A Chart Parser for Analyzing Modern Standard Arabic Sentence

Eman Othman
Computer Science Dept., Institute of Statistical Studies and Research (ISSR), Cairo Univ.
5 Tharwat St., Orman, Giza, Egypt
E-mail:emy_othman@hotmail.com

Khaled Shaalan
Computer Science Dept., Faculty of Computers and Information, Cairo Univ.
5 Tharwat St., Orman, Giza, Egypt
Email:shaalan@mail.claes.sci.eg

Ahmed Rafea
Computer Science Dept., American University in Cairo
113, Sharia Kasr El-Aini, P.O. Box 2511, 11511, Cairo, Egypt.
*E-mail:*rafea@aucegypt.edu

Abstract

The parsing of Arabic sentence is a necessary prerequisite for many natural language processing applications such as machine translation and information retrieval. In this paper we report our attempt to develop an efficient chart parser for Analyzing Modern Standard Arabic (MSA) sentence. From a practical point of view, the parser is able to satisfy syntactic constraints reducing parsing ambiguity. Lexical semantic features are also used to disambiguate the sentence structure. We explain also an Arabic morphological analyzer based on ATN technique. Both the Arabic parser and the Arabic morphological analyzer are implemented in Prolog. The linguistic rules were acquired from a set of sentences from MSA sentence in the Agriculture domain.

1. Introduction

Natural language analysis serves as the basic block upon which natural language applications such as machine translation, natural language interfaces, and speech processing can be built. A natural language parsing system must incorporate three components of natural language, namely, lexicon, morphology, and syntax. As Arabic is highly derivational, each component requires extensive study and exploitation of the associated linguistic characteristics (Khayat, 1996).

Last decade, research work and development in natural language processing (NLP) systems for Indo-European languages has explored specific approaches to parsing in some depth, has consolidated practical experience, and has emphasized some trends, for example towards, lexical function grammars and deterministic parsing, and towards closer integration of syntax and semantics. This has led automatic NLP to gain in strength and self-confidence, as it has been clearly shown that working systems can be built, even if they are only very modest ones(Feddag, 1992). On the contrary, little work has been done in developing parsers involving Arabic (Ouersighni, 2001).

An important consideration, in the development of an Arabic parsing system, is the matter of Arabic morphology. Most of the researches in Arabic NLP systems mainly concentrated on the fields of morphological analysis (Ditters, 2001 and Jaccarini, 2001). In a previous work (Rafea et al., 1993), we analyzed and discussed the problem of implementing a morphological analyzer for inflected Arabic words. With respect to the implementation of the Arabic parser, we took the advantage of the already developed morphological analyzer by integrating it with the Arabic parser.

Parsing Arabic sentences is a difficult task. The difficulty comes from several sources: 1) the length of the sentence and the complex Arabic syntax, 2) The omission of diacritics (vowels) in written Arabic "altashkiil", 3) The free word order nature of Arabic sentence, and 4) The presence of an elliptic personal pronoun "alDamiir almustatir". The main goal was to implement a computer system to parse Arabic sentences (Shaalan et al., 1999). The architecture of the system is given in Fig. 1.

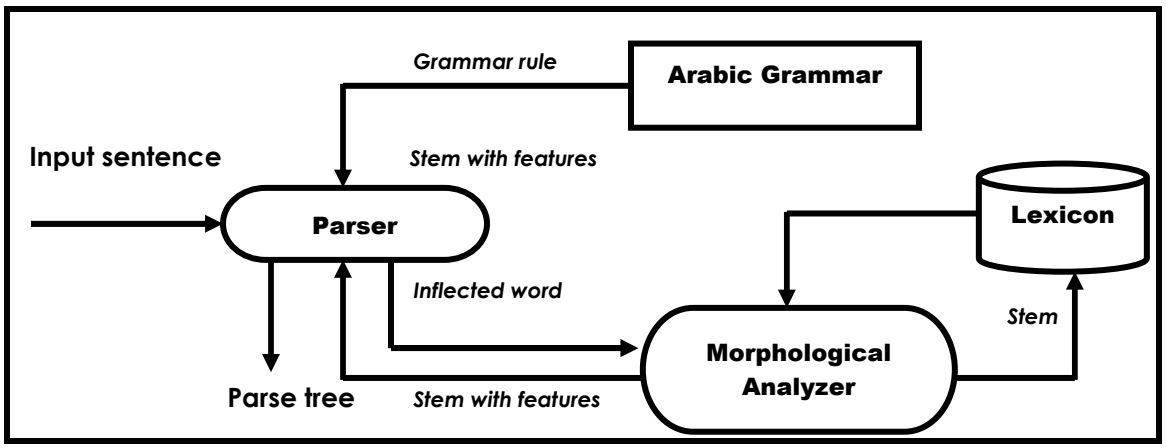


Figure (1) – The architecture of The Arabic Chart Parser

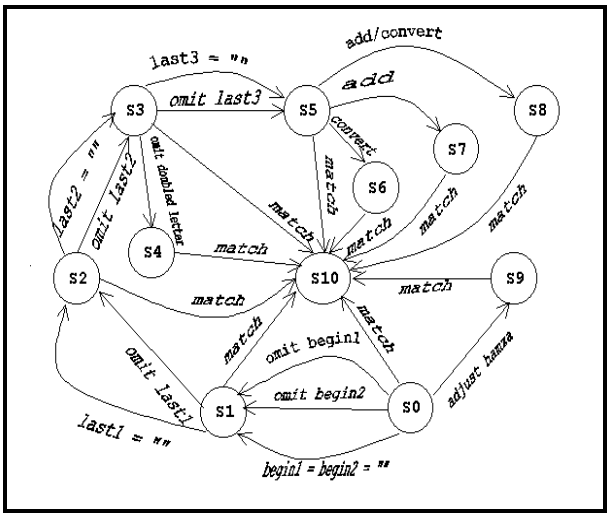


Figure (2) – ATN representing the relation between the additions and stem of an inflected Arabic word.

This article describes our attempt in developing an Arabic Parser for MSA text. The next section briefly presents the Arabic morphological analyzer and the lexicon structure. Then the focus turns to a description of implementation of the grammar rules and the ambiguity resolution, which is the concern of this study. Next, we show a worked example. In a concluding section, we present some final remarks

2. The Arabic Morphological Analyzer and Lexicon

In Rafea et al. (1993) we described a morphological analyzer for inflected Arabic words. An augmented transition network (ATN) (Woods, 1970) technique was successfully used to represent the context-sensitive knowledge about the relation between a stem and inflectional additions. The ATN consists of arcs. Each of which is a link from a departure node to a destination node, called states, see Fig. 2. An exhaustive-search to traverse the ATN generates all the possible interpretations of an inflected Arabic word. The morphological analyzer is implemented in Prolog.

The morphological analyzer consists of three modules: Analyzer module, lexical disambiguation module and features extraction module. Fig. 3 shows an example of analyzing the inflected Arabic word "الطلبين" (alTalabayn). In this example, this word is analyzed into a verb and a noun. The former is discarded because the prefix is only used with nouns.

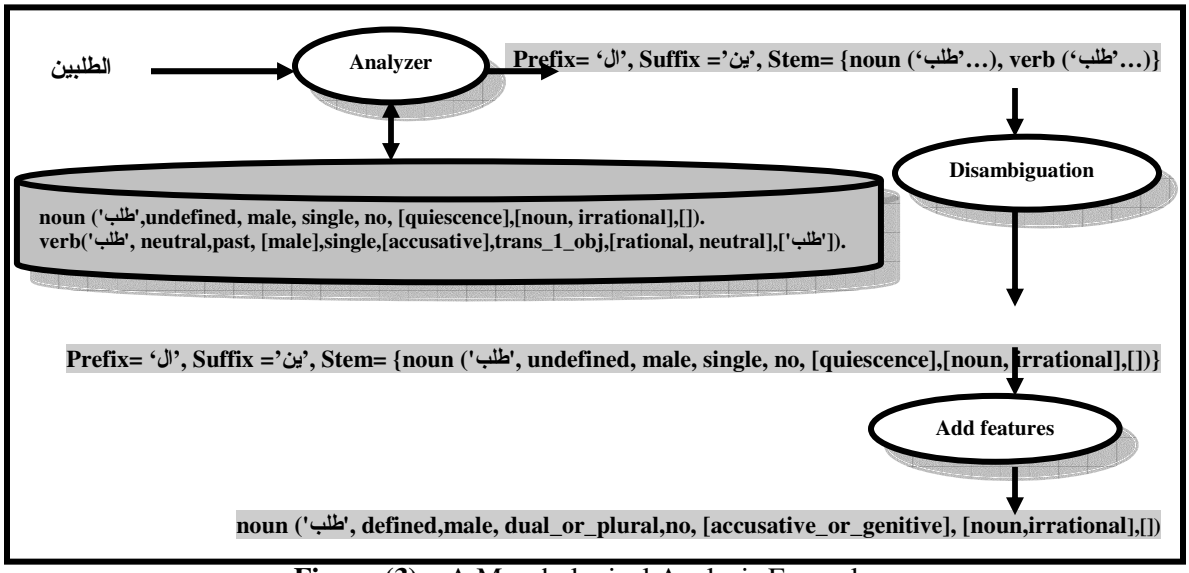


Figure (3) – A Morphological Analysis Example

The lexicon is designed to reflect the word categories in Arabic. There are three morphological categories for Arabic—noun, verb, and particle— each with a different set of features. There are two types of features in the lexicon: syntactic features that eliminate syntactic ambiguity and lexical features that eliminate lexical ambiguity. The values of these features are stored in the lexicon and can be modified during the sentence analysis.

The lexicon entry is represented as a Prolog fact. The following describes the forms of the lexicon entry:

- **Verbs:** A verb has the following form:
  `verb (Stem, Voice, Tense, [Subject Gender, Object Gender], Number, End case, Transitivity, [Subject rationality, Object Rationality], Infinitive)`
  - *Syntactic features:*
    - **Voice:** passive / active
    - **Tense:** past / present
    - **[Subject gender, Object Gender]:** [Male/Female, Male/Female]
    - **Number:** singular /dual / plural
    - **[End Case, Agent]**: [accusative / nominative / genitive, subject / object / proagent],
    - **Transitivity:** intransitive / transitive_1_obj / transitive_2_obj
      This feature gives the grammar the ability to predict the max number of agents expected after the verb that helps in distinguishing between passive and active voice of verbs that do not change their form in both cases.
  - *Lexical features:*
    - **[Subject Rationality, Object rationality]:** [rational/irrational, rational/irrational]
      This feature helps in determining that an agent is the proagent for a verb but not its subject by comparing the rationality feature of this verb with the agent feature.
    - **Infinitive:** [infinitive form] since we did not store the passive form of the verb in the lexicon this feature is needed when the morphology decides that the verb in the passive voice.

- **Nouns:** A noun has the following form:
  `noun(Stem ,Definition ,Gender ,Number ,Adjectivability ,End case ,[Category, Rationality] , irregular plural).`
  - *Syntactic features:*
    - **Definition:** defined/undefined/neutral
    - **Gender:** male/female
    - **Number:** Singular /dual / plural,
    - **End case:** [indeclinable/quiescence/accusative/nominative/genitive, without_noon: to indicate that the noun does not take suffix "ن" in case of dual or plural which means that the noun must be in an annexation form]
    - **Irregular plural:** [broken plural form of the irregular noun]
  - *Lexical features:*
    - **Adjectivability:** yes: if we can get the adjective form by adding the suffix "ى" /no otherwise
    - **[Category, Rationality]:** [category can be any noun type like: adjective, infinitive, demonstrative noun … etc., rational/irrational]. The category is needed because some noun types are not allowed to occur in a certain sentence position like the adjective in the position of subject.
- **Particles:** A particle has the following form:
  `Particle (Stem, Category).`
  The only feature represented here is the **Category:** preposition, conjunction…etc.

3 A Unification Based Grammar (UBG) Grammar for Arabic

Unification based grammar (UBG) formalism (Covington, 1994) is used to write the Arabic grammar rules in the proposed chart parser. The grammar is implemented in SICStus Prolog 3.10. Each grammar rule has the form

```
rule(LHS,RHS):- constrains
```

Each constraint is used for one of three purposes:

1. To make the agreement between the left and right hand side of the grammar rule.
2. To reduce the syntactic ambiguity.
3. To reduce the semantic ambiguity.

The grammar specifies the structure of the Arabic sentence. The Arabic sentence is generally classified as either nominal sentence or verbal sentence. In each case the sentence is either simple or compound. The difference between the simple sentence and the compound sentence is that the former does not have a complementary that could occur at the end of the sentence. This is best clarified by the following example:

ذ هب المزارع إلى الحقل و جمع المحصول

Complementary Simple sentence

Compound Sentence

The grammar consists of 170 rules. The rules are divided to 22 groups of rules each of which represents a grammatical category such as: Object, Subject, Defined, conjunction form, substitution form etc. This categorization is designed in such a way it helps in maintaining the grammar in a modular way.

A Grammar Rule with Simple Disambiguation Constrains

The Grammar rule

```
object(Stem,Gen,Rat)→defined
(Stem,Gen,Num,End_case,[Cat,Rat])])
```

is represented as

```
rule(object(Stem,Gen,Rat),
[defined(Stem,Gen,Num,End_case,
[Cat,Rat])]):-
        Cat\==demonstrative_noun,
        Cat\==connected_pronoun,
        End_case\==nominative,
        End_case\== genitive.
```

This rule tells us that the object can be a defined noun such that the gender and rationality of the object will be the same as the gender and rationality of the defined noun. It defines three constraints that should be satisfied in order to apply this rule during the course of parsing a sentence:

- *Semantic constraints:* the object should be neither a demonstrative noun nor a pronoun connected to a word.
- *Syntactic constraint*: object should not be in a nominative case nor genitive case.

Standard Arabic writing is unvowelled, which results in a very high level of ambiguity. However, our syntactic and morphological analyzers succeeded for resolving the ambiguity of the end case in most of the nouns. In single form that is suffixed with "Alef", it is in accusative case. In dual or regular masculine plural form, the morphological analyzer will determine whether the noun is either in nominative, or accusative or genitive case.

A Grammar Rule with Complex Disambiguation Constrains

The above rule is a simple one, but generally the rule could have more sophisticated constrains. Consider the verbal sentence rule below. This rule defines the verbal sentence that contains only verb phrase. This verb phrase could be just a verb or a verb preceded by a particle. Some constraints should be satisfied in order to apply this rule during the course of parsing a sentence:

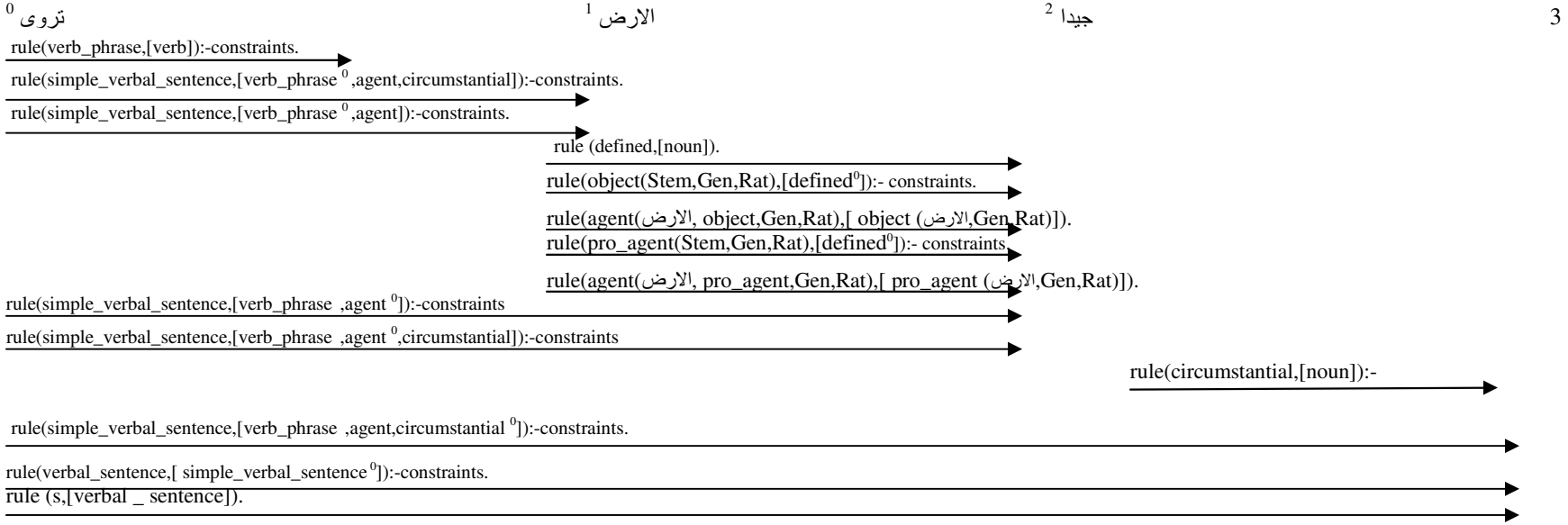
- The verb must not be ditransitive (i.e. taking two objects):
- If the analysis of the verb could not tell us whether it is in passive or active voice, we have to check the transitivity of the verb:
    - If it is intransitive, the verb must be in active voice and the Cat feature will be either connected or absent agent
    - If the verb is transitive and takes only one object, it must be in passive voice and the pro-agent will be either connected or absent. Note that when the verb has the same lexical form in both the active and passive voices, like the verb "زرع", we used semantics features to determine the correct voice.

```
rule( simple_verbal_sentence(Stem,
Time1,Gen,Num,Cat),[verb_phrase(Stem,
Time,Gen,Num,Trans,_,Agent)]):-
((\+var(Agent),Agent\==[])->
Agent=[Sub|Obj];Sub=Agent),
       Trans\==trans_2_obj,
       (Time==neutral,\+var(Trans)->
              (Trans==intrans->
                     Time1=active,
       (\+var(Obj),Obj\==[]->
       Cat=connected_subject;Cat=Sub)
       ;      Time1=passive,
       (\+var(Obj),Obj==[]->  Cat=Sub;
       Cat=connected_pro_agent)
       )
       ;( var(Trans)->true
        ; (Time==active->
                  Trans=intrans,
                  Time1=active,
   (\+var(Obj),Obj\==[]->
       Cat=connected_subject_objct;
       Cat=Sub)
; Trans=trans_1_obj,Time1=passive,
  (\+var(Obj),Obj\==[]->
       Cat=connected_pro_agent_objct;
       Cat=Sub) ))).
```

4 A Worked Example

In our work, the parser is a bottom-up parser that attempts to create a parse tree representing the syntactic structure of the Arabic sentence. The basic process is described in [Allen, 1995]. The following explains how the chart parser is used to parse the verbal sentence "تروى الأرض جيدا". The grammar rules involved are shown in Appendix (1). Fig. 4 shows the parse tree.

S		
verbal_sentence		
simple_verbal_sentence		
		circumstantial
simple_verbal_sentence		
	agent	
	pro_agent	
	object	
	defined	
	noun	
verb phrase		
verb		



** For shortness the rules is written without features*

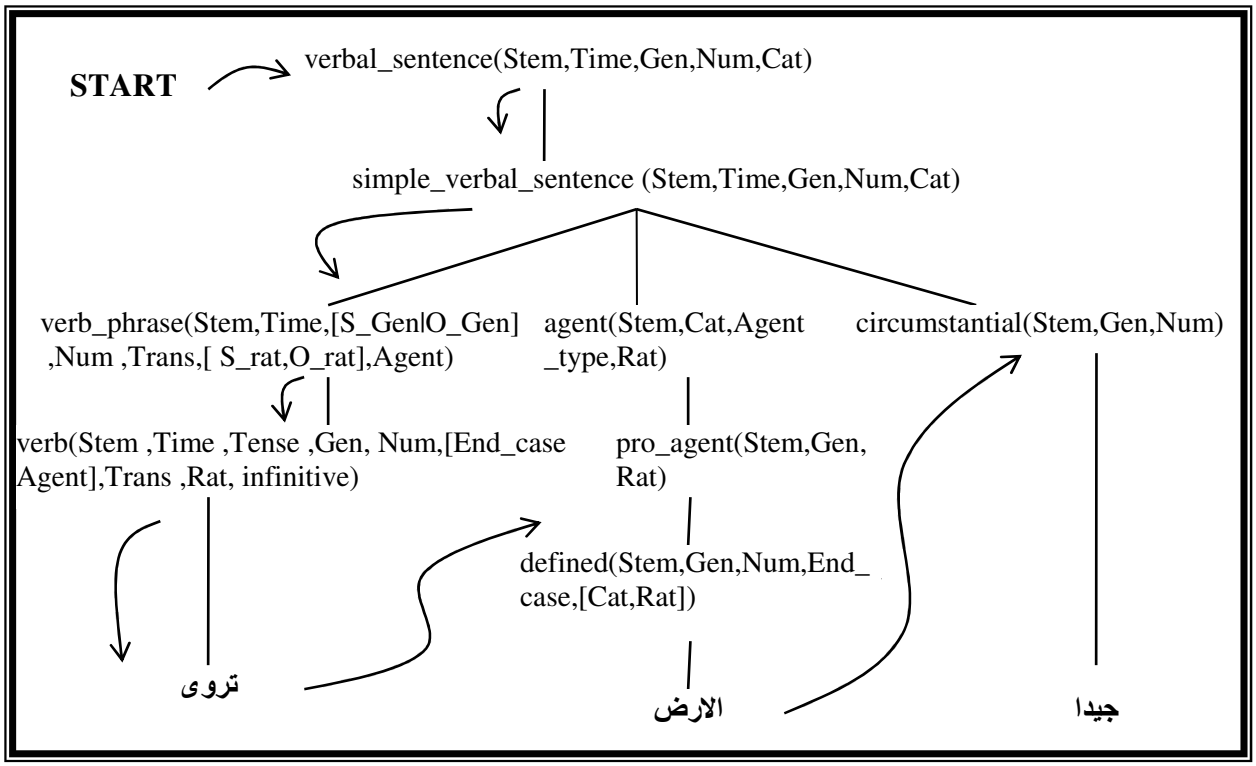


Figure (4) – Parse Tree. The parser discovers the nodes of the tree in the order shown by arrows.

5. Conclusion

This paper has been concentrated on issues in the design and implementation of a bottom-up chart parser for Arabic. We acquired the Arabic grammar rules of irab ('إعراب') and the effects of applying these rules to the constituents of the Arabic sentence. The grammar rules encode the syntactic and the semantic constrains that help in resolving the ambiguity of parsing Arabic sentences. This will have a positive impact on applications such as machine translation because the target sentence will be produced from a structure that represents the intended meaning of the source Arabic sentence.

References

1. Allen, J. 1995. *Natural Language Understanding*, second edition, The Benjamin/Cummings Publishing Company.
2. Covington M. 1994. *Natural Language Processing for Prolog Programmers*, Prentice Hall.
3. Ditters E. 2001. A Formal Grammar for the Description of Sentence Structure in Modern Standard Arabic, *In the proceeding of* Arabic *NLP Workshop at ACL/EACL.*
4. Feddag, A. 1992. *Arabic Morpho-Syntax and Semantic Parsing,* in proceeding of the 3rd international conference and exhibition on Multi-lingual Computing, Univ. of Durham, UK.
5. Jaccarini A. 2001. A modifiable structural editor of grammars for arabic processing, *In the proceeding of* Arabic *NLP Workshop at ACL/EACL.*
6. Khayat M. 1996. *Understanding Natural Arabic*, in proceeding of the First KFUM workshop on information and computer science, Dhahran, Saudi Arabic, pp. k1-k4.
7. Ouersighni R. 2001. A major offshoot of the DIINAR-MBC project: AraParse, a morphosyntactic analyzer for

unvowelled Arabic texts, *In the proceeding of* Arabic *NLP Workshop at ACL/EACL.*

8. Rafea A. and Shaalan K. 1993. Lexical Analysis of Inflected Arabic words using Exhaustive Search of an Augmented Transition Network, *Software Practice and Experience*, Vol. 23(6), pp. 567-588, John Wiley & sons, U.K.
9. Shaalan K., Farouk A., Rafea, A. 1999. Towards An Arabic Parser for Modern Scientific Text, *In Proceeding of the 2nd Conference on Language Engineering*, Egyptian Society of Language Engineering (ELSE), pp. 103-114, Egypt.
10. Woods W. 1970 Transition network grammar for natural language analysis, *comm. ACM*, Vol. 10, pp. 591-66.

Appendix (I) A sample of the Grammar Rules for the Parsing Example

```
rule (s,[verbal _ sentence(_,_,_,_,_)]).
/**********************************/
rule(verbal_sentence(Stem,Time,Gen,Num,Cat),
[simple_verbal_sentence(Stem,Time,Gen,Num,Cat)]).
/**********************************/
rule(verb_phrase(Stem,Time,Gen,Num,Trans,Rat,Age
nt),[verb(Stem,Time,_,Gen,Num,[End_case|Agent],
Trans,Rat,_)]):-
        End_case==nominative.
/**********************************/
rule(simple1_verbal_sentence(Stem,Time1,
[S_Gen|O_Gen],Num,Cat),[verb_phrase(Stem1,Time,
[S_Gen|O_Gen],Num,Trans,[_,O_rat],Agent),
agent(Stem2,Cat1,_,Rat1),circumstantial(_,_,_)]):-
    (\+var(Agent),Agent\==[]->
        Agent=[Sub|_];Sub=Agent),
    (\+var(Stem1),\+var(Stem2)->
        Stem=[Stem1,Stem2];true),
        (Time==neutral, Trans==transitive_1_obj,
        Agent==[]->
                (var(Rat1)->true
                ;
                (O_rat\==neutral, O_rat\==Rat1,
                 Rat1\==neutral->  fail
                 ; Cat=pro_agent,
                Time1=passive,Cat1=pro_agent
                )
        )
                ; (var(Time)->true;fail)
      ).
/**********************************/
rule(agent(Stem,pro_agent,Gen,Rat),
[pro_agent(Stem,Gen,Rat)]).
/**********************************/
rule(pro_agent(Stem,Gen,Rat),[defined(Stem,
Gen,_,End_case,[Cat,Rat])]):-
(Cat==adj->
          write('the pro agent can not be an
adjective'),nl,nl,fail;true),
        Cat\==demonstrative_noun,
        Cat\==connected_pronoun,
        End_case\==accusative_or_genitive,
      End_case\==accusative.
/**********************************/
rule(circumstantial(Stem,Gen,Num),[noun(Stem,undef
ined,Gen,Num,_,[End_case|_],[adj,_],_)]):-
        End_case\==nominative.
```