SOFTWARE—PRACTICE AND EXPERIENCE, VOL. 23(6), 567–588 (JUNE 1993)

Lexical Analysis of Inflected Arabic Words using Exhaustive Search of an Augmented Transition Network

AHMED A. RAFEA*
Computer Science Department, American University in Cairo, 113 Kasr El-Aini Street, Cairo, Egypt

AND

KHALED F. SHAALAN
Computer Science Department, Institute of Statistical Studies and Research, Cairo University, 5 Tharwat Street, Orman, Giza, Egypt

SUMMARY

This paper presents a lexical analyser for inflected Arabic words. An augmented transition network (ATN) technique was used to represent the context-sensitive knowledge about the relation between a stem and inflectional additions. An exhaustive-search algorithm is developed to traverse the ATN, generating all possible interpretations of an inflected Arabic word. The arcs of the ATN are augmented with rules containing conditions and actions. More than one rule is associated with some arcs. The states of the ATN are represented by Pascal procedures.

KEY WORDS: Arabic Natural-language processing Lexical analysis Morphological analysis Heuristic search Automatic translation Augmented transition networks

BACKGROUND

Arabic linguistics came into being in the eighth century with the beginning of the expansion of Islam.[1] This early start can be explained in terms of the tremendous need felt by the members of the new community to know the language of the *Koran*, the holy book of Islam, which had become the official language of the young Islamic state. The modern linguistic revolution has stimulated a number of scholars to apply existing theories as descriptive models for Arabic language. Transformational grammar is reflected in References 2–5, lexical function grammar in Reference 6, functional grammar in Reference 7, case grammar in Reference 8, and definite-clause grammars in References 9 and 10.

The first attempts at the automatic processing of Arabic were devoted to the solution of theoretical and practical problems such as root and pattern description,[11] alphabetization,[12] lemmatization,[13] morphological analysis and indexing,[14] and lexical,

* On leave of absence from I.S.S.R., Cairo University.

0038–0644/93/060567–22$17.00

Received 12 June 1990
Revised 11 March and 15 July 1992

syntactic and idiomatic concordancing.[15] More recently, with the trend towards Arabization of computer usage in Arabic-speaking countries, the automatic analysis and synthesis of Arabic-language data has begun to focus on components of syntax, i.e. from the phonological level it has moved to the morphological and the syntactic levels. Even so, most of the contemporary work in the field has been at the word level or below. Examples of such research have been on character recognition,[16] phonological synthesis,[17] morphological generation[18] and analysis.[19–21]

MATERIAL RELEVANT TO WORD MORPHOLOGY

We consider the following methods to be relevant in solving the Arabic morphology problem:

(a) finite-state machines[22,23] or FSMs
(b) affix grammars[24–26]
(c) augmented transition networks (ATN)[27]
(d) heuristic algorithms.[1,28]

Linguists in general, and computational linguists in particular, do well to employ FSMs whenever possible. They are theoretically appealing because they are computationally straightforward and best understood from a mathematical point of view. Hence they make for simple, elegant and highly efficient implementations. Lately, so-called finite-state morphology in general, and two-level morphology in particular, have become widely accepted as paradigms for the computational treatment of morphology.[22,23] Finite-state morphology appeals to the notion of a finite-state transducer, which is simply a classical finite-state automaton. The term two-level morphology, however, is used in a more restricted way, to apply to a system in which no intermediate forms are posited, even in the original grammatical formalism. The writer of a grammar using a two-level formalism never needs to think in terms of any representations other than the lexical and surface ones. However, the levels of morphology is an important issue in natural-language-processing systems. The two-level morphology was first devised by Koskenniemi[29] for the Finnish language. For Arabic, Farghaly[30] has suggested that three levels of morphology are needed: root, stem and word. Jaccarini[23] succeeded in developing a morphological parser for unvocalized written Arabic. The main object was to get the root out of the word without using a lexicon. However, this approach may not be suitable if understanding is considered. The word ‘هـــــم’ when it appears in unvocalized text can have more than one meaning; if morphology only is considered, without a lexicon, problems may be encountered. Kay[22] has demonstrated a theoretical framework for non-concatenative finite state morphology, but problems with the full range of Arabic roots are not considered.

Ditters[26] has introduced a morpho-syntactic analyser called AIMS. The formalism used is that of a two-level grammar, the extended affix grammar (EAG), whose first level describes non-terminal names. The second level describes the attributes, the features attached to non-terminal names. The formalism itself has been described by Koster[24] and Meijer.[25] However, the backtracking and register capabilities of the two-level formalism do not seem to be supported adequately by current hardware and software developments. The addition of 30-odd affix variables with a limited (even binary) domain of values makes the real-time analysis of large set of data a rather frustrating and technically unacceptable waste of CPU time. A solution may possibly be to extend the formalism with another level which contains the semantic information.[31]

Gheith[27,32] and Selim[33] have introduced a morphological analyser that represents morphological rules (grammar) as an augmented transition network (ATN). This work does not suggest any solutions for one-to-many mapping of words, nor has it raised the problem of erroneous breakdown of an inflected word.

Rafea[28] has produced another morphological analysis to be used in automatic understanding of inflected Arabic words. The morphological analyser is supported by an algorithm to remove the additions joined to the word and check whether the end-result makes sense. The process is repeated if necessary. Stems are stored in their different forms, if any, in a lexicon, which includes entries that cannot stand alone and have no meaning without additions. Even with the size of a lexicon that contains such additional material, this approach is much better than storing the inflected words, which is impracticable for a non-trivial (i.e. realistic) natural language.

The work presented here assumes a three-level morphology, namely the inflected word, the stem and the root, as already mentioned.[30] In place of the root itself, our approach is to extract the stem from the inflected word. This is because the stem is the meaningful entity for word understanding.

In a comparison of the work reviewed above with our own work, the following points can be noted:

1. Our basic technique is to use ATNs to analyse an inflected word, and to break it down into a stem and an addition. It is similar to the approach of Gheith,[27] with the difference that one-to-many mappings of words are avoided and erroneous breakdown of an inflected word is minimized.
2. Our main goal is to understand the inflected word as it appears in Arabic written text and to resolve ambiguity.
3. We regard the meaning of the stem and its additions as being central to the unambiguous understanding of inflected words as they appear in Arabic written text.
4. Rules relating a stem to its additions are built by induction and used to resolve ambiguities.

THE PROBLEM OF BUILDING A LEXICAL ANALYSER FOR ARABIC

Nature of Arabic words

Word formation in Arabic involves three concepts: root, pattern and form.[34] Word forms (e.g. verbs, verbal nouns, agent nouns, etc.) are obtained from roots by applying derivational rules to obtain corresponding patterns.Generally, each pattern carries a meaning which, when combined with the meaning inherent in the root, gives the goal meaning of the lexical form. For example, the meaning of the word form ‘كاتب’ (writer) is the combination of the meaning inherent in the root ‘كتب’ (write) and the meaning carried by the pattern ‘faal’ (ف–ا–ع–ل) which is the pattern of the doer of the root. The set of lexical forms constructed from the same root constitutes what is traditionally called a ‘morpho-semantic field’. All the members of this field share the same basic meaning that is inherent in the root, and are semantically distinguished according to the meaning carried by the patterns by which they are constructed. Thus, Arabic has the characteristics that from one root the derivational and inflectional systems are able to produce a large number of words (lexical forms) each having specific patterns and semantics.

Lexical-analysis functions

The main function of a lexical analyser is to break down the input stream, which is the inflected word, into lexical items or morphemes. A morpheme is the minimal meaningful unit in a language. If the morpheme can function alone, such as the word 'مـــهـنـدس' (engineer), it is called a free morpheme. Other morphemes cannot be used by themselves, such as the general plural ending 'ون' and the letters 'ون' in 'مـهـنـدسـون' (engineers). Such morphemes are called 'bound'. Bound morphemes, in Arabic, serve as additions at the beginning or ending of a stem. Using the definitions of free and bound morphemes, a word can be defined as a single free morpheme, and an inflected word can be defined as a complex form which is a single free morpheme combined with one or more bound morphemes.

Technical difficulties

An Arabic word can be represented as follows:[28]

$$[\text{Begin\_1} \mid \text{Begin\_2}] + \text{Stem} + [\text{Last\_1}] + [\text{Last\_2}] + [\text{Last\_3}]$$

where any term between the square brackets may be absent or occur once, and '|' stands for alternatives. The word is scanned for different additions, and what remains is the stem. Unfortunately, the process is not as simple as it may seem. One cannot simply look up the additions in a list and compare them with the beginning and ending characters of an inflected word. The sources of difficulties can be summarized as follows:

1. There may be overlapping characters within the additions that occur before a stem, and this may also happen in the additions following the stem. For example, the overlap between the ending additions 'تا' and 'ت' in the stem 'أخذ' (took) can be found in its inflected form 'أخـــذت' where the agent is a singular first person (I took), and 'أحذتا' which is the inflected form where the agents are dual second person.
2. There may be overlapping characters between the additions and a stem.
3. The stem itself may be modified due to the addition of the inflectional symbols. Consequently, some modification actions may be required, after the removal of the additions, and before searching the stem in the lexicon. For example, in the stem 'قال' (said), its middle letter can be converted in the inflected word 'يقول' (saying—says).
4. More than one interpretation may arise for a stem and its additions.

METHODOLOGY

A very simple methodology, which is used in small-scale natural-language-processing systems, is to store *all* the inflected words in the lexicon.[35] This solution is not efficient, e.g. because the stems of many variant forms of words can be derived by simple spelling rules such as removing the 's' of a noun in the plural form (as in English) or by removing the Arabic characters 'و' (waw) and 'ن' (noun) from a noun in the plural form in Arabic.

The motivation of the work presented here was to find the most efficient approach to designing and implementing a system of software to solve the problems introduced in the previous section. To arrive at this approach, we have to answer these questions:

1. Is it better to store different forms of stems, if it is necessary, or just store one form? The following example can clarify this point: the stem 'صحراء' (desert) can be stored under '«19»' and 'صحراو' because the latter form will appear as a result of removing the addition 'ان' from the dual form 'صحراوان'. If the lexical analyser is to be kept simple, the form 'صحراو' must be stored.
2. How can context-sensitive knowledge about the syntax of building an inflected Arabic word be represented in an efficient way?
3. How can ambiguities (different interpretations of an inflected word) be resolved?
4. what is the appropriate algorithm for the lexical analyser?

The answers to these questions are presented in the following section.

Lexicon entry

To answer the first question, a methodology is adopted to satisfy some constraints, which we find very important, on the entries in the lexicon. These constraints are

1. The word, at any entry, must be meaningful. (If we store the stem 'صحراء' in the above example, the stem form 'صحراء' is meaningful but the stem form 'صحراو' is not).
2. Retrieval of the form of the stem stored in the lexicon from an inflected word using this stem must be governed by rules that can be applied in an efficient way. This leads to storing the broken plural form, as there are no regular rules to transform a word in plural form into its singular form. Even if artificial rules can be used to obtain the singular form easily from a plural form in some cases, this may be very difficult in other cases. Consequently we take the decision to store all the broken plural forms for homogeneity and consistency.
3. As Arabic is a language that allows many words to be constructed by derivation from others, most researchers in Arabic morphology use a lexicon in which roots are the only entries.[36] This leads to substantial processing effort to obtain the root from a derived word. For example, the words such as 'أستعمل' (to use), 'عامل' (worker—factor), 'تعامل' (to deal with), 'معمل' (laboratory), 'استعمال' (the use), 'عميل' (customer), 'عملة' (currency—coin), 'عمالة' (labour force), 'أعمال' (jobs—business), 'عمولة' (commission), 'معاملة' (treatment) and 'عملى' (practical) are all looked up under the root 'عمل'. If the goal of the language-processing exercise is understanding, another lexicon may be created that contains derived nouns because the relation between derivational rules and the semantic change may be very deep and difficult for coding, as shown in the above example. Therefore, we have decided to store the derived nouns as separate entries in the lexicon. This will serve two purposes:

(a) less processing will be needed to reach the stem, which is a derived noun, as it is stored explicitly in the lexicon
(b) only one lexicon will be used for lexical analysis and understanding the meaning of the derived noun. Consequently, there will be no need for another lexicon to be used for understanding after morphological analysis. This is likely to lead to an overall saving in the storage needed for lexical material.

Context-sensitive knowledge representation

The second question addresses the method used to represent context-sensitive knowledge about an inflected word. We choose to use the ATN[37,38] as the basic structure for representing syntactic knowledge for building inflected Arabic words. This has enabled us to implement the following:

(a) placing the context-sensitive knowledge as rules associated with the arcs of the ATN. The rules have both condition and action parts. More than one rule may be associated with one arc. One of these rules is to be selected whenever the arc is traversed.
(b) getting all possible interpretations of an inflected word by searching the ATN exhaustively along different possible arcs.

Resolving ambiguity

After obtaining all possible interpretations of an inflected word, one has to face the problem of how selection of the right one may take place. One solution of the ambiguity problem is to defer it to the sentence-analysis phase. Here we have found that the ambiguity can be resolved to some extent by a following phase that checks the attributes of the stem, the omitted additions, and the modifications that have occurred in the stem. This checking is done through a set of rules. The rules are formulated by combining the stem attributes, omitted additions and/or stem modification. For example, the following rule relates a verb stem to its last additions:

IF The stem is a verb AND
Last_2 = 'ت' (tea) AND
the ending letter is doubled
THEN fill out the attributes values:
Singular, Feminine, Third Person for the agent.

To show how the above rule can be used to resolve ambiguity, the inflected word 'هممت' (started) can be taken as an example. The output of the lexical analyser for the inflected word 'هممت' is:

1. Stem: 'هـــم'
2. Last_2: 'ت'

In the lexicon there are three entries for the stem 'هـــم'. One is a verb (to start) in which the ending 'م' is doubled. A second is a noun (affliction). The third is a pronoun (they). This multiple-entry phenomenon is called polysemy. Applying the above rule to the output of the lexical analyser, only one interpretation is reached (conclusion of the rule). The other two interpretations of the stem will be rejected because no rule will be fired for those interpretations.

The algorithm of the lexical analyser

The lexical analyser traverses the ATN in a depth-first manner and backtracks to search exhaustively for all possible solutions. We have found that this is the best way to implement an Arabic lexical analyser. In effect, the conditions associated with the

arcs are placed in such a way that the arc to be traversed first is the one that leads to the most probable solution. This ordering is done using intuitive and heuristic measures. As shown in Figure 1, the arcs emitted from state S_0 are ordered such that a solution is sought by omitting 'begin' (the first recognizable prefix part of a word) and repeating this process on the rest of the word in case of failure. We use this order of search because the occurrence of nouns is more frequent than the occurrence of verbs. To clarify the method, the word 'فجر' (dawn) will be taken as an example to show how the order of traversing the ATN will affect the output of the lexical analyser. If the arc labelled 'omit begin' is traversed first, the following is the output:

1. Stem: 'جر' (which is a verb that means 'draw' or 'drag')
2. Begin_2: 'ف' (which is a particle indicating coupling).

However, if the arc labelled 'match' is traversed first, the following is the output of the lexical analyser:

1. Stem 'فجر' (one possible entry for this stem in the lexicon is a noun which means dawn).

This example shows the importance of ordering in a deterministic processing of the ATN. However, non-determinism is a feature of natural-language processing. Therefore, we have to ensure that generating all solutions remains possible in our system.

IMPLEMENTATION

We have written our system in Pascal, running in the VM/CMS operating environment for an IBM/370 4361. In the following sections, we discuss the implementation of the lexical analyser.

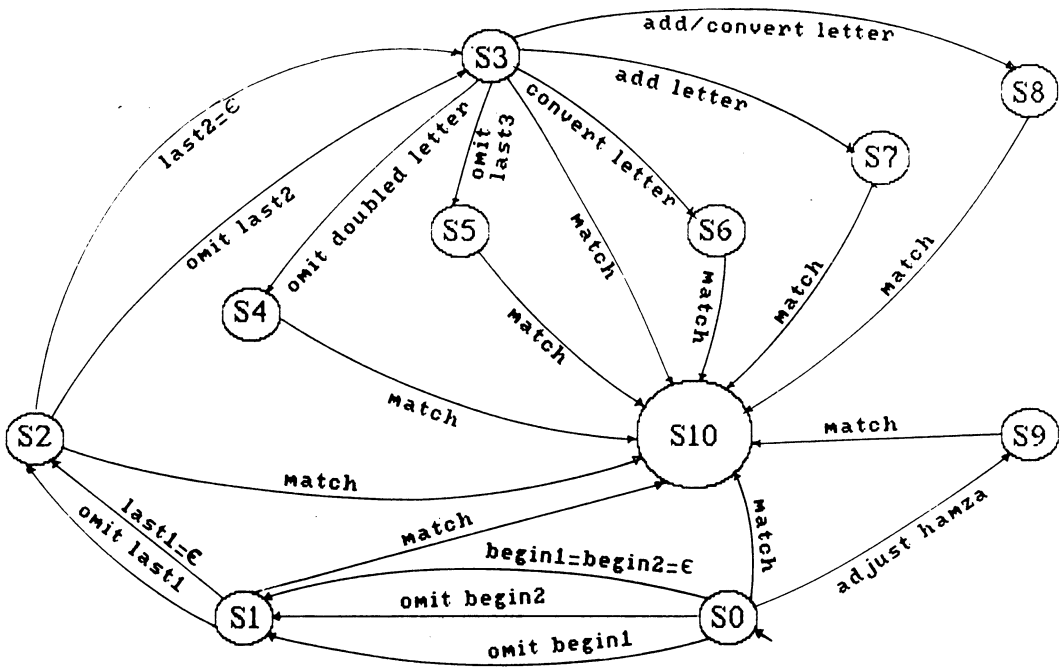

Figure 1. ATN representing the relation between the additions and stem of an inflected Arabic word

The representation of Arabic characters

The writing system for Arabic allows words to be written as linear strings of characters from right to left. Usually there are separators between words. However, in some situations two words may be written without any separation. Analysing all types of words that are not separated by any special character, we have found that some of their particles cannot be separated from the following word, such as 'كلـ'; 'كـــ'; 'فلـــ'; 'فـــ', . . . (bound morpheme), while some others can be written separately without violating Arabic writing rules, e.g. 'لا'; 'و', . . . (free morpheme), although it is common to write them without any separation from the following word. Although our approach can handle both types of particles by including all of them in the set of possible prefixes for stems, a decision has been made not to include any particle that can be written as separated from the following word. This decision is made because some of these particles, such as the coupling particle 'و', are used extensively, so that their addition to the set is likely to lead to an unacceptable increase in the time needed for the analysis of any word that begins with the letters of such particles. This decision leads to the following assumption: 'While scanning the input text, any group of characters that represent a word and can stand alone must be written as separated from the following one'.

A character in Arabic is generated relative to its context in a word (beginning, stem part, or ending). Context-sensitive rules for writing an Arabic word may make the pattern-matching difficult since the character may have more than one form. A character like 'د', has one form, while two forms ('س'; 'ســـ'), three forms ('هـ'; 'ـهـ'; 'ه') or four forms ('ـعـ'; 'عـــ'; 'ـع'; 'ع') can also occur.

For the computer that we have used, Arabic letters are available on the keyboards of terminals, and an internal representation of the Arabic character-set is provided. Pascal programs can express Arabic characters as string constants, which can be stored in variables of type character. Internal representations of some single characters vary according to the different possible forms in Arabic usage, e.g. the internal representation of 'ســـ' is different from the internal representation of 'س'. To solve this machine-dependent problem, we write a special routine that converts any form of a character into a standard form, which is the isolated-character one. The lexical analyser uses this routine while reading any inflected Arabic word.

The lexicon is represented as a lexically-ordered binary tree on a file, and is used by the matching routine that searches for the existence of a given stem in the lexicon. The stem is searched for in its Arabic-script (i.e. Arabic-orthography) version.

The augmented transition network

An augmented transition network (ATN), which describes the relation of an Arabic stem and additions, is built according to Figure 1. Every state of the ATN is implemented by a Pascal procedure. A transition from a state S_i to an adjacent state S_j is implemented by calling the procedure corresponding to S_j from the procedure representing S_i. The condition on an arc is implemented by a Boolean function called from a procedure representing a state to decide if a transition to an adjacent state is possible or not. The action associated with this arc, if any, is called from the function representing the condition. If an arc has more than one rule associated with it, the Boolean function representing the condition will scan the conditions of the rules associated with

the arc in a fixed order. If one of the conditions is satisfied, the corresponding action is activated. Actions associated with arcs are implemented as procedures.

Data structure associated with the ATN

The conditions and actions associated with the arcs use global registers and a set of status flags. Five registers correspond to the additions as classified in the section on methodology above. Each of these five registers corresponds to one of the classes of additions in Figure 1: Begin_1, Begin_2, Last_1, Last_2 and Last_3. The sixth register is reserved for storage of the current assumed stem. The initial value stored in this register is the whole inflected word. The status flags indicate the modifications made to the stem. Any such modification can be described by four parameters:

(a) the action taken (addition, conversion, omission)
(b) the position of the added, converted or omitted letter(s)
(c) the letter to be converted (held currently in the stem register)
(d) the letter to replace the one to be converted in a conversion process or the letter to be added in an addition process.

The combination of these parameters requires 17 flags. The registers and the flags are represented as a record in Pascal. A possible solution of the inflected word will fill the registers and set the flags according to the interpretation made by the lexical analysis. In order to keep all interpretations, an array of this record type is declared, and then used as the interface between the lexical-analysis phase and the subsequent phases.

Rules associated with the arcs of the ATN

These rules can be classified into six types. The first type contains only one rule, while each of the other types contains a set of rules.

1. The rule associated with the arcs leading to the final state. This rule is to match a stem against the stems stored in the lexicon.
   An example of this rule is:
   IF The stem is matched with a lexicon entry
   THEN Retain this situation as a possible solution

2. The action part of a rule of this type is to omit an addition from the stem register (containing the inflected word), and fill the appropriate addition registers. The condition of any such rule matches the first or last letters of the inflected word against one of the addition sets. All rules associated with arcs labelled with omission actions are of this type (see Figure 1).
   Instances of these rules are:

   IF Initial characters of the inflected word are in Begin1_set
   THEN Omit these characters and keep them in begin1_register;
   IF Last characters of the inflected word are in Last2_set
   THEN Omit these characters and keep them in last2_register.

For example, the application of these two rules to the inflected word ‘الناخبون’ yields:

(a) begin1_register :'ال'
(b) last2_register : 'ون'
(c) stem_register : 'ناخب'

3. The action part of a rule of the third type is to convert one of the stem letters to its original form (as stored in the lexicon), and set the appropriate status flag. The condition of such a rule depends on finding a specific letter at a certain position and the value of the omitted addition. This set of rules is associated with the arc labelled with the 'conversion' action (see Figure 1). An instance of this kind of rule is:

IF Last letter stored in stem_register is 'و' AND
((last2_register = 'ان' OR (last2_register = 'ون') OR
((last2_register = 'ات'))
THEN Convert 'و' to 'ء'

For example, the application of this rule to the inflected word 'صحراوان' yields:

(a) last2_register :'ان'
(b) stem_register :'صحراء'
(c) Status_Flags :'و' is converted to 'ء', at the last letter of the stem.

(Note that the inflected word 'صحراوان' has to be broken down into 'صحراو' and 'ان' before satisfying the condition of the above rule).

4. The action part of a rule of this type is to add letters to the unrecognized stem at a certain position to substitute for the omission of these letters due to the inflectional process. The condition of such a rule can be testing the length of the unrecognized stem, or testing that the omitted addition has a certain value. A very special action takes place if all other conditions fail. This action is to add a specific letter 'yell' which is sometimes omitted from a class of Arabic nouns (defective nouns) if the noun appears at a certain position in the sentence without being inflected. This set of rules is associated with the arc labelled with adding actions (see Figure 1). In this class, we have three types of rules:

(i) rules to add one or more of the following letters depending on the length of the unrecognized stem: 'ا'; 'و'; 'أ'; 'ي'. An example of this rule is:

IF The length of the unrecognized stem stored in Stem_register is two letters
THEN Add 'و' at the beginning of the stem.

As an illustration, the application of this rule to the inflected word 'عد' produces:

(a) stem_register :'وعد'
(b) Status_Flags :'و' is added, at the first letter of the stem.

(ii) rules to test the omitted addition before adding any letter. An example of this rule is:

IF ((last1_register = 'وا') OR (last1_register = 'ي') OR (last2_register = 'ون') OR (last2_register = 'ين') OR (last3_register = 'و'))
THEN

Add ‘ي’ last.

As an illustration, the application of this rule to the inflected word ‘مصطفون’ produces:

(a) last_register :‘ون’
(b) stem_register :‘مصطفى’
(c) Status_Flags :‘ى’ is added, at the last letter of the stem.

(Note that the inflected word ‘مصطفون’ has to be broken down into ‘مصطفى’ and ‘ون’ before satisfying the condition of the above rule).

(iii) rules to add ‘ى’ in a heuristic manner to recognize that the last ‘ى’ of a defective noun is omitted in some situations. These arise when all possibilities are investigated and the stem is still not recognized; e.g. if the inflected word is ‘محام’, then adding ‘ى’ at the end of the stem will produce:

(a) stem_register :‘محامى’
(b) Status_Flags :‘ى’ is added, at the last letter of the stem.

5. The action part of a rule of this type is to convert one of the stem letters and to add a letter to an unrecognized stem at a certain position. The condition of a such rule depends on finding a certain letter at a certain position and the value of the omitted addition. This set of rules is associated with the arc labelled with the addition/conversion action in Figure 1. An example of this rule is:

IF Middle letter stored in stem_register is ‘يـ’ AND
((begin2_register = ‘أ’) OR (begin2_register = ‘نـ’) OR
(begin2_register = ‘يـ’) OR (begin2_register = ‘تـ’) OR
(begin2_register = ‘سأ’) OR (begin2_register = ‘سنـ’) OR
(begin2_register = ‘سـ’) OR (begin2_register = ‘ستـ’) OR
(begin2_register = ‘فسأ’) OR (begin2_register = ‘فسنـ’) OR
(begin2_register = ‘فستـ’) OR (begin2_register = ‘فسيـ’) OR
(begin2_register = ‘ ’) OR(begin2_register = ‘فلتـ’) OR
(begin2_register = ‘فليـ’))
THEN Convert ‘يـ’ to ‘أ’, middle AND add ‘أ’, first.

For example, the application of this rule to the inflected word ‘يستطيع’ produces:

(a) begin2_register :‘يـ’
(b) stem_register :‘استطاع’
(c) Status_Flags :‘يـ’ Middle converted to ‘ا’ AND ‘أ’ is added, at the first letter of the stem.

(Note that the inflected word ‘يستطيع’ has to be broken down into ‘ستطيع’ and ‘يـ’ before satisfying the condition of the above rule).

6. A rule of the sixth type takes care of the letter ‘hams’. This letter has different forms and requires very special treatment. The action part of such a rule handles this special letter in the different situations that are distinguished in the rule’s condition part. This set of rules is associated with the arc labelled as ‘adjust hamza’ (in Figure 1). An example of a rule of this class is:

IF The first letter stored in stem_register belongs to the hamza_set AND the second letter in the stem_register is 'ا' alpha
THEN Replace the hamza and the 'ا' (aleph) by 'أ'

For example, the application of this rule to the inflected word 'يؤاخذ' gives:

(a) begin_register: 'يـــ'
(b) stem_register: 'أخذ'

(Note that the inflected word 'يؤاخذ' has to be broken down into 'ؤاخذ' and 'يـــ' before satisfying the condition of the above rule).

The rules associated with one arc are grouped into a single boolean function. Such a function will select one of these rules according to its premise, and activate its corresponding action. The rules are deterministic, i.e. no more than one rule will be fired at a time. Four common conditions are implemented as boolean functions with appropriate parameters to generalize their usage. Three actions are implemented, as procedures: omissions, conversions and additions.

Recognition of the stem and its additions

The recognition of the stem is made by traversing the ATN searching for a possible path from the initial state to the goal state. The searching algorithm is implemented by representing each state as a procedure. The procedure representing the initial state S_0 can be considered as the main procedure of the search. The order in which the arcs are examined will affect the order of the generated solution. We have tried to arrange this order in such a way that most probable interpretations (as noted by examination of substantial samples of Arabic text) are generated first. This ordering is not important in our present work, where all possible interpretations are generated. However, if limitations of time in some application mean that only one solution is to be generated, the ordering becomes important.

A transition from a state to an adjacent state is achieved by calling the procedure that represents this adjacent state if the condition associated with the linking arc is satisfied. The series of procedure calls forms a path from the initial state to the final state.

As all possible solutions are searched, simply reaching the goal state will not terminate the search. Backtracking takes place to the most recent activated procedure representing a state in order to look for another path through the different arcs emanating from this state, i.e. the search is depth-first. This process will be repeated for all states until all possibilities are exhausted. The registers associated with the ATN are retained before a transition from a current state to any other state, by the using value-parameter-passing mechanism of Pascal. Keeping the value of the registers unchanged after returning from a called procedure is managed by passing each relevant register as an actual parameter to the called procedure. This enables the called procedure to start with the value of the register (actual parameter), but any change to this register will not affect the contents of the register in the calling procedure.

Having returned to S_0 after exhausting all possible arcs, all possible solutions are found on a list containing the values of the registers corresponding to each solution.

RESULTS AND CONCLUSION

The lexical analysis as described here was tested using different texts from different sources. It has demonstrated its capability to get all possible interpretations of an inflected Arabic word (see the Appendix). This does not guarantee that the method works on all conceivable occasions, but the samples of text used have been representative and substantial.

Although we are aware that Pascal is not the perfect choice as a programming language for such problems, the availability of its compiler on a machine easily accessible to us and that supports Arabic characters was the major factor influencing our decision to use it. The overall performance of the lexical analysis has been satisfactory: on the computer that we have used, the effective average rate of processing is 1·8 inflected Arabic words per second.

The rules created to represent context-sensitive knowledge concerning the relation between additions and a stem are a novel combination of the grammatical rules of the Arabic language and heuristics.

The work presented here should be helpful for any project that uses Arabic natural language, e.g. for automatic translation, user interfacing to a database or any other computer system or computing-aided learning of the Arabic language.

APPENDIX: AN EXAMPLE USED FOR TESTING THE LEXICAL ANALYSER

مواقف

وامريكا هى التى تضغط على الاشقاء العرب ان يساعدوا الطرفين فى الحرب ، وان يساعدوا طهران وبغداد اذا انتهت الحرب وبدا التعمير .. وكل الدول الغربية تفعل ذلك . ولم اضف الى معلوماتك جديدا .

ولكن يجب ان يظل هذا المعنى امامك دائما . انها مصالح فى الحرب ومصالح فى وقف اطلاق النار ومصالح فى السلام وتجفيف الدموع والدماء بعد ذلك .. واستنزاف دول البترول كلها ، وتمزيق العرب ..

والخطر الداخلى هو صدى هذه الزلازل على الناس ، رغيفا وطاقة ومدرسة ومستشفى وبيتا واولادا ومستقبلا وهجرة من المجتمع او تكفيرا له وانسحابا منه او عدوانا عليه ..

وهذا هو انسب وقت لكى ينشط اللصوص وقطاع الطرق السياسية والدينية ـ وليس كل ذلك سرا على احد . ولكننا ننسى . وليس من قبيل الصدفة ان نؤذن خمس مرات فى اليوم فى كل مسجد . فالسبب هو اننا ننسى او نحب ان ننسى حتى لا نؤدى ما هو واجب علينا .. حتى لا نصحو الى ما هو خطر على كل انسان !

أنيس منصور

عدنا الى الاشقاء العرب ، وهم الينا . لا يهم من هو صاحب الخطوة الاولى . تقاربت العواصم وامتدت الايدى وتلامسنا وتعانقنا . وعفا الله عما سلف . فسوف تكون عندنا من القضايا الجديدة ما يحتاج الى العودة والمعاودة والمراجعة .. ولا نهاية لما سوف يحدث ، ولما يجب ان نحرص عليه وان نخاف منه ..

وبمنتهى الصراحة : ان بلادنا جميعا فى خطر . وسوف نختلف على معنى الخطر . ولكنا فى خطر . فالكويت ليست اسوا حالا من مصر ، ولا العراق ولا سوريا ولا السودان ولا اليمن ... كلنا فى الهم والغم : عرب !

والخطر داخلى وخارجى . اما الخارجى فنحن نعرفه من مئات السنين . وهذا الخطر لا يقتحم الابواب دائما ، وانما هو يدخل من الباب وبالاذن .. يدخل بنا ومعنا قادما من واشنطون ومن موسكو ومن العواصم الغربية .. والذين ينادون بالسلام هم انفسهم الذين يتاجرون بالسلاح . مثلا : امريكا اعطت لايران السلاح واعطت للعراق خرائط بمواقع السلاح وطرق القضاء عليه ..

Textual input

عدنا الى الأشقاء العرب ، و هم الينا . لا يهم من هو صاحب الخطوة الأولى . تقاربت العواصم و امتدت الأيدى و تلامسنا و تعانقنا . و عفا الله عما سلف . غسوف تكون عندنا من القضايا الجديدة ما يحتاج الى العودة و المراجعة . و لا نهاية لما سوف يحدث ، و لما يجب أن نحرص عليه أن نخاف منه. و بمنتهى الصراحة : ان بلادنا جميعا فى خطر . و سوف نختلف على معنى الخطر . و لكننا فى خطر . فالكويت ليست أسوأ حالا من مصر ،و لا العراق و لا سوريا و لا السودان و لا اليمن .. كلنا فى الهم و الغم : عرب !
و الخطر داخلى و خارجى . أما الخارجى فنحن نعرفه من مئات السنين . و هذا الخطر لا يقتحم الأبواب دائما ، و لكن هو يدخل من الباب و بالاذن .. يدخل بنا قادما من واشنطن و من موسكو و من العواصم الغربية .. و الذين ينادون بالسلام هم أنفسهم الذين يتاجرون بالسلاح . مثلا : أمريكا أعطت لايران السلاح و أعطت للعراق خرائط بمواقع السلاح و طرق القضاء عليه .. أمريكا هى التى تضغط على الأشقاء العرب أن يساعدوا الطرفين فى الحرب ،و أن يساعدوا طهران و بغداد اذا انتهت الحرب و بدأ التعمير .. و كل الدول الغربية تفعل ذلك . و لم أضف الى معلوماتك جديدا . و لكن يجب أن يظل هذا المعنى أمامك دائما . انها مصالح فى الحرب و مصالح فى وقف اطلاق النار و مصالح فى السلام و تجفيف الدموع و الدماء بعد ذلك .. و استنزاف دول البترول كلها ، و تمزيق العرب ..
و الخطر الداخلى هو صدى هذة الزلازل على الناس ، رغيفا و طاقة و مدرسة و مستشفى و هجرة من المجتمع أو تكفيرا له و انسحابا منه أو عدوانا عليه .. و هذا هو أنسب وقت لكى ينشط اللصوص و قطاع الطرق السياسية و الدينية _ و ليس كل ذلك سرا على أحد . و لكننا ننسى . و ليس من قبيل الصدفة أن نؤذن خمس مرات فى اليوم فى كل مسجد . فالسبب هو أننا ننسى أو نحب أن ننسى حتى لا نؤدى ما هو واجب علينا .. حتى لا نصحو ألى ما هو خطر على كل انسان !.

(An Article published in *Al-Ahram* Newspaper, 18 March 1988, and written by Anis Mansour.)

Output from the lexical analyser

Forms in text	Possible stems	Stem type	Rejection
عدنا	عاد	فعل	N
الى	الى	آداة	N
الأشقاء	أشقاء	أسم	N
العرب	عرب	أسم	N
و	و	آداة	N
هم	هم	أسم	N
	هم	فعل	N
	هم	أسم	N
الينا	الى	آداة	N
لا	لا	آداة	N
يهم	هم	أسم	Y
	هم	فعل	N
	هم	أسم	Y
من	من	أسم	N
	من	آداة	N
هو	هو	أسم	N
صاحب	صاحب	أسم	N
الخطوة	خطوة	أسم	N
الأولى	أولى	أسم	N
تقاربت	تقارب	فعل	N
العواصم	عواصم	أسم	N
و	و	آداة	N
امتدت	امتد	فعل	N
الأيدى	أيدى	أسم	N

Forms in text	Possible stems	Stem type	Rejection
	و	آداة	N
و تلامسنا	تلامس	فعل	N
	و	آداة	N
و تعانقنا	تعانق	فعل	N
	و	آداة	N
و عفا	عفا	فعل	N
الله	الله	أسم	N
	ل	آداة	Y
عما	عما	آداة	N
سلف	سلف	فعل	N
فسوف	سوف	آداة	N
تكون	كان	فعل	N
عندنا	عند	أسم	N
من	من	أسم	N
	من	آداة	N
القضايا	قضايا	أسم	N
الجديدة	جديد	أسم	N
ما	ما	أسم	N
	أما	آداة	Y
يحتاج	احتاج	فعل	N
الى	الى	آداة	N
العودة	عودة	أسم	N
	و	آداة	N
و المراجعة	مراجعة	أسم	N
	و	آداة	N
و لا	لا	آداة	N
نهاية	نهاية	أسم	N
	ما	أسم	N
لما	أما	آداة	N
سوف	سوف	آداة	N
يحدث	حدث	فعل	N
	و	آداة	N
و لما	ما	أسم	N
	أما	آداة	N
يجب	وجب	فعل	N
أن	أن	آداة	N
نحرص	حرص	فعل	N
عليه	على	آداة	N
أن	أن	آداة	N
نخاف	خاف	فعل	N
منه	من	أسم	N
	من	آداة	N
	و	آداة	N
و بمنتهى	منتهى	أسم	N
المراحة	مراحة	أسم	N
ان	ان	آداة	N
بلادنا	بلاد	أسم	N
جميعا	جميع	أسم	N
في	في	آداة	N
خطر	خطر	أسم	N
	و	آداة	N
و سوف	سوف	آداة	N
نختلف	اختلف	فعل	N
على	على	آداة	N

Forms in text	Possible stems	Stem type	Rejection
معنى	معنى	أسم	N
الخطر	خطر	أسم	N
	ال	آداة	N
و	و		N
لكننا	لكن	آداة	N
	كان	فعل	N
في	في	آداة	N
خطر	خطر	أسم	N
فالكويت	كويت	أسم	N
ليست	ليس	فعل	N
أسوأ	أسوأ	أسم	N
حالا	حال	أسم	N
من	من	أسم	N
	من	آداة	N
مصر	مصر	أسم	N
و	و	آداة	N
لا	لا	آداة	N
العراق	عراق	أسم	N
و	و	آداة	N
لا	لا	آداة	N
سوريا	سوريا	أسم	N
و	و	آداة	N
لا	لا	آداة	N
السودان	سودان	أسم	N
و	و	آداة	N
لا	لا	آداة	N
اليمن	يمن	أسم	N
كلنا	ل	آداة	Y
	كل	أسم	N
في	في	آداة	N
الهم	هم	أسم	N
	هم	فعل	Y
	هم	أسم	N
و	و	آداة	N
الغم	غم	أسم	N
عرب	عرب	أسم	N
و	و	آداة	N
الخطر	خطر	أسم	N
داخلي	داخل	أسم	N
و	و	آداة	N
خارجي	خارج	أسم	N
أما	أما	آداة	N
	ما	أسم	N
الخارجي	خارج	أسم	N
نحن	نحن	أسم	N
نعرفه	عرف	فعل	N
من	من	أسم	N
	من	آداة	N
مئات	مئة	أسم	N
السنين	سنين	أسم	N
و	و	آداة	N
هذا	هذا	أسم	N
الخطر	خطر	أسم	N
لا	لا	آداة	N
يقتحم	اقتحم	فعل	N
الأبواب	أبواب	أسم	N
دائما	دائم	أسم	N

Forms in text	Possible stems	Stem type	Rejection
	و	آداة	N
و	لكن	آداة	N
لكن	هو	أسم	N
هو	دخل	فعل	N
يدخل	من	أسم	N
من	من	آداة	N
	باب	أسم	N
الباب		آداة	N
و	أذن	فعل	N
بالاذن	دخل	فعل	N
يدخل	ان	آداة	Y
بنا	أن	آداة	Y
	ب	آداة	N
قادما	قادم	أسم	N
من	من	أسم	N
	من	آداة	N
واشنطن	واشنطن	أسم	N
و	و	آداة	N
من	من	أسم	N
	من	آداة	N
موسكو	موسكو	أسم	N
و	و	آداة	N
من	من	أسم	N
	من	آداة	N
العواصم	عواصم	أسم	N
الغربية	غرب	أسم	N
و	و	آداة	N
الذين	الذين	أسم	N
ينادون	نادى	فعل	N
بالسلام	سلام	أسم	N
هم	هم	أسم	N
	هم	فعل	N
	هم	أسم	N
أنفسهم	أنفس	أسم	N
الذين	الذين	أسم	N
يتاجرون	تاجر	فعل	N
بالسلاح	سلاح	أسم	N
مثلا	مثل	أسم	N
أمريكا	أمريكا	أسم	N
أعطت	أعطى	فعل	N
لايران	ايران	أسم	N
السلاح	سلاح	أسم	N
	و	آداة	N
و	أعطى	فعل	N
للعراق	عراق	أسم	N
خرائط	خرائط	أسم	N
بمواقع	مواقع	أسم	N
السلاح	سلاح	أسم	N
	و	آداة	N
و	طرق	أسم	N
القضاء	قضاء	أسم	N
عليه	على	آداة	N
أمريكا	أمريكا	أسم	N
هى	هى	أسم	N
التى	التى	أسم	N
تضغط	ضغط	فعل	N

Forms in text	Possible stems	Stem type	Rejection
على	على	آداة	N
الأشقاء	أشقاء	أسم	N
العرب	عرب	أسم	N
أن	أن	آداة	N
يساعدوا	ساعد	فعل	N
الطرفين	طرف	أسم	N
في	في	آداة	N
الحرب	حرب	أسم	N
و	و	آداة	N
أن	أن	آداة	N
يساعدوا	ساعد	فعل	N
طهران	طهران	أسم	N
و	و	آداة	N
بغداد	بغداد	أسم	N
أذا	أذا	آداة	N
انتهت	انتهى	فعل	N
الحرب	حرب	أسم	N
و	و	آداة	N
بدأ	بدأ	فعل	N
التعمير	تعمير	أسم	N
و	و	آداة	N
كل	كل	أسم	N
	ل	آداة	Y
الدول	دول	أسم	N
الغربية	غرب	أسم	N
تفعل	فعل	فعل	N
ذلك	ذلك	أسم	N
و	و	آداة	N
لم	لم	آداة	N
أضف	أضاف	فعل	N
الى	الى	آداة	N
معلوماتك	معلوم	أسم	N
جديدا	جديد	أسم	N
و	و	آداة	N
لكن	لكن	آداة	N
يجب	وجب	فعل	N
أن	أن	آداة	N
يذل	ذل	فعل	N
هذا	هذا	أسم	N
المعنى	معنى	أسم	N
أمامك	أمام	أسم	N
دائما	دائم	أسم	N
انها	ان	آداة	N
مصالح	مصالح	أسم	N
في	في	آداة	N
الحرب	حرب	أسم	N
و	و	آداة	N
مصالح	مصالح	أسم	N
في	في	آداة	N
وقف	وقف	أسم	N
أطلاق	أطلاق	أسم	N
النار	نار	أسم	N
و	و	آداة	N
مصالح	مصالح	أسم	N
في	في	آداة	N
السلام	سلام	أسم	N

Forms in text	Possible stems	Stem type	Rejection
	و	آداة	N
و	تجفيف	أسم	N
تجفيف	دموع	أسم	N
الدموع		آداة	N
	و	أسم	N
و	دماء	أسم	N
الدماء	بعد	أسم	N
بعد	ذلك	أسم	N
ذلك		آداة	N
	و	أسم	N
و	استنزاف	أسم	N
استنزاف	دول	أسم	N
دول	بترول	آداة	Y
البترول	ل	أسم	N
كلها	كل	آداة	N
	و	أسم	N
و	تمزيق	أسم	N
تمزيق	عرب	آداة	N
العرب		أسم	N
	و	أسم	N
و	خطر	أسم	N
الخطر	داخل	أسم	N
الداخلى	هو	أسم	N
هو	صدى	أسم	N
صدى	هذة	أسم	N
هذة	زلازل	آداة	N
الزلازل	على	أسم	N
على	ناس	أسم	N
الناس	رغيف	آداة	N
رغيفا		أسم	N
	و	آداة	N
و	طاقة	أسم	N
طاقة		آداة	N
	و	أسم	N
و	مدرسة	آداة	N
مدرسة		أسم	N
	و	آداة	N
و	مستشفى	أسم	N
مستشفى		آداة	N
	و	أسم	N
و	دجرة	أسم	N
دجرة	من	آداة	N
من	من	أسم	N
	مجتمع	آداة	N
المجتمع	أو	آداة	Y
أو		أسم	N
	و	آداة	N
و	تكفير	أسم	N
تكفيرا		آداة	N
	و	أسم	N
و	انسحاب	أسم	N
انسحابا	من	آداة	N
من	من	آداة	N
	أو	آداة	Y
أو		أسم	N
	و	آداة	N
و	عدوان	أسم	N
عدوانا	على	آداة	N
عليه		أسم	N
	و	أسم	N
و	هذا	أسم	N
هذا	هو	أسم	N
هو	أنسب	أسم	N
أنسب	وقت	آداة	N
وقت	كى	فعل	N
لكى	نشط	فعل	N
ينشط	نشط		

Forms in text	Possible stems	Stem type	Rejection
الخصوص	خصوص	أسم	N
	و	آداة	N
و	قطاع	أسم	N
قطاع	طرق	أسم	N
الطرق	سياسة	أسم	N
السياسة		آداة	N
	و	أسم	N
و	دينية	آداة	N
الدينية		فعل	N
	و	أسم	N
و	لبس	آداة	Y
لبس	كل	أسم	N
كل	ل	أسم	N
	ذلك	أسم	N
ذلك	سر	آداة	N
سرا	على	أسم	N
على	أحد	آداة	N
أحد		آداة	N
	و	فعل	N
و	لكن	فعل	N
لكننا	كان	فعل	N
	نسى	آداة	N
ننسى		فعل	N
	و	أسم	N
و	لبس	أسم	N
لبس	من	آداة	N
من	من	أسم	N
	قبيل	أسم	N
قبيل	صدفة	أسم	N
الصدفة	أن	آداة	N
أن	أذن	فعل	N
نؤذن	خصص	أسم	N
خصص	مرة	أسم	N
مرات	شى	آداة	N
شى	يوم	أسم	N
اليوم	شى	آداة	N
شى	كل	أسم	N
كل	ل	آداة	Y
	مسجد	أسم	N
مسجد	سبب	أسم	N
فالسبب	هو	أسم	N
هو	أن	آداة	N
أننا	نسى	فعل	N
ننسى	أو	آداة	N
أو	و	آداة	Y
	أحب	فعل	N
نحب	أن	آداة	N
أن	نسى	فعل	N
ننسى	حتى	آداة	N
حتى	لا	آداة	N
لا	نؤدى	فعل	N
نؤدى	ما	أسم	N
ما	أما	آداة	Y
	هو	أسم	N
هو	واجب	أسم	N
واجب	على	آداة	N
علينا	حتى	آداة	N
حتى	لا	آداة	N
لا			

Forms in text	Possible stems	Stem type	Rejection
نصحو	نصحو	فعل	N
الى	الى	آداة	N
ما	ما	أسم	N
	أما	آداة	Y
هو	هو	أسم	N
خطر	خطر	أسم	N
على	على	آداة	N
كل	كل	أسم	N
	ل	آداة	Y
انسان	انسان	أسم	N

REFERENCES

1. Y Hlal, 'Information systems and Arabic: the use of Arabic in information systems', in *Linguistics and Signal & Information Processing*, Harper & Row Inc., 1987, pp. 191–197.
2. F. Anshen and P. Scriber, 'A focus transformation of modern standard arabic', *Language*, **44**, 792–797 (1968).
3. Y. Aoun, 'Structure Interne du groupe nominal en Arabe:l l'idâfa', *Analyses/Théories*, 1–40 (1978-1); 1–25 (1978-2).
4. R. Hartman, *Understanding zur Syntax der Arabischen Schriftsprache, eine Generative-transformationelle Darstellung*, Harrassowitz, Wiesbaden, 1976.
5. M. AL-khuli, *A Contrastive Transformational Grammar, Arabic and English*, E.J. Brill. N.V., Leiden, 1970.
6. A Fassi Fehri, 'al-lisāniyyāt wa- 'lugatu" l-arabiyya: namādiğ tarkiīiyy a wa-dataliyya', Dār al-Baydā, Dar Tūbqāl li-'l- našr, 1985.
7. A. Moutaouakil, *Pragmatic Functions in a Functional Grammar of Arabic*, Faris Publications, Dordrecht, 1989.
8. M. Al-Waer, 'Toward a modern and realistic sentential theory of basic structures in standard Arabic', *Proc. Conference on Applied Arabic Linguistics and Signal & Information Processing*, Rabat, Morocco, 1983, pp. 195–202.
9. A. Fahmy, M. Fakhry and M. Mohammed, 'Application of logic programming to Arabic language processing', *13th International Conference of Statistics, Computer Science, Social and Demographical Research*, Cairo, Egypt, 1988, pp. 113–128.
10. S. Mehdi, 'Arabic language parser', *International Journal of Man–Machine Studies*, **25**, 593–611 (1986).
11. D. Cohen, 'Essai d'une analyse automatique de l'Arabe', *Etudes des linguistique Sémitique et Arabe*, Mouton, The Hague, 1970, pp. 48–78.
12. R. Bathurst, 'Automatic alphabetization of Arabic words: a problem of graphic morphology and combinatorial logic, in R. Wisbey (ed.), *Papers from a Cambridge Symposium*, Cambridge University Press, 1971, pp. 185–190.
13. G. Bohas and Y. Hlal, 'Un système informatique pour la métrique Arabe, reconnaissance du métre: emploi des deux hémistiches', *Bulletin d'Etudes Orientales*, Institut Francais de Damas, Damascus, 1984, pp. 7–31.
14. B. Schubert, 'Ein automatisches Verfahren zur Morphologischen Analyse and zur Herstellung von Indices für arabische Texte', *ZDMG*, 1973, pp. 283–251.
15. A. Jones, 'Some Oxford projects in oriental languages', in R. Wisbey (ed.) *Papers from a Cambridge Symposium*, Cambridge University Press, 1971, pp. 191–197.
16. A. Amin, 'Un système pour la reconnaissance et pour la comprehension de l'Arabe écrit et imprimé', *Thèse de Doctorat d'Etat*, Université de Nancy, 1988.
17. H. Abaad, 'Un transducteur phonologique de l'Arabe', *Compte Rendu du Colloque sur la Communication entre Langues Européennes et Langues Orientales (Arabe–Chinois–Japonais)*, Commission des Communautés Européennes, Luxembourg, 1984, p. 46.
18. M. Ziadah, 'Présentation linguistique d'une modèle de synthèse morphologique de l'Arabe littéraire', *Compte Rendu du Colloque sur la Communication entre Langues Entre Lanuges Européennes et Langues Orientales (Arabe–Chinois–Japonais)*, Commission des Communautés Européennes, Luxembourg, 1984, pp. 45.

19. A. Chabir, 'Le verbe en Arabe: essai d'une analyse morphologique automatique', *Compte Rendu du Colloque sur la Communication entre Langues Européennes et Langues Orientales (Arabe–Chinois–Japonais)*, Commission des Communautés Européennes, Luxembourg, 1984, pp. 46–47.
20. J. Dicy, 'Vers un modèle d'analyse automatique du mot graphique non-vocalisé en Arabe', *Compte Rendu du Colloque sur la Communication entre Langues Européennes et Langues Orientales (Arabe–Chinois–Japonais)*, Commission des Communautés Européennes, Luxembourg, 1984, p. 46.
21. D. Vermel, 'Le traitement informatisé de la langue Arabe: l'analyse morphologique', *Compte Rendu du Colloque sur la Communication entre Langues Européennes et Langues Orientales (Arabe–Chinois–Japonais)*, Commission des Communautés Européennes, Luxembourg, 1984, pp. 116–118.
22. M. Kay, 'Nonconcatenative finite-state morphology', *3rd Conference of the European Chapter of the Association for Computational Linguistics*, 1987, pp. 2–10.
23. A. Jaccarini, 'Automatic generation of unvocalized Arabic recognition programs: theory and applications', *3rd Conference on Arabic Computational Linguistics*, Kuwait, 1989, pp. 66–87.
24. C. Koster, *Affix Grammars, Algol 68 Implementation*, North-Holland, Amsterdam, 1971, pp. 95–105.
25. H. Meijer, *Programmer: A Translator Generator*, Bloembergen Santee, Nijmegen, 1986.
26. E. Ditters, 'An extended affix grammar for the noun phrase in modern standard Arabic', *Corpus Linguistics* **11**, Rodopi, Amsterdam, 47–77, (1986).
27. M. Gheith, 'Realisation d'un programme comprenant des exercices de mécanique posés en Arabe', *Thése de Docteur-Ingénieur*, Université de Paris VI, 1980.
28. A. Rafea and Aisha Rafea, 'Understanding an Arabic word in a text automatically', *Nineteenth Annual Conference on Statistics, Computer Science and Operations Research*, Cairo, Egypt, 1984, pp. 52–77.
29. K. Koskenniemi, 'Two-level morphology: a general computational model for word-form recognition and production', *Doctoral Dissertation*, University of Helsinki, 1983.
30. A. Farghaly, 'Three-level morphology', presented at the *Arabic Morphology Workshop*, Stanford University, 1987.
31. E. Ditters, 'The role of semantics in automated syntactic analysis of Arabic linguistic data', *2nd Conference on Arabic Computational Linguistics*, Kuwait, 1989, pp. 125–147.
32. M. Gheith and M. Mashhour, 'A computer-based system for understanding Arabic language', *Workshop on Computer Processing of the Arabic Language*, Kuwait, 1985, pp. 21–42.
33. S. Selim, M. Gheith and M. Mashhour, 'A general morphological analyzer', *20th Annual Conference on Statistics, Computer Science and Operations Research*, Cairo, Egypt, 1985, pp. 21–42.
34. A. Moutaouakil, 'Lexical derivation in Arabic: roots and patterns', in *Linguistics and Signal & Information Processing*, Harper & Row Inc., 1987, pp. 93–97.
35. M. Harris, *Introduction to Natural-Language Processing*, Reston Publishing Company, 1985.
36. A. Kaye, 'A comparative-etymological dictionary of three African Arabic dialects on computer', *Second Conference on Arabic Computational Linguistics*, Kuwait, 1989, pp. 88–103.
37. A. Barr and E. Feigenbaum, *Handbook of Artificial Intelligence, Vol. 1*, William Kaufman Inc., Los Altos, California, 1981.
38. W. Woods, 'Transition network grammar for natural language analysis', *Comm. ACM*, **10**, 591–606 (1970).