PARSING ARABIC VERB PHRASES USING PREGROUP GRAMMARS

Ahmad M. Abd Al-Aziz

British University in Egypt (BUE), Cairo, Egypt

ABSTRACT

Parsing of Arabic phrases is a crucial requirement for many applications such as question answering and machine translation. The calculus of pregroup introduced by Lambek as an algebraic computational machinery for the grammatical analysis of natural languages. Pregroup grammar used to analyse sentence structure in many European languages such as English, and non-European languages such as Japanese. In Arabic language, Lambek employed the notions of pregroup to analyse some grammatical structures such as conjugating the verb, tense modifiers and equational sentences. This work attempts to develop an initial phase of an efficient automatic pregroup grammar parser by using linear approach to analysethe verbal phrases of Modern Standard Arabic (MSA). The proposed system starts building Arabic lexicon contains all possible categories of Arabic verbs, then analysing the input verbal Arabic phrase to check if it is well-formed by using linear parsing algorithm based on pregroup grammar rules.

KEYWORDS

Pregroup grammar, Lambek calculus, Modern Standard Arabic (MSA), Arabic language analysis, Computational Arabic linguistic&Arabic sentence parser

1. INTRODUCTION

Grammar has many approaches, the most common one is “generative-transformational grammar system” proposed by Noam Chomsky. This approach involves the use of defined operations named transformations to produce new sentences from existing ones. Another approach named “categorial grammar” pioneered by Ajdukiewicz [1]. This approach aims at introducing a computational component for linguists by assigning every word in lexicon one or more syntactic types—originally named “categories”, and then check whether a given string of words is a well-formed grammatical sentence by making a calculation on the corresponding string types. Pregroup grammar is a modified version of categorial grammar in which types take the form of strings of simple and compound types with left and right adjoints [2, 3]. The pregroup grammar approach has been studiedand applied on many languages such as English, French, Italian, German, Polish, Japanese and Persian [4]. In Arabic, the first attempt is madein [3] to analysethe Arabic phrases structure using the pregroup grammar notions, it was the first work example on non-European and non-Indo-European language. In their work, they introduced a small part of the algebraic machinery for many Arabic grammatical structures such as verbs, tense modifiers, equational sentences and personal pronouns attached to prepositions. Another attempt is made in [4] to analyse the movement of clitic in different languages such as Arabic, French, Persian and Italian. In theirwork, they analyse the cliticization and they extended the pregroup grammar notions with two cyclic meta-rules.

At the time of writing, there are a handful of works that have been done formally aiming at analysing the Arabic sentences structure using pregroup grammar with limited existence of serious attemptsto develop anautomatic Arabic parser using this approach of grammar. This workproposes an initial phase of an automatic systemused as Arabic parser based on pregroup grammar notions introduced by Lambek and Bargelli in [3]. This initial phasecovers some

DOI: 10.5121/ijnlc.2018.7303

grammatical rules for both verbal and nominal phrases. The proposed system constitutes of two main stages. The first stage is to build an Arabic mini-lexicon using XML files; it contains the grammatical rules for verbal phrases, and the types of some action transitive and intransitive verbs. The second stage is to apply an algorithm based on linear approach to make a calculation on string type sored in lexicon, to check the correctness of the input phrase's grammatical structure.The remainder of this paper structured as follows. Section 2 describes the related efforts of different pregroup grammar parsing approaches. Section 3 provides an overview of pregroup grammar notions. Section 4 lists and describes the basic types. Analyzing the Arabic verb structure phrased is provided in section 5. Section 6 describes the entries of Arabic lexicon, section 7 describes the proposed algorithm, the testing and evaluation is described in section 8 and finally the conclusion provided in section 9.

2. RELATED WORK

The formal main aim of pregroup parsing is checking whether a given string $x = a_1 \ldots, a_n$, where $a_i \in \sum$ (a set of words in any given language), and $i = 1,\ldots,n$is a member of the language generated by a pregroup grammar G, that is whether$x \in L(G)$[4].Many works take pregroupgrammar parsing as matter of interest in recent years. All of them applied on foreign languages such as English and French. Oehrle proposed aparsing algorithm based on lexical graph representation.In this this algorithm, a separate directed linear graph is built for each word of the string, then they are joined together and finally new arcs corresponding to possible contractions are added [5]. However, the main concern of this algorithm is the complexity of execution. Béchet introduced other idea of pregroup parsing algorithm. The main idea of this algorithm is to transform pregroup grammars into context-free grammar (CFG). Béchet presents a recognition algorithm based on a restricted form of partial composition called "functional composition", then applied a Cocke-Kasami-Younger (CYK) to deduce the CFG from pregroup grammar that generates the same language[6]. However, the algorithm constructs a CFG whose size is exponential to the given pregroup grammar.Savateev introduced a recognition algorithm running in polynomial time to decide whether a given word in a string belongs to a language generated by a given unidirectional Lambek grammar [7]. Morozo considers the polynomial time algorithm introduced by Savateev and presents a cubic parsing algorithm for ambiguous pregroup grammar, he presents a java application that uses algorithm for natural language processing [4]. Another approach for parsing pregroup grammars proposed by Preller. heintroduces a linear algorithm for unambiguous pregroup grammars to run in linear timeby adding some restrictions imposed on the type lexicon.Preller divides the tasks of this algorithm into two main tasks. The first one is to choose the lexical entry for each given word. The second one is to decide whether a choice of type has a reduction, and if there is one, it provides at least one such reduction [8]. This work applied the algorithm introduced by Preller in [8].The motivation behind this selection is it has less execution complexity than Oehrle and Béchet algorithms. This work applies the same concept of Preller's algorithm and modifies the restrictions on type lexicon as shown in section 6.

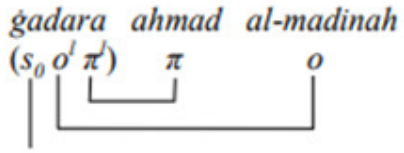
3. AN OVERVIEW OF PREGROUP GRAMMAR

The idea of pregroups, like other categorial grammars, is that sentences are built from words by use of the lexical rules only, with assumption that words of string have certain syntactic properties that can be described by a finite set of pregroup types [8]. A string of words is considered a *sentence* if, and only if, for each word there is such a type in the lexicon that the concatenation of those types can be transformed into a sentence by performing some pregroup calculations and rules and on it. Formally, the idea of pregroup grammar is to assign for each word in the lexicon one or more types (simple or compound) that are basic defined as a string of types in a form of a^l, a^{ll}, a, a^r, a^{rr}. A pregroup is a partially ordered non-commutative monoid in

which each element a has a left adjoint a^l and a right adjointa^r.For given example in English, " John met Mary" is a sentence constitues three words, pregroup grammar assigned a type for each word as follows. "John" is a simple type N means noun, "met" is a tansitive verb has a compound type of (N^r S N^l), means this element is a tranisitive verb that needs a noun on its right and other noun on its left. Finally, "Mary" is a simple type N means noun. To check if this sentence has a well-formed grammar and to analyze sentence structure, some rules should be applied. In pregroup grammar, there are two main rules, *contractions* such that $aa^l \rightarrow 1$, $aa^r \rightarrow 1$, and *expansions* such that $1 \rightarrow aa^l$, $1 \rightarrow aa^r$ where ($\rightarrow$) refers to extending partial order from one *basic* types to another, and 1 is an empty string. Originally, Lambek used ($\leq$) to denote the order in pregroup, but he later adopted the arrow ($\rightarrow$) for partial order [10]. By applying the contraction rule on the previous example, then we get (N N^r) S (N N^l), that leads to S which is the simple type denotes staement.

The pregroup is partially order set A of basic types i.e. a set with a binary relation ($\rightarrow$) which is *reflexive* such that$a \rightarrow a$, *transitive* such that if $a \rightarrow b$ and $b \rightarrow c$then $a \rightarrow c$and *antisymmetric* such that if $a \rightarrow b$and $b \rightarrow a$ then$a = b$. The monoid operation of the pregroup is *non-commutative*, where $ab = ba$does not hold [3]. A sentence s is grammatical sentence if, and only if it reduced to the type s (or any other basic types such as q in the case of question). This procedure depicted by the under-link diagram, by assigning a vertical line to each basic type and connecting reduced adjoint types via horizontal line. In grammatical sentence, afterreduction process, there should exactly only one vertical line for desired type such as s for sentence or $q$for question [2].

Taking basic types applied on Arabic phrase such asπdenotes external subject, $o$denotes external object and $s_0$denotes declarative sentence in the past tense, we obtain simpletypes such asπ^l, π^r, o^l, o^r,..., and compound types such as (s_0 o^l π^l) for the type of simple transitive verb in the past tense followed by external subject then external object, where the sentence's first word is a verb in form of (VSO), since Arabic language is written from right to left. For example, the types of constituents of the sentence *(ġadara ahmad al-madinah)* or *(ahmad left the city)* are as follows.



There are properties of pregroup grammar such as the adjoints are unique and that unit is self adjoint i.e. $1^l = 1 = 1^r$. The adjoint of multiplication is the multiplication of adjoints but in the reverse order i.e. $(a.b)^l = b^l.a^l$and $(a.b)^r = b^r.a^r$. In addition, The composition of the opposite adjoints is identity i.e. $(a^l)^r = (a^r)^l = a$. the adjoint operation is not involutive i.e. $a^{ll} = (a^l)^l \neq a$ and $b^{rr} = (a^r)^r \neq b$ [2].

4. BASIC TYPES

This work uses the pregroup grammar notions introduced in [3] to analyze the Arabic verb phrase as follows.

π_k = the *k-th* person pronoun, where *k= 1, 2m, 2f, 3m, 3f, 4, 5m, 5f, 6m, 6f* stands for the three persons singular followed by the three persons plural; *m* stands for masculine and *f* stands for feminine.
s = statement, where the tense is irrelevant.

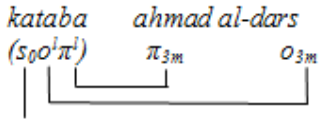
s_i = statement in the i-*th* tensemood, where *i*=0 for past, and *i*=1 for presentindicative, 2 for the jussive and 4 for the future tense.
n = singular noun.
o_k= direct object, where *k= 1, 2m, 2f, 3m, 3f, 4, 5m, 5f, 6m, 6f*indicates three persons singular followed by the three persons plural; *m* stands for masculine and *f* stands for feminine.

Thoughout this analysis, we postiulate the following rules:
$\pi_k \rightarrow \pi$, $s_i \rightarrow s$, $\pi_k \rightarrow n.o_k \rightarrow o$, $o_k \rightarrow n$.

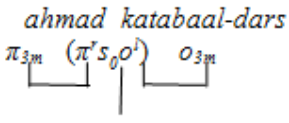
5. TYPING THE ARABIC VERB

Arabic language has two types of sentences; verbal sentence and nominal sentence. Nominal sentences are also called copular or equational sentences. The prototypical of verbal sentence starts with just a verb conjugated with pronominal subject. The classical form of Arabic verbal phrase is (Verb-Subject-Object) or (VSO). Whereas, nominal sentence has two different forms; equational or (verbless) sentence, and sentence starts with subject followed by verb.

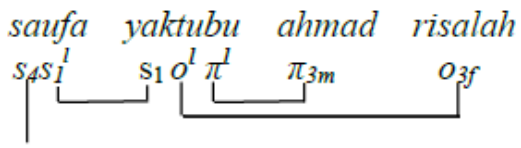
The first simple form for Arabic verbal sentence starts with such a verb in one of tense mood followed by non-pronominal subject and object in its left, the verb agrees with the subject in person and gender.Taking example of sentence introduced by Lambek in [3] as *(kataba ahmad al-dars)* or *(ahmad wrote the lesson).*



Here, the sentence is verbal sentence starts with verb *(kataba or wrote)* in past tense-mood as *i*=0 followed by subject π^l*(ahmad)* on its left and o^l(*al-dars or the lesson*) on its left. After applying the *contracton* rule that postulated in secont 4, thephraseis deduced into s_0, implies that this phrase is verbal sentence in the past tense. In Arabic, the previous phrase may be written as nominal pharasewhere the subject comes before the verb in form of (SVO). This structure considered be special case of nominal sentence where the predicate of nominal senetnec is a vebal sentence.



In Arabic, there are some morphemes change the tense-mood of a verb form [3]. Thus *saufa* and the prefix *sa+* (will in English)have type $s_4 s_1^l$change the present tense into future. Taking example of sentnece as (*saufayaktubuahmadrisalah*) or *(ahmad will write a letter*).



Here, the sentence is verbal sentence starts with tense modifier *saufa* (or will) that changes the present tense *yatubu* (or writes) into future tense-mood. There are many other cases of typing Arabic verb phrases introduced in [3], but this work focuses on parsing Arabic present and past tense-mood for simple senetneces structure contain non-pronominal subjects and direct objects.

6. THE ARABIC LEXICON

Lexicography is the branch of computational linguistics concerned with the design and construction of lexica for any practical use [10].All languages have a set of vocabulary items "lexicon" and a "grammar" describing how such words or items may be combined together to form larger items (sentence). To let computer program to be able to correctly process any given language, it must applya predfined precise lexical and syntactic rules. In pregroup grammar, a lexicon is a list of ordered pairs <*v* :*X*>,which is called *lexicalentries* , where *v* is a word of the language and *X*an element of the pregroup, called *type*[12]. The same item may be listed in lexicon several times, but with different types. To analyse a string of words $v_1 \ldots v_n$, one chooses types X_i such that $v_i : X_i$ belongs to the lexicon for $1 \leq i \leq n$ and checks ifsuccessive applications of the generalized contraction rule reduce the concatenated type $X_1 \ldots X_n$ to the sentence type *s*. Each such choice of contractions is a *reduction* from$X_1 \ldots X_n$to *s*. A string of words $v_1 \ldots v_n$ is recognized as a sentence if the type $X_1 \ldots X_n$reduces to *s* for some type assignment for $v_1 \ldots v_n$[12]. This work developed the Arabic lexicon in XML file.The file contains two sections. The first section contains the grammatical rules (or grammar) used for generalized contraction as postulated in section 4. The second sectioncontains the pregroup types for two verbs; the present tense verb (*yaktubu* or *writes*) and the past tense verb (*kataba* or *wrote*). In addition to the types of two nouns; (*ahmad*) as it can be a subject or direct object in any given sentence, and (*risalah* or letter) as it also can be a subject or direct object in any given sentence as shown in Figure 1.

```
<lexicon>
        <relations>
        <rel smaller="s_1" greater="s" />
        <rel smaller="s_2" greater="s" />
        <rel smaller="pi_3m" greater="pi" />
        …
        </relations>
        <entries>
        <entry form="يكتب">
                <component>v</component>
                <type>s_2^{0} o^{-1} pi^{-1}</type>
                <type>pi^{1} s_2^{0} o^{-1}</type>
        </entry>
        <entry form="كتب">
                <component>v</component>
                    <type>s_1^{0} o^{-1} pi^{-1}</type>
                <type>pi^{1} s_1^{0} o^{-1}</type>
        </entry>
        <entry form="احمد">
                <component>n</component>
                <type>pi_3m^{0}</type>
                <type>o_3m^{0}</type>
        </entry>
        <entry form="رسالة">
                <component>n</component>
                <type>pi_3f{0}</type>
                <type>o_3f^{0}</type>
        </entry>
        </entries>
</lexicon>
```

Figure 1. Arabic lexicon for pregroup grammar

We modify the lexicon proposed by Preller in [8] by adding the grammatical pattern called <component>…</component >for each word in lexicon. The <component>tag specifies the grammatical component for each word such as *v*denotes the verb, *n* denotes noun. This will eliminate the choice of lexical entry in lexicon and avoid processing type assignment which have no reduction to the sentence type, as the same word may have more than one type. Taking simple example as (*kataba ahmad risalah*) or (*ahmad wrote a letter*). By extracting the component entries from lexicon, the grammatical pattern will be [*v, n, n*], means the phrase starts with verb followed by two nouns. In this case, our algorithm (as discussed in section 7) takes the type$s_0\, o^l\, \pi^l$ for processing, and considers the $\pi^r s_0\, o^l$ as a loser type, and it is not taking in processing phase.

7. PARSING ALGORITHM

The problem of parsing pregroup grammars is complex for different reasons. The most obvious one is that pregroup grammars allow ambiguities of different types. The first kind of ambiguity can be formed as the same word can take different types, this is surely can be done if the same word takes different role in the sentence.The second type of lexical ambiguity as introduced in [12] is when there are more reductions to the sentence type for the same string of words. In Arabic language, this type of lexical ambiguity is common as we discussed in section 5, when SVO and VSO sentence structure has different reduction.Preller and Princeintroduced in [12] a full version of linear algorithm for unambiguous pregroup grammars for French verb phrases. In their algorithm, the sufficient conditions are given to be linear and complete. This paper appliesand develop the same algorithmon Arabic verb phrase with some modification on XML file lexicon. The modification as discussed in section 6 is by adding a <component>…</component> tag that specifies the grammatical component for each word. To

avoid processing type assignment which have no reduction to the sentence type. The linear algorithm of parsing Arabic verb phrases is shown in Figure 2.

Taking example of input string of words (*kataba ahmad risalah*) or (*ahmad wrote a letter*). The system takes the input of verbal sentence and loads lexicon, grammatical rules and grammatical patterns. The system initializes two empty stacks; R-Stack= [] to store the irreducible types and L-Stack= [] to store the successful links after reduction. The system assigned the grammatical pattern for each word in the string as follows.

$$[kataba, ahmad, risalah] \rightarrow [V, N, N]$$

The system takes the grammatical patterns and looks for each word in Arabic lexicon to assign a type for each works as follows.

$$\begin{aligned} kataba\ \text{كتب} &: (s_0\ o^l\ \pi^l) \\ &: (\pi^r\ s_0\ o^l) \\ Ahmad\ \text{أحمد} &: \pi_{3m} \\ &: o_{3m} \\ risalah\ \text{رسالة} &: \pi_{3f} \\ &: o_{3f} \end{aligned}$$

Pregroup grammar linear parsing algorithm

Input: *String of words x, Arabic pregroup lexicon, grammatical rules, Arabic grammatical patterns.*

```
1- Initializing R-Stack, L-Stack // R-Stack for irreducible types, L-Stack for storing links
2- Checking the grammatical pattern // to solve the problem of searching in all types of lexicon
3- Assigning type to word x_i from lexicon.
4- Concatenate assigned simple types with the order.
5- for x_{i-1} to x_{i-length(string)}:
       Push the type of x_i into R-Stack
       Compare the type of x_i with the type of x_{i-1} // the added element compared with previous one
      Check grammatical rules
       if two contracted:
         Push link [i, j] into L-Stack // where i is the word's order for x_i and j is the word's order of x_{i-1}
         Pop contracted types out of R-Stack
     else:
        Go to step 5.
6- Bool Check_ R_Stack_Func (R_Stack):
   if R-Stack !=Null and R-Stack contains only one type= S //where S denotes Statement
      Return True
  Else: Return False
7-Bool Check_R_Stack_Func (R_Stack):
  for each element [i, j]and [m, n] in R_Stack:
      If ( i!= j) and j > i and (m,n >i and m,n <j)
        Return True
    Else:
       Return False
8- if Check_R_Stack_Func (R_Stack) and Check_R_Stack_Func (R_Stack) ==True
      Print ("Statement is correct")
  Else:
    Print ("Statement is incorrect")
```

Figure 2. Linear pregroup algorithm

The system checks the Arabic grammatical pattern to select the right type for each word, as the phrase starts with verb "*kataba*" or "*wrote*" in a past tense, then the system assigns the first type ($s_0\ o^l\ \pi^l$) to it, and it considers the second type as loser type. Similarly, for the second word

"*ahmad*", the system assigns the first type (π_{3m}) to it, as according to the Arabic grammatical pattern, the noun that comes after the verb is a subject. Then the system considers the second type in Arabic lexicon as loser typ. Finally, for the word "*risalah*" which is a noun comes after the verb and subject. The system considers it as object, and assigns the type (o_{3f}) to it, and considers first type as loser type. The final assigned types for the words of strings as follows.

$$[kataba, ahmad, risalah] : [(s_0\ o^l\ \pi^l), \pi_{3m}, o_{3f}]$$

For each type, the system concatenates each simple type with its order, this will help to determine the link between the types as follows.

$$[kataba, ahmad, risalah] : [(s_0 1\ o^l 2\ \pi^l 3), \pi_{3m} 4, o_{3f} 5]$$

The system pushes the first type into R-Stack from left to right as shown in Figure 3. Then pushes the second type. Once the second type is pushed, the system makes a comparison between the first and the second type by checking the rules stored in lexicon XML file if contracts, the system pushes the link in L_Stack (the order of types in statement) and pops the reducible types from R_Stack.

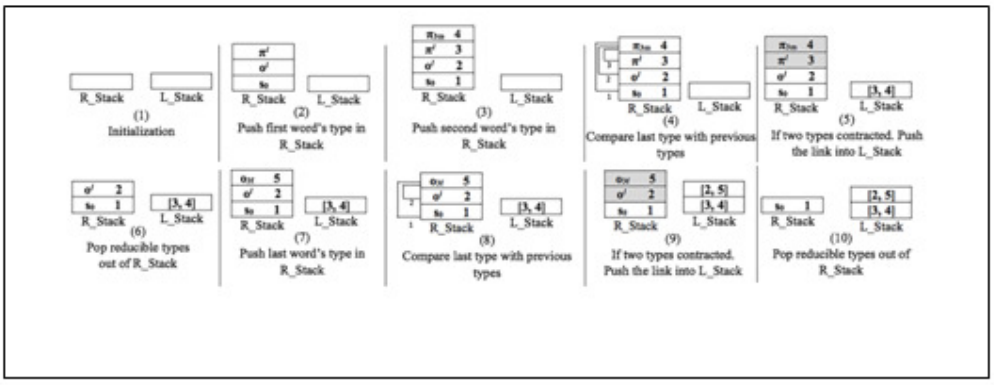


Figure 3. R_Stack and L_Stack push and pop operations

Once the R_Stack contains only one acceptable simple type (*s*in our example). The system checks the values in L-Stack which stores the successful links after reduction based on reduction rules introduced in [12]. The system accepts the phrase if, and only if the R_Stack has only one simple type (*s* in our example) and the reduction links are correct.

8. SYSTEM EVALUATION

The system has been tested and evaluated by input some Arabic verbal phrases in form of SVO and VSO structure in the past tense that constitutes of the same word items such as "*kataba ahmad risalah*" or "*ahmad wrote a letter*" and "*ahmad kataba risalah*". In addition, the system has been tested also for some verbal phrases in the present tense such as "*yaktubu ahmad risalah*" or "*ahmad writes a letter*" and "*ahmad yaktubu risalah*". The system succeeds in checking all the formal sentences and considers them as a well-formed sentence. The system has been tested also for some wrong grammatical sentences such as "*rihalah yaktubu ahmad*" or "*a letter writes*

ahmad", the system rejects this sentence as it is not a member of the language generated by a pregroup grammar defined in lexicon.The main limitation of this system is the lack of the Arabic lexicon to include more verb tenses such as future tense. In addition, the lack of visual tool to presents the link reduction graph after parsing process.

9. CONCLUSIONS

The main contribution of this work is to developthe first phase of pregroup grammar parser for Arabic language based on the formal analysis of Arabic sentences introduced by Lambek. This work builds the Arabic mini-lexicon contains verbs in present and past tense, and applies an algorithm based on linear approach to check the correctness of Arabic verb phrases. In the future, we are aiming at completing the Arabic parser in order to analyze more Arabic grammatical phrases such as doubly transitive verbs, double adjoints and the full analysis of clitic movement in Arabic language. In addition, we will add the visual tool presents the reduction link graph after parsing process.

REFERENCES

[1] Joachim, Lambek,(2006) "Pregroup and natural language processing", Springer Science, Vol. 28, No. 2, pp41-48.

[2] Sadrzadeh, Mehrnoosh, (2007) "Pregroup analysis of Persian sentences", In Claudia Casadio and Joachim Lambek, editors, Computational algebraic approaches to natural language, Polimetrica, Milano, Italy, pp121–144.

[3] Daniele, Bargelli&Joachim, Lambek,(2001) "An algebraic approach to Arabic sentence structure", Linguistic Analysis, Vol. 31, pp301–315.

[4] Moroz, Katarzyna, (2010) "Savateev-style parsing algorithmfor pregroup grammars", InProceedingsof Formal Grammar14,Lecture Notes in Computer Science, Vol. 5591.

[5] Richard, Oehrle, (2004) "A parsing algorithm for pregroup grammars", In Proceedings of Categorial Grammar, pp59-75.

[6] Denis, Béchet, (2007) "Parsing pregroup grammars and lambek calculus using partial composition", Studia Logica, Vol. 87, pp199-224.

[7] Yury, Savateev, (2010) "Unidirectional Lambek grammars in polynomial time", Theory of computing systems, Vol. 46, pp662-672.

[8] Anne, Preller, (2007) "Linear processing with pregroups", Studia Logica. Vol. 87, pp171-197.

[9] Joachim, Lambek,(2001) "Type grammars as pregroups", Grammars, Vol. 4, pp21-39.

[10] Joachim, Lambek, (2005) "What are pregroups?", In Claudia, Casadio& Robert Seely & Philip Scott, editors, Language and grammar: Studies in mathematical linguistics and natural language, CSLI publications, Stanford, pp129-136.

[11] Aref, Abu Awad &Essam Hanandeh, (2016) "Developing a transition parser for the Arabic language", International Journal of Advanced Computer Science and Applications, Vol. 7, pp173-175.

[12] Anne Preller & Violaine Prince, (2008) "Pregroup grammars with linear parsing of the French verb phrase", In Claudia Casadio and Joachim Lambek, editors, Computational algebraic approaches to natural language, Polimetrica, Monza, Italy, pp53–84.

AUTHOR

Ahmad Abd Al Aziz, PhD,is a lecturer at British University in Egypt (BUE). He was double majors' post-doctoral researcher, first in Social Sciences and second in computer sciences. He gained the PhD Degree of social sciences in February 2014 from Ain Shams University (Egypt) and second PhD. Degree of computer science in field of computational linguistics in December 2016 from Cairo University (Egypt). Ahmed published a number of scientific papers in natural language processing.