)$. Moreover, a correct translation requires some grammatical manipulations, as for example when permuting the "adjective-noun" pattern in English to the "noun-adjective" pattern in Hebrew.

Turning now to the application of linguistics to information retrieval systems, we note that in a formatted management information system, e.g. manpower administration system, the format of information in the database is fixed and a request can be completely and exactly defined by either manual or automatic lookup in a table of codes. For example, the request "salary ≥ 1000 " yields precisely the personnel records for which the value in the salary field is not less than 1000. On the other hand, in a natural-language retrieval system, such as a selective dissemination information system, exhaustiveness and relevance are difficult to achieve not at the expense of one another. Whether such a retrieval system is "full-text" or "indexed" (see [10]), the user typically defines his search request as a set of forms and operators. It is here where linguistic problems arise.

For the illustration of various aspects of these problems, we shall use, as an example, the search topic "airplane hijacking."

Finding all synonyms of a term (in our example, "airplane," "plane," "airliner," "aircraft," "jet," etc.) is a well-known problem of any information system involving natural language (see [20]). We shall not deal with this problem here, nor with that of "textual equivalence" of phrases (such as "airplane hijacking," "hijacking of airplanes," and "hijacking of the crowded airplane") which are typically solved by requiring that the various components appear close together (e.g. "hijacking" and "airplane" should appear in the same sentence). (See [13, Sect. 3.4].) What we are interested in here is the problem of "grammatical equivalence" of forms, e.g. "hijack," "hijacking," and "hijackers," all of which are morphologically and semantically related to the d-form "hijack."

In the operation of an information retrieval system various processes may require either (or both) of the grammatical functions $s \rightarrow F(s)$ and $f \rightarrow S(f)$. We mention here briefly three such processes which served as the main motivation for creating KEDMA:

(i) *Local expansion* of a given d-form s into the set $F_1(s)$ of those l-forms derivable from s which appear in the database DB of the system ($F_1(s) = F(s) \cap \text{DB}$). This process is required in the retrieval stage of an information retrieval system because if the query to be answered contains the term "hijack" then it should be matched against the occurrences of all the grammatical variants of the stem "hijack" in the searched text. For this purpose the mapping $s \rightarrow F_1(s)$ is applied to the query terms. If desirable, it may be restricted to certain grammatical forms (e.g. the past tense of a verb or the plural of a noun) by using the grammatical information associated with each $f \in F(s)$.

(ii) *Content analysis* requires the mapping $f \rightarrow S(f)$. In content analysis one tries to mechanically find classes of related terms, either by global analysis of the whole database [15, 17, 18, 21] or locally by analyzing the texts retrieved in each step of an iterative search [1, 2]. Normally the various associated clustering processes, e.g. weightings and correlations, should be carried out with d-entries rather than with individual grammatical variants, which are only accidentally (so to speak) different.

(iii) *Query analysis* requires the mapping $f \rightarrow S(f)$ if the user of an information retrieval system is allowed to formulate his query in as close an approximation to natural language as possible. In our system the minimum requirement is that the user need not know the exact definition of a d-form while formulating his query as a set of keywords and operators. This is not a luxury, since the exact definition of a d-form in Hebrew (and probably in many other languages) is rather subtle. Therefore we want to relieve the user of the necessity to look up special dictionaries for each problematic case. Thus, when the term "hijacker" is given, the system automatically derives the

stem “hijack” from the form “hijacker” and then expands the stem “hijack” into the set F (“hijack”).

The manner in which KEDMA enables realization of these three applications to information retrieval is explained in Section 6.

The linguistic problems raised above are often disregarded in English text processing, probably because in English various simplistic solutions are quite satisfactory in most cases [16, 17, 19, 21, 22]. For one thing, the number of grammatical variants of any d-form is small. Second, an alphabetical sort of any English text groups together the grammatical variants of any t-form, since they are formed by adjoining suffixes. The number of exceptions (go–went, mouse–mice, etc.) is small. In many practical cases they can be ignored or handled manually. Thus, in particular, the above-mentioned problems of concordance construction, local expansion, and content analysis almost do not exist in English.

Hebrew (or Arabic) and English represent, in a way, the two extremes of computational linguistic complexity and the lack of it, respectively. In between them there is a wide spectrum of partial complexities.

For example, French provides an example of a language whose morphology is halfway—in terms of complexity—between that of Hebrew and of English. Unlike Hebrew, it does not allow much preposition prefixing, and in general conjugations and declensions do not affect the first letter of the word. On the other hand, most nouns and adjectives, for example, assume different forms in the masculine/feminine singular/plural cases. Following is a representative (and by no means exhaustive) list of derivation patterns for masculine/feminine: ami/amie (friend); beau/belle (beautiful); gardien/gardienne (keeper); epoux/epouse (husband/wife); veuf/veuve (widower/widow); menteur/menteuse (liar); directeur/directrice (manager); prince/princesse (prince/princess), etc. More importantly, however, the verb conjugation patterns are much more complicated in French than in English. The verb has to be conjugated (and thus usually changes form) by number (singular/plural), person (first/second/third), voice (actual/passive), tense (up to ten “tenses”), and six modes. Such a conjugation results in a few tens of variants (usually about fifty different variants even for the most common and regular verbs), and it is not uncommon that only the first one or two letters of the infinitive are retained in some of the variants: faire–fut (do), recevoir–recut (receive), etc.

Also, German and Russian display a typically inflected derivational pattern. Rather than going into details, we only give two examples which demonstrate some of the richness of the variants of d-forms.

In German: haltend, halte, halten, haltet, haltest, haelt, haeltst, hieltst, hielte, hielten, hieltet, hieltest, ... , hintanzuhalten, ... , zusammenhalten,

In Russian: delayet, peredelayet, delayu, delayesh, delayem, delayut, delal, delali, delala, ... , sdelayet, dodelayet, dodelal, sdelal, sdelayu, dodelayu,

For automatic text processing in these three languages and many more, a KEDMA-type approach may be useful.

4. General Structure of Hebrew Morphology

The general structure of Hebrew morphology can best be characterized by a few numbers. The total number of entries in a modern and comprehensive Hebrew dictionary [9] does not exceed 35,000 (including some 3500 “international” forms such as “democracy,” “symphony,” etc.), derivable from some 3000–5000 roots. In comparison the number of entries in a standard collegiate American dictionary is of the order of 150,000. On the other hand, the number of grammatical and spelling variants of each entry in a Hebrew dictionary is very high and may easily reach *twenty thousand* l-forms per verb for certain classes of verbs. This unusual pattern is due to the highly inflectional and derivational nature of the language.

Briefly speaking, nouns and verbs can be inflected or conjugated to indicate various attributes of gender, number, tense, mode, etc., producing what we term *kernels*.

The great majority of nouns and adjectives may have up to four different forms (singular/plural, masculine/feminine). These forms are derived from the singular/masculine form by adjoining to it certain suffixes. The derivation may be preceded by the deletion of some of the last letters of the singular form. A verb root (a stem of three, sometimes four, letters, usually the infinitive form of the verb) may be conjugated in up to seven BNYNYM (binyanim, “modes”), four tenses (past, present, future, and imperative), three persons, two genders, and two numbers. The conjugation is usually accompanied by the addition of prefixes, suffixes, and infixes (some of the resulting forms may, however, “overlap”). In some modes a “verbal noun” is also derived, somewhat similarly to the verbal noun in English (“going,” “writing,” etc.).

From most of the kernels one may derive *compounds* by adding suffixes to indicate ten possessive forms (mine, yours, etc.) for nouns, or ten causative forms (me, you, etc.) for verbs. The exact form of these suffixes depends on the person/gender/number of the pronouns, as well as on the grammatical status of the kernel to which they are suffixed.

Finally, one may prefix to most of these compounds a variety of prepositions or combinations such as B (in), L (to), M (from), ŠKŠH (that when the), taken from a list of about 100 combinations, thus generating all possible l-forms of the language

It should be emphasized here that the original d-entry may undergo quite radical metamorphoses during the derivation process outlined above. For example, the Hebrew form for “daughter” is BT (bat); the phrase “and when our girls” is given by the form WKŠBNWTYNW (ucheshebenotenu), whose parsing is W (and) + KŠ (when) + BNWT (girls) + YNW (ours), and which has retained only one letter from its original d-entry (the T of BNWT is the plural’s morpheme). Similarly, “to see” is “RAH” and “and since I saw him” “WMŠRAYTYHW” (umishereitihu), which has retained only the two letters RA of its original form.

Superimposed upon this structure is another difficulty, due to the fact that written Hebrew is essentially a nonvocalized language. This introduces a very serious problem of lexical ambiguity. (Just think, e.g. of the nonvocalized English form “pr,” which can mean: peer, per, poor, pier, pore, pour, pray, pure, pyre, etc.) In order to somewhat reduce this ambiguity, special letters (“matres lectionis”) are sometimes infixed, producing the “full” (ktiv male – KTIB MLA) versions of the form. Since there are no rigid rules for adding these letters, different variants can arise, all contributing to the large number of forms derived from one d-form.

The derivation process is depicted in Figure 2 as a generation tree, together with some rough estimates on the total number of elements on each level of the tree: roots, d-forms, kernels, compounds, and forms. As an example, some derivations for the root RAH (raah – to see) are given in Figure 3. (See Appendix I for the Hebrew transliteration convention.) Incidentally, note that a d-form is also a kernel, a kernel is a compound, and a compound is an l-form. The number of forms given for each level do not take into account the effect of overlaps which occur when the same string of letters is produced from different d-forms, resulting in homographs. The number of different strings on each level is in fact considerably smaller.

Hebrew, then, as well as other highly inflected languages, possesses the following properties:

(i) It has relatively few d-forms, inducing a large number of l-forms (a bottom-heavy vocabulary tree).

(ii) Grammatical and other variants of a given l-form (which are semantically close to each other) are scattered over the alphabetical list of forms because of the adjunction of prefixes and infixes.

(iii) Related l-forms may have only very few common letters – in extreme cases only

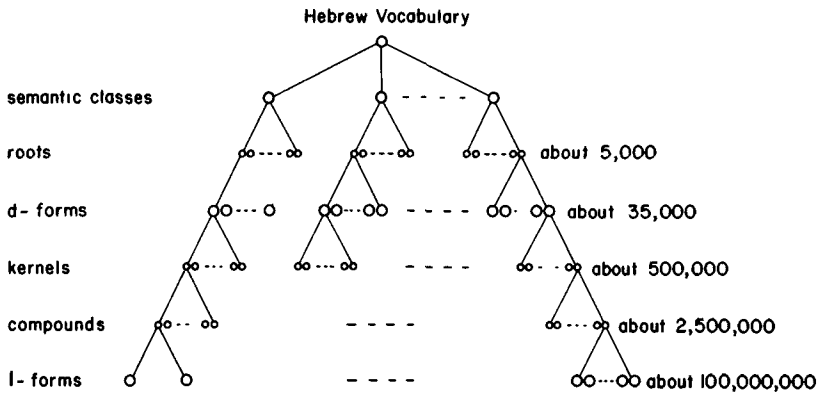


FIG 2 The Hebrew vocabulary tree. The numbers show that Hebrew morphology is characterized by a small number of d-forms and a large number of derived kernels, compounds, and l-forms.

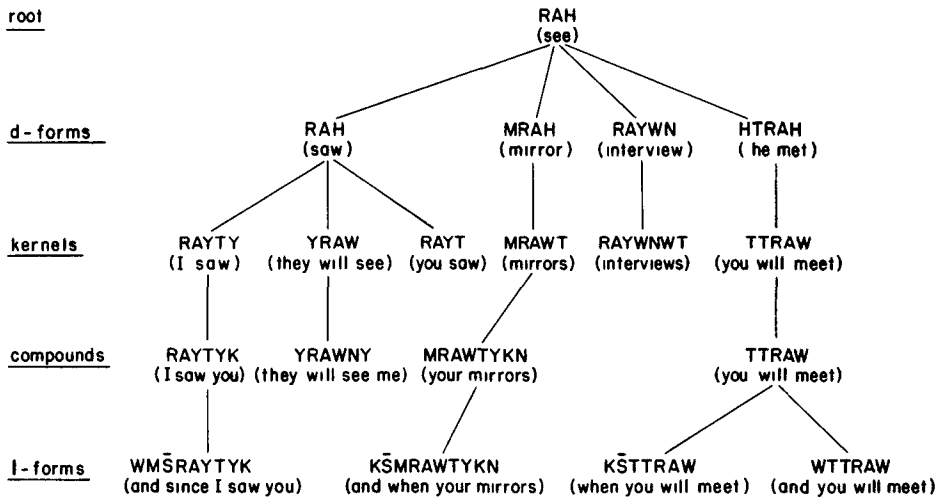


FIG 3 Some of the derivations of the root RAH (to see). It should be pointed out that most of the forms are highly homographic. Only one of the possible meanings is given in the translations on the tree.

one – owing to the adjunction of prefixes, suffixes, and infixes and the deletion of other letters during inflection.

(iv) The inflected nature of the language and the omission of all vowels in most of the Hebrew texts (in particular, in all of the Responsa literature) induce a very large number of homographs: about four per form on the average (and in extreme cases up to a few tens).

5. Synthesis, Analysis, and Their Fusion

As indicated in Section 2, both analysis and synthesis may be used to create the grammatical relations of the vocabulary tree. A feasibility study of morphological analysis of Hebrew (see [4, 8]) indicated that although such a process is feasible, it requires very accurate manually prepared dictionaries and quite delicate computer programs that would be too cumbersome and lengthy for practical implementation. The (dynamic) synthesis approach, which is easier to implement and less sensitive, has therefore been preferred. Algorithms were formulated for the automatic generation of the set $F(s)$ of all l-forms derivable from any given d-form s , requiring the user to

supply along with his keywords only the most rudimentary grammatical information. This process was implemented for the Responsa retrieval system in 1968, and has since been field tested and used in hundreds of searches. Only in very rare cases was it found that an l-form, derivable from a given keyword, was not generated by the synthesis programs.

However, the synthesis process does not provide the inverse relation (l-form to d-form), necessary for content analysis and other applications detailed in Section 3. Also, it is not easily applicable in an on-line interactive mode of operation: The method of generating all valid l-forms of a given d-form, which is followed by "killing off" most of them (since only a very small fraction actually appears in the given texts), becomes less and less attractive with increasing size of the database.

The dynamic expansion based on the synthesis process takes the following cumbersome form: Let $Q = \{k_1, \dots, k_n\}$ be the set of keywords for some query, $F(k_i)$ the set of all l-forms generated by dynamic synthesis for the keyword k_i , and $F_1(k_i)$ the subset of $F(k_i)$ consisting of the forms appearing in the database: $F_1(k_i) = F(k_i) \cap \text{DB}$. Typically, Q is on the order of 10–20 and the set $F(Q) = \bigcap_{k_i \in Q} F(k_i)$ is on the order of 100,000, in many cases about 500,000, l-forms. Therefore the dictionary lookup needed to restrict $F(Q)$ to the required set $F_1(Q) = \bigcap_{k_i \in Q} F_1(k_i)$, which is typically of the order of a few hundred, involves processing *hundreds of thousands* of items. Thus the set $F(s)$ is larger than the desired set $F_1(s)$ by three orders of magnitude, and although economical and rapid techniques have been created to carry out the "killing off" process [5], they are unsuitable for an on-line environment in a growing database which currently stands at 21 million words. For further details about the synthesis approach, see [6]. The analysis process, on the other hand, could enable the creation of the set $F_1(s)$ of a given d-form s without first generating the set $F(s)$.

The analysis process (l-form to d-entry) could be achieved, in principle, by synthesizing the vocabulary tree top down (d-form to l-form expansion), and then using the tree relations upward. Unfortunately, the size of the lowest level (l-form level) makes the construction and storage costs impractical. The two following facts can, however, give some hints about an optimal approach to the required process $f \rightarrow S(f)$:

(i) The "compound" level of the Hebrew vocabulary tree is of reasonable size (about two and a half million nodes).

(ii) The difficulty in analyzing a Hebrew l-form is in the compound to d-form step, while the analytical process of deriving compounds from l-forms is simple—basically prefix truncation with some grammatical cross-checking. It is, in principle, comparable to the stemming process in English.

Thus it is feasible to synthesize all of the compounds levels in a one-time process, and dynamically use l-form to compound analysis to link the l-forms and compounds levels. By combining partial synthesis (utilizing procedures similar to those developed and debugged for the Responsa project) with partial analysis (using relatively simple tables of prefixes), a feasible combined *synthesis-analysis* approach is achieved, which provides a complete morphological analysis of any given l-form. The synthesis of all compounds produces a Compounds to d-forms File (CD), which includes for each compound the d-form from which it is generated and the grammatical information pertaining to this generation. The analysis, on the other hand, is realized by means of a two-stage process:

(i) A partial analysis produces all possible splittings $f \rightarrow (p, c)$, where p ranges over all possible prepositional combinations as given in a preposition-mask table PM and c is the residual string of letters ("candidate compound"). Since f itself may be a compound, the "null splitting" ($p = \emptyset, c = f$) is also included in the above set of splittings.

(ii) Each candidate compound c is looked up in CD, and if found there, its grammatical attributes, as extracted from CD, are used to check whether the splitting is grammatically sound. If the conditions are satisfied, then (p, c) is accepted as a *valid splitting* of f . The set of all compounds from which f is derivable is thus given by $E(f) =$

$\{c:(p, c)$ is a valid splitting $\}$. Since the compound record in CD also contains the d-forms from which the compound is derivable, i.e. $S(c)$, the desired derivation $f \rightarrow S(f) = R(E(f))$ is obtained (see also Figure 4).

Both stages (i) and (ii) of the above process consult PM, which contains a row for every possible prepositional combination and a column for every possible grammatical attribute. The entry $PM(i, j)$ in the table is + if and only if the i th prepositional combination is compatible with the j th grammatical attribute. Stage (i) uses the PM-table during the splitting process to identify those combinations of the initial characters of the given form which are possible prefixes, and stage (ii) uses the grammatical attributes of the PM-table to check the grammatical validity of the proposed splitting. For example, the preposition KŠH (“when the”) cannot be prefixed to a verb in the past, future, or imperative, or to a noun in the pronomial or construct state.

Following is an example of such an analysis. The l-form WKŠMHŠBK (ucheshe-machshechem – and when your computer) can be split into the following (p, c) pairs:

- (a) (\emptyset , WKŠMHŠBK) (no such compound in CD),
- (b) (W, KŠMHŠBK) (“and” + no such compound in CD),
- (c) (WK, ŠMHŠBK) (“and like” + no such compound in CD),
- (d) (WKŠ, MHŠBK) (“and when” + “your computer”),
- (e) (WKŠM, HŠBK) (“and when from” + “your accountant” or “and when from” + “he thought you”).

The first three splittings are rejected since c is not found in CD. The second interpretation of splitting (e) is rejected because the grammatical attributes of “he thought you” do not permit prefixing “and when from.” We are thus left with the possible d-forms “computer” and “accountant” and the corresponding grammatical attributes. (The problem of deciding in a given context between the two possibilities has to do with the general and complex problem of ambiguity resolution, which is outside the scope of this paper; the interested reader may wish to consult [7] for this topic.)

By reversing the aforementioned process, the expansion mapping, from a d-form s into the set $F(s)$ of its derivative l-forms, is also divided into two stages. First a lookup in the d-forms to Compounds File (DC) (the inverted file of CD; see Section 6) yields the set $C(s)$ of all the compounds of s . A preposition synthesis algorithm is then applied to each compound $c \in C(s)$, by consulting the grammatical attributes of c in DC and the grammatical attributes in PM. This gives the set $G(c)$ of all l-forms derivable from

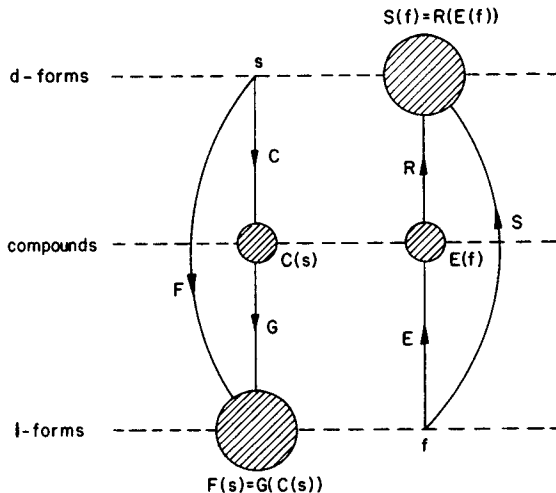


FIG 4 A scheme of the mappings $F(s) = G(C(s))$ from the d-forms to the l-forms, and $S(f) = R(E(f))$ in the inverse direction. The latter is effected via the “Compounds to d-forms File” CD.

c , leading to the required set $F(s) = G(C(s))$. For example, the d-form MĤŠB (machshev-computer) when looked up in DC is found to have a set of 20 compounds {MĤŠB, MĤŠBYM, ..., MĤŠBKM}. Prefixing the appropriate preposition combinations yields finally a set of 1020 derivative l-forms {MĤŠB, WMĤŠB, ..., KŠHMĤŠBYM, ..., KŠMĤŠBKM, ...}.

We conclude this section with a flowchart summarizing the expansion and stemming processes, using the synthesis-analysis approach (Figure 5). The vertical flow of the diagram represents the one-time process of the DC and CD production: After the MHD (Mechanical Hebrew Dictionary) is created, it is partially synthesized to generate the DC file which is sorted by compounds to produce the CD file. The horizontal paths represent the two dynamic processes of expansion and stemming as discussed earlier.

6. Linguistic Files

This section contains a description of global and local files which are needed for implementation of the synthesis-analysis process. The global files provide the up-down and down-up relations between the d-forms level and the compounds level of the vocabulary tree, respectively. The local files provide the same relations in a much more efficient way for any given database

The basic file, MHD (Mechanized Hebrew Dictionary), was constructed from an extensive modern Hebrew dictionary [9]. Expert linguists were assigned the task of coding, for every d-form in this dictionary, a complete lexical record: Each record of a nominal d-form contains the following information: a short semantic description, the root of the d-form, its part of speech, gender, and number, historical and linguistic layers (Biblical, Talmudic, medieval, modern), the one-letter prepositions which may be prefixed to the d-form and its derivative forms, the possibility of pronoun adjunction, codes for the generation of the feminine/dual/plural forms, construct state, and various

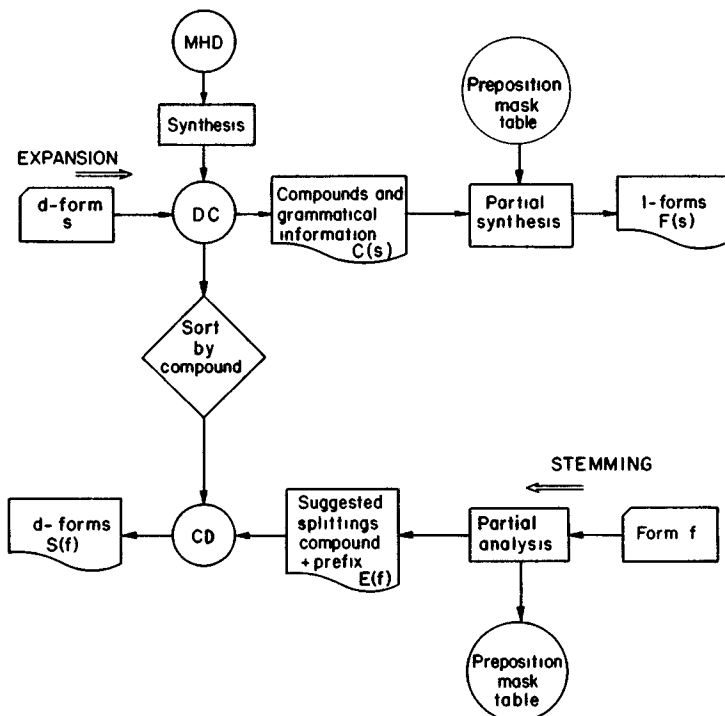


Fig 5 Flowchart summarizing the synthesis-analysis method for expansion and stemming

spelling variants. The record of a verbal d-form has in principle a similar structure, with some appropriate and obvious modifications. Different d-forms which are homographic but have different vocalizations are listed separately in the MHD by giving them different serial numbers.

Both the extensiveness of the files (their containing all d-forms) and the completeness of the grammatical information included are important, and determine the quality of their linguistic applications. On the other hand, it would be an altogether impossible task to prepare perfectly complete and accurate files. Not only would the MHD have to be complete, but every unusual and rare form, from Biblical to modern times, would have to be taken into consideration by the synthesis algorithms, or treated manually as a “special case.” Since our objective is not a perfect scientific tool for theoretical linguistic research but rather a practical and operational tool to serve current and projected information systems, it suffices if our files include “almost all” of the l-forms in use in medieval and modern Hebrew. The d-forms not appearing in the source dictionary may be added to the MHD from specialized dictionaries and on the basis of new texts, as it becomes necessary.

After some checkings and cross-checkings were made to eliminate clerical and linguistic errors, the grammatical synthesis programs were applied to the MHD to generate the auxiliary file DC (This process took about eight consecutive hours on an IBM 360/50 computer with a memory partition of 256 K bytes) The synthesis procedure of each d-form consists of several stages of forms generations: (i) d-form spelling variants, (ii) kernels – by declensions, (iii) compounds – by pronomial conjugations, (iv) compound spelling variants.

Sorting DC alphabetically by compounds gives CD, which is the main file. The records in CD contain for each compound: (i) the compound itself; (ii) its “ancestors,” i.e. the d-forms from which it was derived, together with their corresponding roots; each d-form and its corresponding root is identified by a code number; (iii) an exhaustive grammatical mask (GM), which details the grammatical attributes mentioned above. The CD file, which is sorted alphabetically by compounds, contains about 2.5 million records of 30 bytes per record (before compression).

An important technical point should be mentioned here. The file CD which is used for constructing the local files is too large and bulky even for batch processing. Accordingly, a compression procedure was applied to CD, the details of which will appear elsewhere. In its fixed length record organization CD required some 70 million bytes. In a variable length organization this is reduced to 32 million bytes. The compression techniques cut this down to only 17 million bytes, a further saving of 47 percent.

It is thus seen that the global files CD and DC together with the partial (prefix) synthesis and analysis provide a complete realization of the up-and-down relations of the vocabulary tree. For realizing the three applications to information retrieval mentioned in Section 3, a set of local files LDF, LFD, and SFD is derived from the global files for any given database. For the Responsa retrieval system, these files have the following form:

The “Local d-forms to t-forms File” (LDF) contains a record for every d-form s for which $F_1(s) \neq \emptyset$, i.e. s has some variant t-form in the database. The record of s contains all the derived t-forms which appear in $F_1(s)$, including forms with prepositions, together with the appropriate grammatical attributes. This file thus realizes the local expansion $s \rightarrow F_1(s)$.

The “Local t-forms to d-forms File” (LFD) contains a record for every form f of the database. Each record contains all the d-forms $S(f)$ of the language from which f is derivable, together with the appropriate grammatical attributes. This file thus provides the stemming of $f \rightarrow S(f)$ required for content analysis.

The “Saturated t-forms to d-forms File” (SFD) is an expanded version of LFD, where the stemming $f \rightarrow S(f)$ is given not only for t-forms of the given database but also

for l-forms of a subset of the vocabulary of the language. The subset is selected so as to facilitate natural-language query analysis.

A serious problem presents itself in the context of the local files when dealing with t-forms of the text, such as proper names, abbreviations, acronyms, misprints, etc. For such a form f , the analysis process yields $S(f) = \emptyset$. The correct processing of such forms, however, is of vital importance for the Responsa retrieval system (and, in general, for every “real-life” information system).

Basically, two special analysis processes are defined for handling such t-forms: (i) relating t-forms to existing d-forms, such as an abbreviation of an l-form to its d-form; (ii) relating a class of t-forms which are not l-forms to a “standard t-form,” such as the class of all variations of a proper name to a standard form of the name. Such a standard t-form – which is not a d-form – is defined as an *external d-form*. It is added to LDF and LFD together with its relations to the derived t-forms.

Correct handling of t-forms which are not l-forms is another reason for preferring the local files approach over the dynamic stemming and analysis techniques. As mentioned above, the local files can gradually be made more complete by the use of various complementary man-machine processes to add t-form \leftrightarrow d-form mappings. Such processes may in certain cases be too cumbersome, or practically impossible, to implement dynamically.

7. Concluding Remarks

The actual construction of the system of global files and the various programs for synthesis and analysis for the Hebrew language was a task which required considerable effort. The linguistic effort alone invested in constructing the MHD is equivalent to three man-years of work. The construction of the local files for the Responsa database (some 120 million characters currently) will be carried out on a second improved version of the MHD which has just been completed. A subset of the local files was constructed some time ago for a database of 1,300,000 words, and used for research in local clustering. The results of this work will appear separately [3].

The initial tests and uses of these versatile tools provided by KEDMA indicate that the effort invested in this project is well justified. The morphological analysis provided by KEDMA was used by two of the authors for researches on the automatic creation of concordances of Hebrew texts [12] and on local feedback techniques in an iterative search process [2]. We point out that the latter research included the application of the local clustering feedback process to a small database of documents in English (supplied by the National Bureau of Standards). For this work we were unable to find a tool for morphological analysis ready to be applied to any given text in the English language. (Special purpose, system tailored, stemming algorithms exist in several systems [17, 20, 21].) It is quite surprising that after almost twenty years of continuing research in information retrieval and computational linguistics, no really operational, reasonably general, and flexible linguistic tools of this kind seem to be available for the English language. One possible reason for this may be the fact that the creation of such a tool requires very patient and nonglamorous manual work; and for small databases the tool is dispensable. However, it is our strong belief that if research in information retrieval is to be of consequence to real-life problems and systems, it must also face those aspects of large operational systems, which, though trivial in principle, are certainly hard in practice.

Appendix I. Hebrew Transliteration

The Hebrew alphabet consists of 22 letters, all of which are consonants. Five of these, when appearing at the end of a word, are designated by special symbols and called “terminating letters.” Vowels are denoted by diacritical marks, but these are omitted in most texts (and all the texts considered here) Some consonants (B, K, P, T) transform

TABLE I HEBREW-ENGLISH transliteration

א	A	ה	H	ע	Ä
ב	B	ט	T	פ	P
ג	G	י	Y	צ	C
ד	D	כ	K	ק	Q
ה	H	ל	L	ר	R
ו	W	מ	M	ש	S
ז	Z	נ	N	ת	T
		ס	S		

into different consonants (V, CH, F, S, respectively) depending on the presence or absence of a special diacritical mark, which is also omitted in our texts.

To give the English-speaking reader as faithful a picture as possible of the problems caused by Hebrew, we adopt here a one-to-one transliteration of Hebrew letters to English letters (see Table I). Each Hebrew form in the paper has been written in up to three forms: the transliteration (in capital letters), followed by the pronunciation and the translation. For example: ŠLWM (shalom–peace).

ACKNOWLEDGMENTS. Many persons took part in the research effort which culminated in the work described in this paper. Dr. D. Schindler participated in the early stages of system design and programming supervision. The technical parts of the grammatical algorithms, as well as the actual codification of the Hebrew dictionary, were worked out by H. Baum and J. Guggenheimer, with the assistance of E. Landsman and Z. Linzer. Various parts of the programming were done by Y. Arnon, A. Meron, D. Harel, M. Moreshet, D. Raab, and E. Raban. The help of all of them is most gratefully acknowledged.

REFERENCES

1. ATTAR, R , AND FRAENKEL, A S Local feedback in full-text retrieval systems *J ACM* 24, 3 (July 1977), 397–417
2. ATTAR, R , FRAENKEL, A S , AND STEIN, J Local feedback in full-text English and Hebrew retrieval systems Tech Rep No 5, Inst for Inform Retr and Computat Linguistics (IRCOL), Bar-Ilan University, Ramat Gan, Israel, Feb 1976
3. ATTAR, R., FRAENKEL, A S , AND STEIN, J Local feedback in full-text iterative retrieval systems–II In preparation
4. CHOUKA, Y Automatic grammatical analysis of the Hebrew verb (in Hebrew) Proc 2nd Nat Conf Inform Process Assoc of Israel, Rehovot, 1966, pp 49–66, English abstract, *The Finite String* 4, 4 (1967)
5. CHOUKA, Y Fast searching and retrieval techniques for large dictionaries and concordances (in Hebrew) *Hebrew Computat Linguistics* 6 (July 1972), 12–32 (publ by Bar-Ilan U , Ramat-Gan, Israel)
6. CHOUKA, Y , COHEN, M , DUECK, J , FRAENKEL, A S , AND SLAE, M Full-text document retrieval, Hebrew legal texts (Report on the first phase of the Responsa Retrieval Project) Proc ACM Symp on Inform Storage and Retrieval, J Minker and S Rosenfeld, Eds , U of Maryland, College Park, Md , 1971, pp 61–79 Expanded version of this paper appeared as Heft 3, *Arbeitspapiere Rechtsinformatik*, J Schweitzer Verlag, 1972
7. CHOUKA, Y , AND DREIZIN, F Mechanical resolution of lexical ambiguity in a coherent text. Algorithms and experimental results Proc Second Int Conf on Comptng in the Humanities, Ottawa, Canada, 1976, paper 43
8. CHOUKA, Y , AND SHAPIRO, M Machine analysis of Hebrew morphology potentialities and achievements (in Hebrew) *Leshonenu* 27 (1964), 354–372, English abstract, *The Finite String* 3, 5 (1964), 6
9. EVEN-SHOSHAN, A *The New Dictionary*, 7 volumes Kiryat-Sefer, Jerusalem, 1956
10. FRAENKEL, A S Legal information retrieval In *Advances in Computers*, Vol 9, FL Alt and M Rubinoff, Eds , Academic Press, New York, 1968, pp 113–178
11. FRAENKEL, A S All about the Responsa Retrieval Project you always wanted to know but were afraid to ask Expanded summary, Proc Third Symp on Legal Data Processing in Europe, Oslo, 1975, Council of Europe, Strasbourg, 1976, pp 131–141; also in *Jurmetrics J* 16 (1976), 149–156, and *Informatica e Diritto II* (1976), 362–370
12. FRAENKEL, A S , AND SPITZ, E Automatic construction of Hebrew concordances with ramification to

- English concordances. Tech. Rep. No. 1, Inst for Inform. Retr and Comput. Linguistics (IRCOL), Bar-Ilan University, Ramat Gan, Israel, June 1975.
13. GEBHARDT, F. Recent results related to JURIS Proc. Thrd Symp. on Legal Data Processing in Europe, Oslo, 1975, Council of Europe, Strasbourg, 1976, pp. 25-41
 14. HERDAN, G. *The Advanced Theory of Language, Choice and Chance*. Springer-Verlag, 1967
 15. LESK, M E Word-word associations in document retrieval systems. *Amer. Documentation* 20 (1969), 27-38.
 16. MINKER, J. Information storage and retrieval—a survey and functional description. Tech. Rep TR-369, Comptr Science Dept , U of Maryland, College Park, Md , April 1975
 17. MINKER, J , PELTOLA, E , AND WILSON, G.A Document retrieval experiments using cluster analysis *J Amer Soc Inform Sci* 24 (1973), 246-260
 18. MINKER, J , WILSON, G A , AND ZIMMERMAN, B H An evaluation of query expansion by the addition of clustered terms for a document retrieval system *J Inform Stor. Retr* 8 (1972), 329-348
 19. SALTON, G *Automatic Information Organization and Retrieval* McGraw-Hill, New York, 1968
 20. SALTON, G *Dynamic Information and Library Processing* Prentice-Hall, Englewood Cliffs, N J , 1975
 21. SPARCK-JONES, K. *Automatic Keyword Classification for Information Retrieval*. Butterworth, London, 1971.
 22. SPARCK-JONES, K , AND KAY, M. *Linguistics and Information Science*. Academic Press, New York, 1973

RECEIVED JULY 1976, REVISED JULY 1977