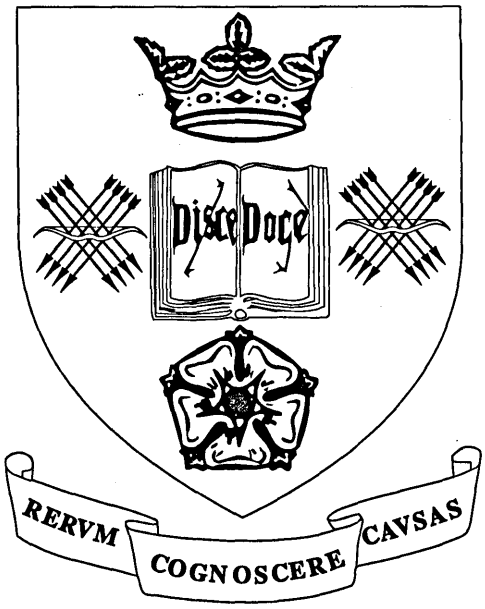


DESIGN AND IMPLEMENTATION OF AN ENGLISH TO ARABIC MACHINE TRANSLATION (MEANA MT)

Ahmed H. Alneami

Doctor of Philosophy in Computer Science

Department of Computer Science
University Of Sheffield
February 26, 1996

Abstract Of The Thesis

A new system for Arabic Machine Translation (called MEANA MT) has been built. This system is capable of the analysis of English language as a source and can convert the given sentences into Arabic. The designed system contains three sets of grammar rules governing the PARSING, TRANSFORMATION AND GENERATION PHASES. In the system, word sense ambiguity and some pragmatic patterns were resolved. A new two-way (Analysis/Generation) computational lexicon system dealing with the morphological analysis of the Arabic language has been created. The designed lexicon contains a set of rules governing the morphological inflection and derivation of Arabic nouns, verbs, verb "to be", verb "not to be" and pronouns.

The lexicon generates Arabic word forms and their inflectional affixes such as plural and gender morphemes as well as attached pronouns, each according to its rules. It can not parse or generate unacceptable word inflections. This computational system is capable of dealing with vowelized Arabic words by parsing the vowel marks which are attached to the letters. Semantic value pairs were developed to show the word sense and other issues in morphology; e.g. genders, numbers and tenses. The system can parse and generate some pragmatic sentences and phrases like proper names, titles, acknowledgements, dates, telephone numbers and addresses. A Lexical Functional Grammar (LFG) formalism is used to combine the syntactic, morphological and semantic features. The grammar rules of this system were implemented and compiled in COMMON LISP based on Tomita's Generalised LR parsing algorithm, augmented by Pseudo and Full Unification packages.

After parsing, sentence constituents of the English sentence are represented as Feature Structures (F-Structures). These take part in the transfer and generation process which uses transformation grammar rules to change the English F-Structure into Arabic F-Structure. These Arabic F-Structure features will be suitable for the Arabic generation grammar to build the required Arabic sentence. This system has been tested on three domains (sentences and phrases); the first is a selected children's story, the second semantic sentences and the third domain consists of pragmatic sentences. This research could be considered as a complete solution for a personal MT system for small messages and sublanguage domains.

ACKNOWLEDGEMENT

First of all, I would like to thank my supervisor Jim McGregor for his encouragement, guidance and his kindly support during my years of study in the department.

I am grateful to Masaru Tomita, The Associate Director of the Centre for Machine Translation, Carnegie-Mellon University, for allowing us to use his Parser/Complier and his guidance in solving some LFG equations.

Also, my thanks goes to Klaus Lagally Institute Fuer Informatik, Universitaet Stuttgart, Germany for allowing me to use the ArabTex package in this research. I wish to thank my friends who were involved in correcting and reading the generated Arabic script.

My thanks go to the departmental staff, technicians and secretaries for their help and assistance.

DEDICATION

This thesis is dedicated in the memory of my father Hussien.
Also to my mother, my beloved wife,
and my children
Meana and Mazin.

Contents

List of Figures

A Glossary of All Codes and Abbrivations Used in the Thesis

List all codes and abbreviations used in this thesis:-

ACC	Accusative.
Arabic F-structure	Arabic Feature Structure.
AI	Artificial Intelligence.
ATN	Augmented Transition Network.
CAL	Computer Aided Learning.
CFG	Context-Free Grammar.
CFPSG	Context-Free Phrase Structure Grammar.
$< CL >$	Common Lisp Prompt.
CMU	Carnegie Mellon University.
DAGs	Directed Acyclic Graphs.
DCG	Definite Clause Grammar.
DVI	Device Independent file format.
EAG	Extended Affix Grammars.
FAHQMT	Fully Automatic, High Quality MT.
FS	Feature Structure.
G	Grammar.
GEN	Genitive.
GLR	The Generalized LR Parser/Compiler.
GLR*	Version of GLR that can parse almost any input sentences by ignoring unrecognisable parts of the sentence.
GSS	Graph Structured Stack.
ILT	InterLingua features.
LFG	Lexical Functional Grammar.
LR	Parsing Algorithms, because they scan the input from Left-right and construct a Right-most derivation in revers.
MSA	Modern Standard Arabic.
MT	Machine Translation.
NLP	Natural Language Processing.
NOM	Nominative.
OSV	Object Subject Verb.
OVS	Object Verb Subject.
SL	Source Language.
SVO	Subject-Verb-Object.
SWA	Standard Written Arabic.
TL	Target Language.
VSO	Verb Subject Object.
VOS	Verb Object Subject.

Chapter 1

Summary

1.1 Introduction

Computer understanding of natural language is a difficult project. To implement such a project requires both linguistic knowledge of the particular languages being used and world knowledge relating to the topic being discussed (i.e. the domain). It is impossible to parse and translate a random sentence selected from a natural language world. Such general MT is still unattainable. Most existing machine translation systems are designed to translate off-line large documents, such as technical reports and manuals. However, there is a growing need for the translation not only of large texts but small texts such as telexes, emails and formal messages. Conventional MT systems, are intended for the translation of large texts. Our system could be referred to as a Sentence Translation System, which is a simple version of an MT system. The main aim of translation in this work is to understand the English sentence and furthermore, to enable the user to obtain the Arabic sentence. This system makes the following assumptions:

1. The user has no knowledge of Arabic at all.
2. The user is neither a computer specialist nor a linguist.
3. Human assistance is required in the form of simple post-editing.

1.2 System Architecture

Since the system is depending on a unique formalism strategy for parsing, transferring and generating, the architecture model is designed to be in

one environment. (Figure 1.1). The system modularity is very important, because whenever possible, each module has to be developed and be tested with a different grammar type. All modules are working within the Common Lisp programming Environment, with the incorporation of The Generalized LR Parser/Compiler used in CMU. In Appendix J, the main menus of MEANA MT are shown with a mini guide.

Detail of each chapter

In chapter one: *Introduction*, (this chapter) a summary of the system architecture is given.

Chapter two: *Literature Review*, we will introduce and discuss globally the current approaches to MT in general and Arabic NLP in particular. The significance of using a sublanguage for a MT is given. Types of parsing strategies for the MT and why I choose the GLR parser was discussed. The conclusion of that chapter is that only fragments of MT have been dealt with in previous work, and an Arabic MT has not been fully investigated in the last few years.

Chapter three: *Analysis of The Source Language and The Parser*, deals with the parsing methodology used in the system and details of the grammar formalism. A brief description of the Generalised LR Parser/Compiler is given. Details of how to compile the English grammar for parsing, and the LFG rules are introduced. Details are also given of the first domain which is the children's story.

Chapter four: *The Arabic Computational Lexicon*, introduces an experimental approach to create an Arabic computational lexicon. Typing every lexical rule by hand is a time consuming task. This module is divided into four submodules which are; the Noun, Verb, Verb "to be", Verb "not to be" and Pronoun inflections. All these submodules and their derivations are described in detail in this chapter.

Chapter five: *Semantic Specification*, gives a full treatment of some difficult semantic problems and how to solve these problems. Handling word sense, subject and the verb tense have been given. Other semantic attributes for the dual and plural forms are implemented.

Chapter six: *General Pragmatics*, deals with processing general pragmatic sentences or phrases. How to parse and translate some phrases

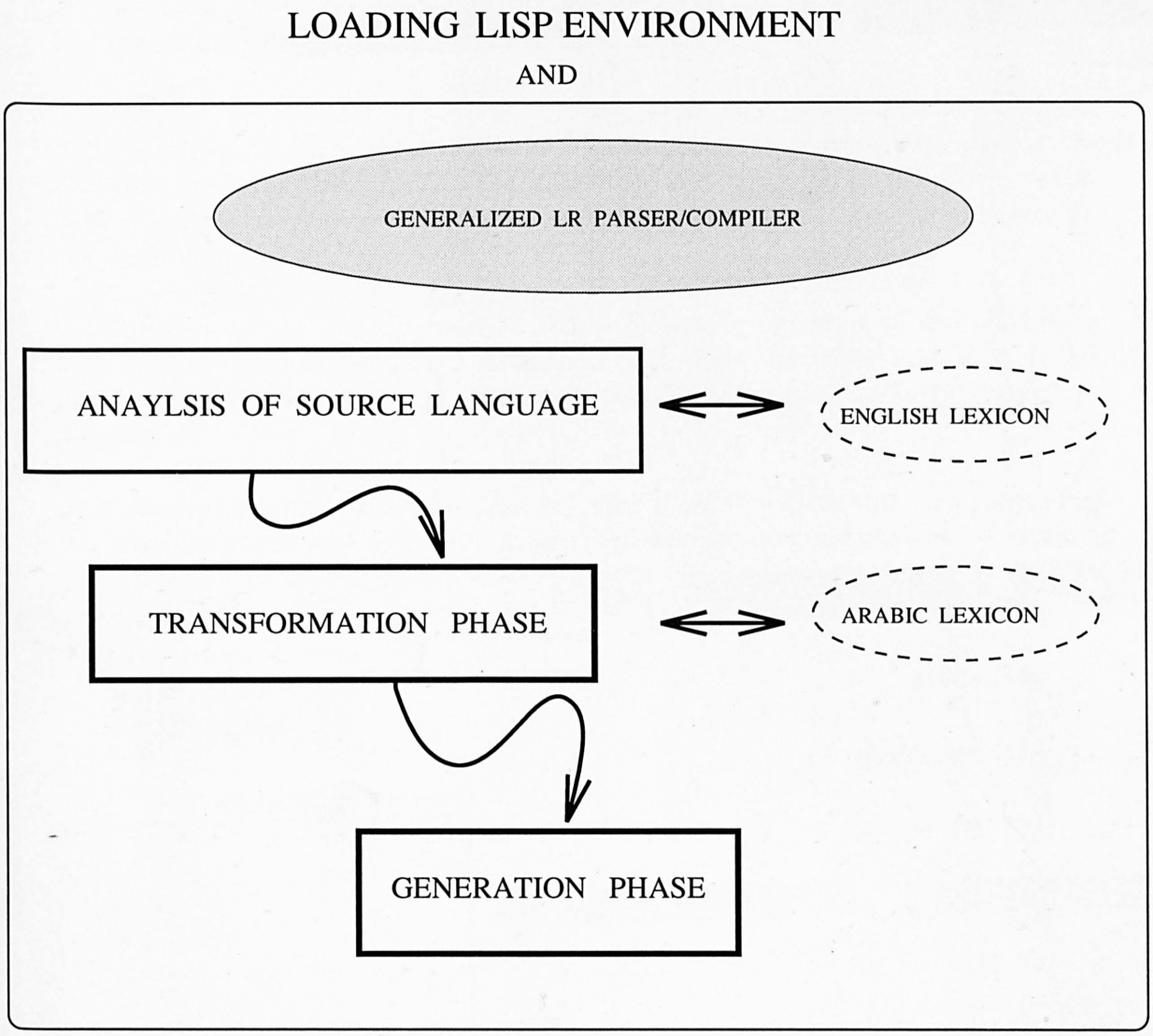


Figure 1.1: **Machine Translation Architecture (MEANA MT)**

such as Acknowledgements, the addresses, proper names, places, titles, dates and telephone number are given. A discussion of the Modern Standard Arabic forms is introduced.

Chapter seven: *The Transformation Phase*, shows the translation mapping between the English Internal Representation Features and the Arabic Internal Representation Features in detail.

Chapter eight: *The Arabic Generation Grammar*, the discussion of the Arabic common features are given. The Arabic Generator grammar rules and how to build the Arabic sentence will be dealt with in detail. Generation of different phrases in Arabic are introduced with complete examples for each domain.

Chapter nine: *Converting the Text into Arabic Script*, the discussion of how to convert the text into Arabic script. The description of the LaTeX package and the ArabTex approach which has been used are given in detail.

Finally Chapter ten: *Conclusion*, which deals with the assessment of the present work and points out some future developments and directions concerning this research.

Chapter 2

Literature Review

2.1 Machine Translation

Machine Translation (MT) has been a pursuit almost as old as computers themselves. Early systems were disappointing and little more than on-line dictionaries, but it has been hoped that as systems become linguistically more sophisticated, more interesting results will be obtained. It is the general consensus amongst the MT community that Fully Automatic, High Quality MT (FAHQMT) is still far away; some will even argue that it is impossible. The scope of this work is ambitious: we shall be examining a new approach to Arabic MT which makes the design and implementation of systems easier and more robust, and closer to providing a usable tool for human translators.

The motivation behind the work is the belief that MT is a useful testbed for linguistic theories and formalisms. There may not be a good theoretical definition of exactly what translation is, and how it differs from bilingual paraphrasing. Ultimately, we may want to say that translation is whatever professional translators do, and they are the best judges as to the performance of a system.

One other reason why it is believed that MT is worthwhile is that it forces the grammar writers to write grammars for at least two languages, and this might encourage the formalism to move away from some of the anglocentrism into which many computational linguists may be tempted.

2.2 Currents Approaches to Machine Translation

Broadly speaking, there are two methods of machine translation which are in current use. These are known as the transfer and interlingual

approaches. Both involve parsing the Source Language (SL) in order to obtain a more abstract representation of the text. The process is then reversed for the Target Language (TL) to obtain the target text from that abstract representation.

The two approaches differ in terms of what this abstract representation is: In an interlingual approach, it takes the form of an interlingual common to the two languages being translated, which typically consists of some representation of the meaning of the text. If the two representations are different for the two languages (for instance, if they are some kind of syntactic structure tree), then a transfer component is needed that maps from the SL intermediate structure into the TL one. In order to map from the SL text into the TL, one may follow two paths: one goes via the interlingual representation, the other one via a transfer component.

2.2.1 Interlingual Approaches

Interlingual approaches to MT were first proposed in the translations model of [Wea49], and later in systems such as CETA [Hut89]. These involve a parsing component that takes the source language text and maps it into an intermediate (interlingual) structure, and a generation component that maps this into the target language text. The advantage of using a common interlingual is that for translating between n languages, only n analysis and n generation components are required. The interlingual is some kind of logical representation of the meaning of the text, and the underlying idea is that translation is essentially an exercise in the preservation of meaning between the languages. Discourse and pragmatic aspects are carried through if they are taken to be part of this meaning, which must therefore be understood in quite a broad sense.

2.2.2 Transfer Approaches

In a transfer approach, exemplified by systems such as those of GETA in Grenoble [BGQA87], or SUSY in Saarbrucken [Maa87], there are essentially three components involved in the translation for each pair. An analysis component maps the input string to some representation similar to deep syntactic structure, possibly annotated with discourse and pragmatic information. A transfer component then maps this to a similar structure for the target language, and a generator then uses that to produce the TL expression. In general, the analysis and generation modules may be monolingual in that the representations used by the transfer

element are independent of the other languages in the translation pair. Consequently, in order to translate between n languages, n generation and analysis components are needed, together with $n(n-1)$ unidirectional transfer components. The transfer component is very much language-pair specific, and must be written bearing in mind both monolingual components in order to ensure compatibility.

2.3 Previous Research in Arabic NLP

There have been several different approaches towards Arabic NLP in the last decade. Most of these approaches have dealt with syntax and morphological grammar only. I will mention some of these only within that range. In the beginning, the term Modern Standard Arabic (MSA) refers generally, to the uniform variety of Arabic which is used all over the Arab world as the usual medium of written communication in contemporary books, periodicals, magazines, newspapers, science, business and personal letters. A single Arabic word could be a whole sentence, whereas in English a minimum of two words are required.
There was an initial attempt at constructing a transfer grammar between English and SWA referred by Satterthwait's work [Sat62]. It was conducted in connection with the Mechanical Translation group in MIT. In his thesis, he describes two programms to direct a computer to compose sentences in Arabic from English. (i.e. Mechanical Translation programme). His programmes produced many sentences and sequences which were nonsense in Arabic. This was mainly due to his grammar and to the lack of vowels and other diacritical marks.

A second attempt to provide a formal simple grammatical approach was made by Snow [Sno65]. Abdul-Hamid [AH72], presented a study of a Transfer grammar of English and SWA on the syntactic level. The study was based on a strictly synchronic description of both languages and followed modern linguistic principles. His work only dealt with simple sentences. It consisted of a formal description of the two languages.

Al-Sweel [Als83], investigated the word order variation in SWA. In his study, the lexically based theory was applied to word order variations. It has been established that SWA has two orders. These are Verb-Subject-Object (VSO), and Subject-Verb-Object (SVO). Al-Sweel showed the need for more than one word order in SWA, and showed that any of the clause constituents, i.e. Verb, Subject or Object may precede any other for pragmatic reasons.

Other systems such as Arabic Language Interpretation (ALI) [Meh87],

accept Arabic declarative sentences concerning the Iraq-Iran war and translate them to a case frame semantic representation and finally to English sentences. The model involves simple Arabic verbal sentences of VSO type, interrogative sentences, and nominal sentences. An Arabic language parser was developed for syntactic parsing of Arabic sentences based on Definite Clause Grammar (DCG) formalism [Meh86].

The El-Sadany and Hashish system [ESH89], named "An Arabic Morphological System" was developed at IBM Cairo Scientific Centre. That was a two-way Arabic Morphological System (analysis/generation) capable of dealing with vowelized, semivowelized, and nonvowelized Arabic words. The main objectives of the work described in this paper can be summarised as follows:

- Stress the need for research in the field of Arabic computational linguistics in general and morphology in particular.
- Specify and define a set of basic criteria to classify the previous work on Arabic computational morphology and act as a guide for the present and future work in this field.
- Model the rules governing the morphological inflection and derivation of the Arabic language (i.e., the Arabic rules of grammar).
- Develop an Arabic computational lexicon that contains all the Arabic roots with their associated information, to be used by the morphological system and for other linguistic tasks.
- Develop a two-way (analysis/generation) morphological system capable of dealing with vowelized, semivowelized, and nonvowelized Arabic texts.

This system consists of three separate models: computational lexicon, Arabic grammar module, and analyser/generator module. The grammar module contains, among others, morphophonemic and morphographimic rules formulated using the conventional generative grammar. Moreover, the developed system covers all of the Arabic language. The system has been written using Prolog. An Arabic computational lexicon has been developed as a specific application. The computational lexicon is tailored to provide lexical information for the Arabic morphological system, which consists of 5700 distinct lexical entries. This system with the generator provides good interactive Computer Assisted Learning (CAL) for teaching the Arabic language, but only verbs have been provided for native Arabic speaking students.

Feddag and Foxley [FF93], developed a lexical analyser for a practical NLP system such as a database front-end. The lexicon stores only roots of verbs and uses a programme intelligent enough to handle all derived forms automatically. This is significant as these alone represent 70% of the total dictionary. This front-end model can be used in various applications such as an Arabic spell checking system or for teaching Arabic grammar to non-Arabic speakers.

The Extended Affix Grammars (EAG) formalism [Mei90] consists of two levels of context-free grammars. Both levels contain non-terminals, terminals and a set of rules. The formal definition of Extended Affix Grammar is given by Koster [Kos71]. Modern Standard Arabic (MSA) [Dit92], gives a formal description of the noun phrase and verb phrase in MSA syntax.

Ebrahim, Clarke and Fahmy [ECF89], had considered the possibility of using Arabic in building MT. Their paper describes the concept of such a MT with reference to the logic programming technique for building such a system for MT. They assumed their tool could be used to cater for morphological aspects of Arabic in both the analysis and generation stage in MT. It is not clear how far they went nor are details given.

Roochnik [Roo89] proposes an "ARABATN: A Morphology-Syntactic Parser for Arabic". This system is an MSA parser written in Lisp based on Augmented Transition Network (ATN). It is not MT, but could be a module of it if adopted by an MT system.

Beesley (et al) 1989 [BTS89], [Bee90], developed a computer programme at ALPNET that takes On-line words written in Modern Standard Arabic and identifies all of their valid morphological analysis. That programme is based on Koskenniemi's Two-Level morphology [Kos83], [Kos84], [Kar83]. This programme can produce rough English translation and is intended to become part of an Arabic to English MT system.

Farhat [Far89] proposed a Contemporary Arabic Dictionary called (CALDB). It contains a complete up-to-date inventory of Arabic Lexical items with the relevant information required by language users and computational linguists. The CALDB database did not have the ability to show the diacritical marks.

A fast algorithm is proposed by Ebrahim (et al) [EC90] which replaces each figurative expression (such as metaphors, idioms and proverbs) in the source text, if possible by the corresponding figurative expression in the Arabic language or, if not, by the corresponding literal expression in the target language.

Another model for Arabic morphology as a generation system has been proposed by Shahein and Yousef[SY90]. In this model, a new technique

called "EC-Hashing" has been proposed to accelerate the analysis process using an ATN grammar. This could be useful for CAL or tutor systems. Extraction of the syntactic and morphological information from Arabic words was dealt with in Feddge and Foxley [FF90].

2.4 Previous Arabic MT Systems

This section investigates the applicability of current and foreseeable MT technology to translate between one or more western European languages and Arabic. I will present the current status of state-of-the-art Arabic MT systems, and outline some of these approaches. The following systems have been suggested in the conference organised by the King Abdulaziz city for Science and Technology in Riayah (1985).[1]

2.4.1 Michael Downs Associates System

Michael Downs Associates has conceived a design for an optimum English-to-Arabic Translation System, especially for use in the Kingdom of Saudi Arabia [Ewe85]. Basically, the computer steps through four main processes to accomplish its raw translation:

Pre-Analysis: is the process of breaking the source text into the smallest units which are meaningful to the computer. These might be phrases, words, or one word, depending greatly upon the complexity of the source sentence structure. Performed in two steps, the computer first scans the source document with the pre-processor routine which isolates sentence units, inserts formatting markers and identifies typesetting commands. Then the dictionary module morphologically analyses source words to remove infection and also locates the definition of the stem forms of target words in the system dictionary. Inflection information is preserved for later use.

Analysis: is the next phase. During this phase, the computer reconstructs the small individual units into the next higher level of meaning. Linguistically, this is called parsing. The compacting routine isolates word groups that act as a single unit such as verb strings and idiomatic phrases. These word groups are then compacted into a single unit for the computer to deal with and are further flagged with syntactic and semantic information for later analysis. Homographs are examined within their context to determine which meaning should be selected. Finally,

[1] Up to my knowledge, there is no evidence that any system has been implemented yet.

the phrase structure analysis routine performs a left to right parsing of the source text, identifying higher and higher levels of phase structure. Phrase and clause structure information generated by this analysis is preserved for subsequent reorganisation into their corresponding target language structure.

Transfer: is the third phase of machine translation and is where the structure of the source language is transformed into the target language structure. The insertion routine inserts articles, prepositions, etc., into the text as required by the target language.
Synthesis: is the final phase of machine translation. Here the translated text receive its final touches. Inflection of verbs, adjective, determiner, nouns, etc., is performed to provide agreement between sentence elements. Then the cosmetics routine makes the miscellaneous orthographic and other necessary, low-level adjustment that are required in the target language output. Finally, the post-processor puts the machine translation into the format of the original text, restores typesetting commands and performs special handling of untranslated words or phrases. The machine translation is then stored in an output file, awaiting editing and/or supervisor approval.

This system is a dictionary based system. Several dictionaries can be used on-line simultaneously, including a core-dictionary that is supplied with the system; industry/discipline specific terminology dictionaries; and user specified terminology. This was a theoretical approach for Arabic MT and there is no indication that the system has been implemented.

2.4.2 SYSTRAN System

The SYSTRAN System is a very big and sophisticated system which deals with multilingual environments. The English-Arabic system was an abstract module suggested to link with the SYSTRAN system [Tra85]. Automatic Translation process involves several operations, and each operation must be done in a cost effective way, from inputting until outputting. SYSTRAN has the following properties:-

1. At the input the system is connected a self training scanner which can read 16 fonts and can be taught others.
2. Errors in the source text were unintelligible for the computer, it adopted a self detecting errors programme to point to each word which is not in the dictionary.
3. The SYSTRAN system is implemented on an IBM computer allowing very fast translation at a speed of 300.000 words an

hour.

4. The translation may be then post-edited on a terminal, the upper window of which contains the source text and the lower on the Arabic Text.

More details in [Tra85].

2.4.3 GETA System

The GETA system is designed for handling multi-lingual translations at various degrees of automation [Vaq85]. The GETA system is designed for handling multi-lingual translations at various degrees of automation. It can be used both extremes: either as aid for human translation using software tools for dictionary look-up and editing, or as a purely automatic translation without any human interface. In fact, the only experimental production of translations made with this system (Russian- French translation for scientific abstracts in various fields such as aircraft technology, metallurgy, energy ... etc.) uses a combination for both types.

It is not possible to have human interaction during the automatic translation phase, except (by an optional mode) for introducing and coding new lexical entries in the monolingual dictionary of the source language. The multilingual aspect of the system comes from the fact that the result of the automatically computed analysis of the input text can be transferred to several models of generation corresponding to the various target languages. To reach this aim, GETA has developed:

- A basic software called ARIANE-78;
- A method for building linguistic models; item Various applications of this method to several prototypes.

Framework for grammars considering a sentences as a unit for translation, the automatic translation process has to perform three phases:

1. Analysis: this means the transformation of the input text into a structural description as seen just above. This phase is divided into two sequential sub-phases, namely morphological analysis. The morphological analysis yields a labelled tree structure.
2. Transfer: also divided into two parts, this phase begins with a lexical transfer (a bilingual dictionary look-up) followed by a structural transfer. The latter is mainly an extension of the

lexical transfer to solve ambiguities by means of a larger context exploration and computes the contrastive aspects between the source and target languages (for example, contrastive values in tense, mode, aspect, determination ... etc.). Finally, if necessary, the structural transfer computes some surface levels of interpretation whenever the analysis phase failed to reach the level of logical and semantic relations.

3. Generation: also divided into two parts, namely a syntactic generation followed by a morphological generation. The former selects a syntactic structure by computing the new syntactic functions and the new syntagmatic classes of the target language from the only values of logical and semantic relations labelling the tree structure. The compute form of the words built by concatenation (with possible alteration) of the segments (roots, prefixes, suffixes, ... etc.) is performed by the morphological generation.

This approach was suggested to implement English-Arabic MT. Only two sentences were analysed and generated. Nothing shows such a system has been implemented.

2.4.4 METAL System

METAL [LBS85] translates the German language as a source and returns its value in equivalent sentences in English. Another proposal was made to link the Arabic language to the METAL system [SA85].

2.5 The Significance of a Sublanguage for Automatic Translation

Kittredge's paper [KL82] addresses three questions:

1. What is a sublanguage?
2. Why is a sublanguage analysis important for automatic translation?
3. How can a translation system take advantage of the sublanguage properties?

The first of these questions appears to have a simple answer. Natural languages clearly have specialised varieties which are used in reference

to restricted subject matter. We speak, for example, of the language of chemistry to mean a loosely defined set of sentences or texts dealing with a particular part of reality. But when we consider the automatic translation of specialised language, we are forced to be more precise. We must describe sublanguages as coherent, rule-based systems. The attempt to write grammars for special-purpose sublanguages raises a number of theoretical and practical problems, which are only now being intensively discussed. But since the only path to high-quality automatic translation seems to lie through a sublanguage (at least during the next decade or two), We have no choice but to solve these problems.

2.5.1 What is a Sublanguage?

Two definitions: In the science of linguistics, one of the most difficult problems has always been the one of definitions. Language can be viewed from so many different perspectives that no single definition of a basic term such as sentence or noun is likely to suffice to characterise all aspects of the term. To further complicate matters, language is often a fuzzy phenomenon. One is often unable to say, for example, whether a particular sentence made up of English words is or not good English .

It is therefore not surprising that the study of sublanguage meets the same definition problems that arise in general descriptive linguistics. But looking at language in restricted domains gives a much better picture of the relationship between language and information than is the case when we study the whole language.

In Kittredge's paper, he informally defines a sublanguage to be any subsystem of a language which has the following properties.

1. "The language subsystem is used in reference to a particular domain of discourse, or family of related domains,
2. The set of sentences and texts in the language subsystem reflect the usage of some community of speakers, who are normally linked by some common knowledge about the domain (facts, assumptions ... etc.) which goes beyond the common knowledge of speakers of the standard language.
3. The subsystem has all the essential properties of a linguistic system, such as consistency, completeness, economy of expression and so forth.
4. The language subsystem is maximal with respect to the domain, in the mathematical sense that no larger system has the same properties."

The definition is vague on a number of points, but serves to indicate some of the important precise theoretical definitions of a sublanguage as discussed by Harris [KL82]. Certain proper subsets of sentences of a language but his definition and thus constitute sublanguages of it .

2.5.2 Why is a Sublanguage Important for Automatic Translation?

Linguistically, we are now in a better position to see just why and how sublanguage study is useful for automatic translation. Let us first summarise the major points about using sublanguages.

First, the notion of sublanguage is a theoretical construct which stresses the systematic nature of specialised language. In a sublanguage, the rules for constructing meaningful sentences can be made much more precise than in the language as a whole. Most importantly, these rules can be made in terms of word classes which are discovered by studying exactly how language is used in the particular domain (i.e., studying the distribution properties of words in texts).

Second, in a sublanguage systems the rules for constructing sentences may be quite different from (and even contrary to) the rules for sentences of the standard language. The grammar of standard English does not contain the grammars of all English sublanguages, because some structures or operations exist only in particular sublanguages and have no rule in standard English grammar.

Third, sublanguages may be rather small (e.g., weather bulletins), or very large (e.g., texts in aircraft hydraulics or organic chemistry). What qualifies a variety of language as a sublanguage is not its size or complexity, but its adherence to systematic usage. The well-behaved sublanguages of science and technology may use terminology from the everyday world, but this seepage from general language is usually possible only in specific grammatical positions. Kittredge admits that some sublanguages appear to be more systematic than others. It is in fact the DEGREE of systematicality which will determine how amenable a sublanguage is to automatic translation.

For predicting the problems of making correct translation correspondences, much less is known for full natural language. The experience of the University of Montreal (TAUM project), which concentrated entirely on English-to-French translation, showed that some of the following grammatical and semantic phenomena were generally problematic in establishing correspondence between the two languages. In most cases, the restriction of the correspondence problem to a technical sublanguage

allowed fairly good rules to be set up[KL82] :

1. Tense and aspect: most English sublanguages use a subset of possible forms, and their functional equivalents in French are both idiosyncratic for the French sublanguage, and simpler than in general French;
2. Verb modality: translations of English can, must, should, etc. present many problems for general language that can be solved in specific sublanguages.
3. Passive: although English passive has at least six possible renderings in French, within aircraft hydraulics manuals, the correspondence algorithm is far simpler.
4. Lexical choice; there is a complex interaction between structural transfer rules and the valency (i.e., semantic case slots) of available verbs in the target language; this complexity is usually much reduced within the limits of a given sublanguage.
5. Textual constraints: this is one area where corresponding technical sublanguages of English and French were found to share many of the same features; thus, textual constraints of the target language must be used properly to give a natural-sounding text.

Some or all of these concepts could be taken in to consideration when creating or using the sublanguage in a MT.

2.6 Types of Parsing Strategies for On-line Machine Translation

Research has been done by Patrick Shann [Sha89], on the real world performance of on-line parsers in connection with different grammar types rather than on the theoretical space and time complexity of the parsers. The design of the MT system required controlled input and no post-editing of the translated texts. In order to guarantee correct translation, the input system accepts only words and sentences that are accepted by their grammar and dictionary and it refuses illegal input. The sentences are parsed word by word from left to right so that any illegal input is detected immediately. After each word, the system has to predict, with the help of the parser, all the words that can possibly continue the sentence that is being made. The type of on-line parser time is considered

very critical. The interface window has to be refreshed immediately after each word chosen by the user. The grammar type can heavily influence the overall processing efficiency.

Shann's work involved choosing the best and the most efficient on-line parsing algorithm for developing a machine system for translating Avalanche Forecast Bulletins from German to French. Their research was to investigate the possibility of choosing one parser from various parsing algorithms such as Top-Down Filter (TDF), Look-AHead (LA), Rule compilation, Tomita's extended LR parser (Tom), and different Chart Parsing Techniques were used as follows:

1- Top-Down (TD),
2- Left-Corner (LC),
3- Bi-directional (BI),

Details of the report is given in [Sha89] but our concern is the comparison result which is given below.

2.6.1 Patrick's Comparison Result

Since the goal was the selection of a suitable parsing strategy for the on-line MT system, and since the time is critical, one of the important questions was what combination parser and rule-body procedure is best for that purpose. In the comparison Tomita vs. Earley's algorithm. Tomita has used pure context-free phrase-structure grammars (G) consisting of a varying number of rules: G1 8, G2 40, G3 220 and G4 400 rules. These grammars were tested with two sets of sentences (S), S1 :40 sentences from texts and S2 :13 artificial sentences that have an increasing number of prepositional phrases. These artificial sentences are useful for testing growing sentence ambiguity since the number of parses grows exponentially[Kay85]. From Patrick's report [Sha89], I will quote the final decision: "possible candidates for an on-line parser that parses strictly from left to right are Tom, LC + TDF and TD. The grammar type we have chosen is the "Tomita parser", because it was slightly more preferable with the large grammar for the Avalanche corpus, and last, but not least, because of its elegance".

Also, Tomita himself has implemented an experimental system by extending an existing English-Japanese machine translation system to demonstrate feasibility of the interactive approach to machine translation.

2.7 Conclusion

The effective application of Arabic MT will require hard work, but need not start from scratch. Some current systems in NLP are available for adaptation; this offers the opportunity of starting with substantial linguistic models of source languages (e.g. French, German, Spanish, Japanese etc.).

From the above survey, we can say that none of the above systems can produce English-Arabic MT. Most of the published research deals with lexicon, morphology and parsing. No generation system or subsystem is available for Arabic. Papers whose published by King Abdulaziz city for science and technology were more or less a theoretical concepts to link other MT system to the Arabic language. Till now no evidence that any of them has been implemented. After a close study of the possible English parser or tool to adopt for the practical work, I found that Tomita parser and tool is the efficient and faster algorithm for this purpose. Other tools like Alvey tools a bit complicated and gives a lot of internal representations to the sentence which I think unnecessary.
From the sublanguage discussion, I have dealt with the selected children's story and I have applied other definitions for other semantic problems which are not found in the story. Also a full description of some of the pragmatic sentences has been tested in the system. The grammar formalism and the chosen tools was selected according to careful consideration of recent research in MT. As indicated by Patrick Shann [Sha89], their selection was made according to different criteria such as size of the grammar, time and efficiency. This new system can be considered as a first step to building personal[2] MT from English to Arabic. It converts an English sentence into Feature Structure (FS), which are mapped to Arabic FS; then special Arabic generator grammar rules will flesh the Arabic sentence with its new constituents with respect to Standard Written Arabic grammar and lexical data. This system uses unique formalism rules for parsing, transferring and generating. These have been designed for in use unsegmented languages, such as Japanese [TMMK88], in which there are no boundary spaces between words; and now, I propose to use them in our system for Arabic.

[2]A "personal" could be a personal messages or a local related subject or topic or related business

Chapter 3

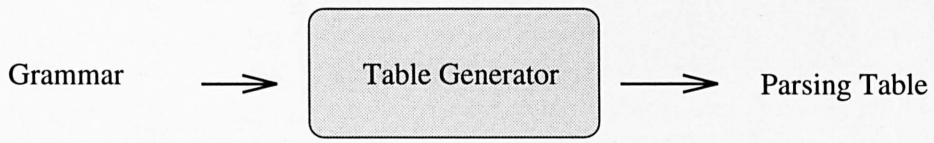
Analysis of The Source Language and The Parser

3.1 The Parsing Methodology

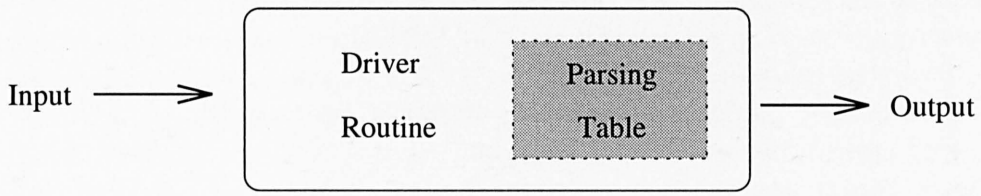
The LR[1] parsing algorithms [AU72], [AU77], were developed originally for programming languages. The LR parsing algorithm is a shift-reduce parsing algorithm which is deterministically guided by a parsing table indicating what action should be taken next. The parsing table can be obtained automatically from a context-free phrase structure grammar, using an algorithm first developed by DeRemer [DeR69], [DeR71]. That algorithm is totally deterministic and no backtracking or search is involved. Aho [AU77], stated a variety of reasons for the attractiveness of the LR parser. I will quote them below:

- "LR parsers can be constructed to recognise virtually all programming languages constructs for which context-free grammars can be written.
- The parsing method is more general than operator precedence or any of the other common shift-reduce techniques discussed in the last chapter, yet it can be implemented with the same degree of efficiency as these other methods. LR parsing also dominates the common forms of top-down parsing without backtrack.
- LR parsers can detect syntactic errors as soon as it is possible to do so on a left-to-right scan of the input."

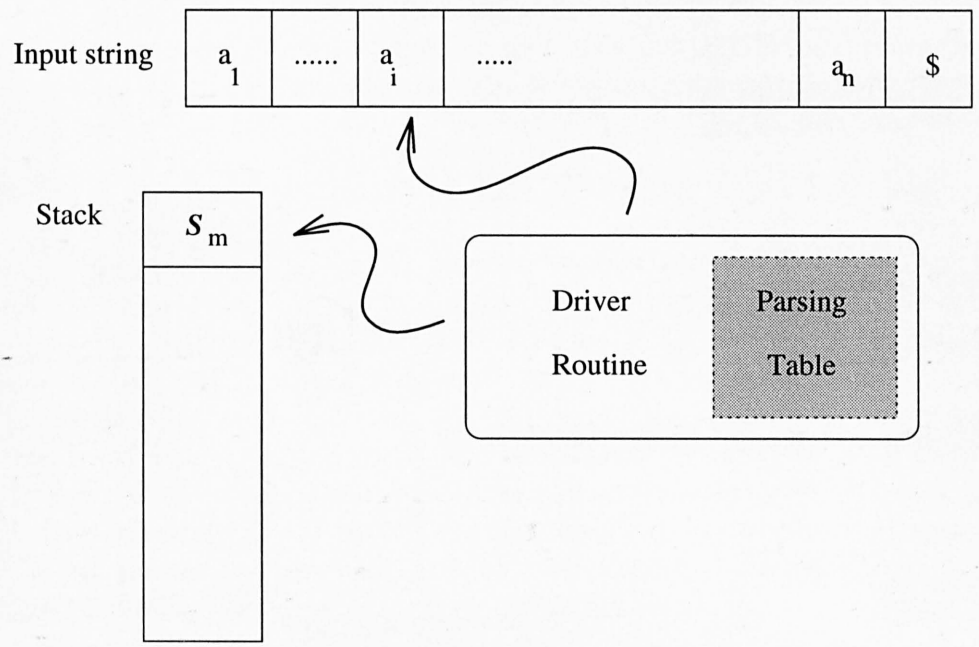
[1]Called LR parsers because they scan the input from Left-right and construct a Rightmost derivation in reverse.



(a) Generating the parser.



(b) Operation of the parser.



(c) LR parser.

Figure 3.1: **Generating an LR Parser.**

Logically, LR parser consists of two parts, a driver routine and a parsing table. The schematic form of an LR parser is shown in Figure 3.1 (a and b). LR parsers use a "Bottom-up" parsing method, called sometimes "Shift-Reduce" parsing, because it consists of shifting input symbols onto a stack until the right side of a production appears on top of the stack. The right side may then be replaced by (reduced to) the symbol on the left side of the production rules and the process repeated. Figure 3.1 (c) depicts the parsing of an input string. The stack contains a string of the form $s_0X_1s_1X_2s_2...X_m,s_m$, where s_m is on top. Each X_i is a grammar symbol and each s_i is a symbol called a *state*. Each state symbol summmarises the information contained in the stack below it and is used to guide the shift-reduce decision. The parsing table consists of two parts, parsing action functions "ACTION" and goto functions "GOTO". The programme driving the LR parser behaves as follows: it determines s_m, the state current on top of the stack, and a_i, the current input symbol. It then consults ACTION$[s_m, a_i]$, the parsing action table entry for state s_m and input a_i. Then entry ACTION$[s_m, a_i]$ can have one of four values (Shift, Reduce, Accept or Error). (The algorithm and examples are given in [AU77], Ch6).
Unfortunately, we cannot directly adopt the LR parsing technique for natural language. This is because not all Context-Free Phrase Structure Grammars (CFPSG's) can have an LR parsing table.

3.2 The Generalized LR Parser/Compiler

The Generalized LR (GLR) parser, developed by Tomita [Tom86], extended the original LR parsing algorithm to the case of non-LR languages, where the parsing table contains entries with multiple parsing actions. Tomita's algorithm uses a Graph Structured Stack (GSS) in order to efficiently pursue in parallel the different parsing options that arise as a result of the multiple entries in the parsing tables. The second data structure uses pointers to keep track of all possible parse trees throughout the parsing of the input, while sharing common subtrees of these different parses. A process of local ambiguity packing allows the parser to pack subparses that are rooted in the same non-terminal into a single structure that represents them all. Details of the Tomita's extended algorithm is shown in [Tom86].
The GLR parser is the syntactic engine of the Universal Parser Architecture developed at Carnegie Mellon University (CMU) by Tomita et al [TMMK88]. The architecture supports grammatical specification in Lex-

ical Functional Grammar (LFG) framework that consists of context-free grammar rules augmented with feature bundles that are associated with the non-terminals of the rules. Feature structure (F-structure) computation is, for the most part, specified and implemented via unification operations. This allows the grammar to constrain the applicability of context-free rules. The result of parsing an input sentence consists of both a parse tree and the computed feature associated with the non-terminal at the root of the tree.
The grammar is interpreted on a character-by-character basis rather than a word-by-word basis; that is, the final symbols of the grammar are characters, not words. It is so designed with a view to the use of this system for unsegmented languages, such as Japanese [TMMK88], in which there are no boundary spaces in between words and now, in our system for Arabic as well as for English. As a result of this character-based feature, the lexical dictionary and the morphological rules can be written in the same formalism as the syntactic rules. The runtime parser cannot parse a sentence without a compiled grammar.

3.2.1 Grammar Formalism

Each grammar rule of GLR consists of a context-free phrase structure rule followed by a list of equations representing the features. The list of equations is enclosed in parentheses as follows:

(context-free phrase structure rule (list of equations))

The context-free phrase structure rule used in this system consists of a left-hand side, an arrow, and a right-hand side. The left-hand side must be a single non-terminal symbol. The arrow is one of the following: <==>, ==>, <==, -->, <--> or <--. The type of arrow used indicates how a rule is to be used, for example either for parsing or as a sentence generator[2]. See the following example of a grammar rule for the nominative sentence.

```
(<SENTENCE>  ==> (<NP> <VP>)
            (((X1 case) = *nominative)
             ((X1 agreement) = (X2 agreement))
             ((X0 subj) = X1)    ; Move <NP> features to sentence.
              (X0 = X2)))        ;Move <VP> features to sentence.
```

[2]In that case, the double headed arrows mean that the rule can be used for generation and parsing. The arrow pointing to the left means that the rule can be used only for parsing, and the arrow pointing to the right means that the rule can be used only for generation

The above rule show that the parsing of the sentence could have NP followed by VP which satisfies the agreement X1 and X2 between each of them. The sentence "case" will have the nominative in the final feature structure. The variable X0 is referred to SENTENCE, X1 and X2 are referred to NP and VP subsequently. The `((X0 subj) = X1)` means: make the sentence head with "subj" in the F-structure.

3.2.2 Equations

Equations for the GLR are very similar to LFG equations except that they use the variables X0,X1,X3 ...etc. in place of the up-arrow and down-arrow. X0 takes the place of the up-arrow. It refers to the functional structure corresponding to the left hand side non terminal of the phrase structure rule. X1 takes the place of the down-arrow, referring to the second element of the right hand side of the phrase structure rule, and so on. Here is an example comparing an LFG rule with a GLR rule:

LFG RULE:

$$\begin{array}{lll} S \text{ --> } & NP & VP \\ & (\uparrow SUBJ) = \downarrow & \uparrow = \downarrow \end{array}$$

The Generalized LR Parser/Compiler rule:

```
(<S> ==> (<NP> <VP>)
         (((X0  subj) = X1)
          (X0 = X2)))
```

The left hand side of an equation is a path. A path is:

- A variable (e.g. X0, X1, X2, etc.)
- A variable followed by any number of character strings separated by spaces (e.g. (X1 subj), (X0 agreement) (X2 xcomp subject)). The character strings must not include certain special characters such as the quotation mark. This type of path must be enclosed in parentheses.

The right hand side of an equation is:

- A path.

- A character string (e.g. foot, headache, 12), excluding some special characters such as the quotation mark.
- A list consisting of the word *OR* followed by any number of character strings (e.g. (*OR* nominative accusative), (*OR* past past-participle)).

3.2.3 The Start Symbol

The left-hand side of the first rule of the grammar is the start symbol, using the same kind of equations which are used for defining phrase structure rules. Defining the start symbol allows the grammar writer to decide if the system parses full sentences, or accepts both sentences and sentence fragments (phrases), or only accepts phrases; more details are given in [TMMK88]. For example, in order to parse a full sentence as well as noun phrase fragments, the grammar would have to begin with following two equations:

```
(<START> ==> (<S>)
             ((X0 = X1)))

(<START> ==> (<NP>)
             ((X0 = X1)))
```

The left hand side of a start equation contains the non-terminal <start>, and the right hand side consists of a single non-terminal symbol, which designates some constituent of the phrase-structure grammar.

3.2.4 Pseudo and Full Unification

The GLR parser/compiler v8.5 has two different modes for Unification[3] implementation: PSEUDO Unification or FULL Unification. Full unification is the standard unification. PSEUDO unification is suggested by Tomita and Knight [TK88], as an alternative way to implement unification; it does not do unification, but does something very close to it. The implementation of pseudo unification is simpler and more efficient. Details of pseudo equations like: Disjunction equation (*OR*), Constraint equation (=c) and others used in the pseudo Unification Grammar are found in [TMMK88]. In fact it produces the same result as full (Canonical) unification most of the time, and it does not seem to present any

[3]Unification is a process which forms the union of two feature sets, and which detects value conflicts between them.

problems in practical applications, such as Natural Language interfaces and MT. On the other hand, feature structures in pseudo unification are always simple trees rather than Directed Acyclic Graphs (DAGs). Thus, the implementation of pseudo unification is much simpler, bypassing all the tough problems caused by DAGs. Detail can be found in [Tom86].

3.3 GLR* The Modified Parser

GLR* is a recently developed robust version of the Generalized LR Parser, that can parse almost *any* input sentence by ignoring unrecognizable parts of the sentence. On a given input sentence, the parser returns a collection of parses that correspond to maximal, or close to maximal, parsable subsets of the original input. This work was described by Lavie [LT93]. His recent work is on developing an integrated heuristic scheme for selecting the parse deemed "best" from such a collection. He described the heuristic measures used and their combination scheme. Preliminary results from experiments conducted on parsing speech recognised spontaneous speech are also reported.

3.4 The Analysis Phase

The English grammar is built to deal with children's stories, semantic and other pragmatic sentences. But before using this grammar we need to prepare the grammar for parsing. This process is called compilation. The system is connected to an English Lexicon which will be available when the parser requests a lexical entries about the words. (See Figure 3.2 for full description of the the analysis phase.)
To begin with, let us illustrate the process by writing a simple grammar and compiling it for parsing. Below is a simple grammar for the phrase "A mouse's gratitude" [4] (which is the title of a children's story used in testing this work):-

```
;Simple  grammar for the phrase  "A mouse +s gratitude".
(<SENTENCE>  ==> (<SIMPLE-TITLE>)
                 (((X0 mood) =  *nominative)
                  (X0 = X1)))

(<SIMPLE-TITLE> <==> (<NP> <NP>)
```

[4]The "+" symbol will represent the possessive case, that is because the apostrophe is interpreted differently in Common Lisp.

```
                (((X0 action) = X2)
                 ((X0 subject) = X1)))

(<NP> <==> (<DET> <N> <POS>)
           (((X1 number) = (X2 number))
            ((X0 definite) = X1)
            ((X0 noun) = X2)
             (X0  = X3)))

(<NP> <==> (<N>)
          ((X0  = X1)))

;*********** The Lexical entries *******
(<N>  --> (g r a t i t u d e)
         (((X0 root) = gratitude)
          ((X0 number) = sg)
          ((X0 agreement) = 3sg)
          ((X0 cat) = noun)
          ((X0 subcat) = trans)
          (X0 = X1)))

(<N>  --> (m o u s e)
         (((X0 root) = mouse)
          ((X0 number) = sg)
          ((X0 gender) = both)
          ((X0 cat) = noun)
          ((X0 agreement) = 3sg)))

(<POS>  --> (+ s)
          (((X0 root) = +s)
           ((X0 number) = sg)
           ((X0 subcat) = trans)
           (X0 = X1)))

(<DET>  --> (a)
          (((X0 root) = a)
           ((X0 definite) = yes)
           ((X0 number) = sg)))
```

The grammar rule with the symbol ==> indicates that, the rule is for a non-terminal node. For a terminal word, there must be spaces placed between the elements of the right hand side. Although spaces must be

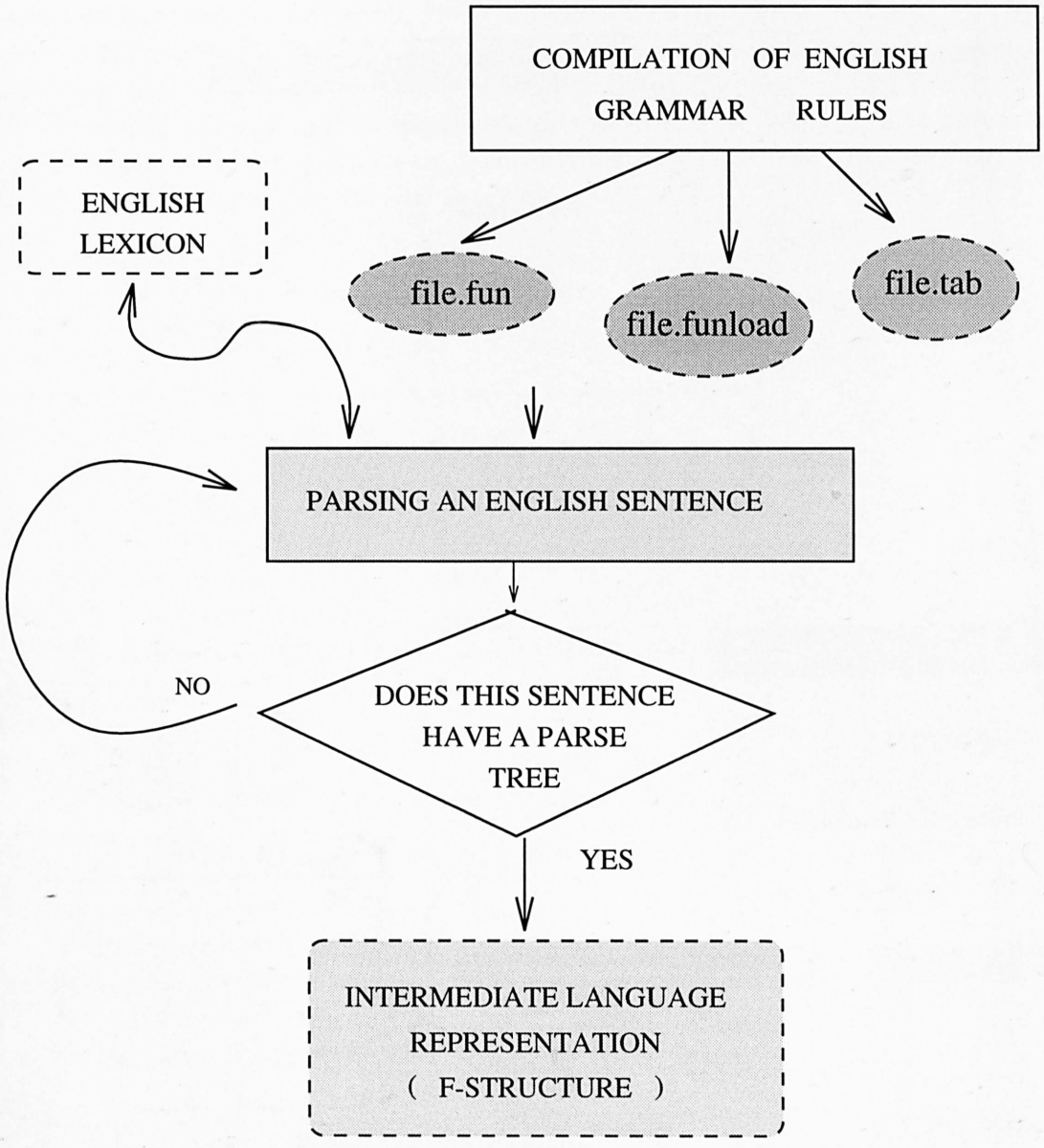


Figure 3.2: **The Analysis Phase**

used in defining rules, spaces are ignored in parsing. The above rule will parse as follows: "A mouse +s gratitude", "Amouse+sgratitude", "A m ou se +s grati tude" with more spaces in between, and so on. Non-terminal symbols must be enclosed in angle brackets. A grammar file has to have the extension ".gra", in this example it is called "title.gra". Now from the Lisp environment, the GLR system is loaded. In order to make the grammar ready for parsing, a compilation process takes place first; the compilation process starts by using the function (compgra "grammar-name") without the extension. Something like the following will appear during the compilation of the grammar.

```
<cl> (compgra"title")
 - Reading title.gra
 - title.gra read

************************************************
***** Start compiling title.gra
 - Reading title.gra
 - title.gra read
*** Grammar Pre-processor started
*** Grammar Pre-processor done
*** LFG Compiler started
*** LFG Compiler done
*** LR Table Compiler started
 - converting grammar
 - there were 8                    rules
 - there were 8 really different rules
 - there were 18 symbols
 - there were 12 terminal symbols
 - there were 6 non terminal symbols
 - making augmented grammar
 - making all items
 - 34 items made
 - collecting all items
LR [  0]
LR [  1]
LR [  2]
LR [  3]
LR [  4]
LR [  5]
LR [  6]
LR [  7]
```

```
LR [  8]
LR [  9]
LR [ 10]
LR [ 11]
LR [ 12]
LR [ 13]
LR [ 14]
LR [ 15]
LR [ 16]
LR [ 17]
LR [ 18]
LR [ 19]
LR [ 20]
 - the number of states is 26
 - generating parsing table
LR'[  0]
LR'[ 20]
 - reforming goto table
*** LR Table Compiler done
 - Writing File title.tab
 - File title.tab written
 - Writing File title.fun
 - File title.fun written
 - Writing File title.funload
 - File title.funload written
***** Setting up the runtime parser
  Loading File - title.tab
; Loading /export/home/ahmed/mt/ArabMT/title.tab.
  Loading File - title.fun
; Loading /export/home/ahmed/mt/ArabMT/title.fun.
  Parser Ready for title
<cl>
```

After the compilation, three files will be created for the parser (these files are named as title.tab, title.fun and title.funload), and the compiled grammar is loaded automatically. A phrase saying "Parser Ready for title" is displayed. To activate the parser, the command (p"sentence") is used and parsing of the sentence "A mouse +s gratitude" will be started. There are some debugging facilities needed some times to find out why a rule is killed, and when it should be fired [TMMK88], Below is the parsing sequence in a debugging mode:

```
<cl> (dmode 2)
<cl> (p"A mouse +s gratitude")
>A mouse +s gratitude
A
 rule #    7 TITLEF-28        <DET>(1) --> A
M
O
U
S
E
 rule #    6 TITLEF-24        <N>(7) --> MOUSE
+
S
 rule #    8 TITLEF-32        <POS>(10) --> +S
 rule #    3 TITLEF-12        <NP>(11) --> <DET>(1) <N>(7) <POS>(10)
G
R
A
T
I
T
U
D
E
 rule #    5 TITLEF-20        <N>(21) --> GRATITUDE
 rule #    4 TITLEF-16        <NP>(22) --> <N>(21)
 rule #    2 TITLEF-8         <SIMPLE-TITLE>(23) --> <NP>(11) <NP>(22)
 rule #    1 TITLEF-4         <SENTENCE>(24) --> <SIMPLE-TITLE>(23)

 1 (1) Interpretation found and took 4.240 seconds of real time
node # 24

;**** Interpretation 1 ***
((MOOD *NOMINATIVE)
 (SUBJECT
  ((DEFINITE ((NUMBER SG) (DEFINITE DEF) (ROOT A)))
   (NOUN
    ((AGREEMENT 3SG) (CAT NOUN) (GENDER FEMININE) (NUMBER SG) (ROOT MOUSE)))
   (SUBCAT TRANS) (NUMBER SG) (ROOT +S)))
 (ACTION
  ((SUBCAT TRANS) (CAT ADJ) (AGREEMENT 3SG) (NUMBER SG) (ROOT GRATITUDE))))
```

```
<cl>
```

The final interpretation is the F-structure for that sentence. In the case of an ambiguous input sentence, it will produce all possible structures. Only three top-level F-structures are produced by default. Now to see the syntactic tree, a function "disp-tree" is used, then the following syntactic tree is shown for that phrase.

```
<cl> (disp-tree)
<SENTENCE>(24) --> <SIMPLE-TITLE>(23)
    <SIMPLE-TITLE>(23) --> <NP>(11) <NP>(22)
        <NP>(11) --> <DET>(1) <N>(7) <POS>(10)
            <DET>(1) --> A
            <N>(7) --> M O U S E
            <POS>(10) --> + S
        <NP>(22) --> <N>(21)
            <N>(21) --> G R A T I T U D E
```

The numbers next to the nodes represent the rule position in the compiled LR table (i.e "title.tab"). In subsequent chapters, either form is used to show a parse.

3.5 System Domains

The previous section presented a brief overview of the GLR and GLR* approaches, describing the various phases involved. We are going to show the advantages of this system for three categories of NL sentences. The grammar rules used for parsing and generation, are not identical for the two languages. The differences between the grammar rules used for parsing are different from these used for generation have to do with the fact that expressions which are translations of each other need not have the same number of words. Three types of domain have been used for testing the system. First domain consists of natural language sentences taken from a children's story which has a variety of sentence types. Second, we look at more complex cases of the semantic problems in MT. The third domain, consists of general pragmatic sentences or phrases which have been covered in both languages. In the following section details of the first domain are given. The semantic sentence domain is discussed in chapter four, and the pragmatic sentence domain is given in chapter five.

3.6 The Children's Story

One of my daughter's stories is called "A mouse's Gratitude". This story was selected to test the system. See the selected part of the story in Appendix A (1). It contains many sentence and phrase types in NL. These types are: Declarative sentences, Wh-question (Interrogative and Surprising sentences), Adjective phrases, Conjunction, Compliment, Imperative (Polite form). See the types of the story's sentence written in LFG in Appendix A (2).
Since the GLR requires the *START* equations to be the first equation listed in the grammar, the parser always assumes that it will accept only whatever structure is specified in the left hand side of the first grammar rule. By making the left hand side of the first rule be the non-terminal <START>, and then defining <START> one or more times, the grammar can cause the parser to accept a customized set of structure types. It is important to write the mono-lingual components of the system with great independence of each other. See the following lexicon entries which have been used for the story's title "A mouse's gratitude":

```
(<N>  --> (g r a t i t u d e)
          (((X0 root) = gratitude)
           ((X0 number) = sg)
           ((X0 agreement) = 3sg)
           ((X0 cat) = adj)
           ((X0 subcat) = trans)
           (X0 = X1)))

(<N>  --> (m o u s e)
         (((X0 root) = mouse)
          ((X0 number) = sg)
          ((X0 gender) = both)
          ((X0 cat) = noun)
          ((X0 agreement) = 3sg)))
```

These entries will be unified according to the grammar rule conditions. If the conditions are satisfied then unification of both entries will give the final F-structure to the phrase. This may combine by forward rule for possessive case that is represented here by the symbol (+s). See the grammar rule which parses the above phrase:

```
(<SIMPLE-TITLE> <==> (<NP> <NP>)
                 (((X0 action) = X2)
                  ((X0 subject) = X1)))
```

The non-terminal symbol NP needs more description. Each of the NP will therefore be formed as follows:-

```
(<NP> <==> (<DET> <N> <POS>)
          (((X1 number) = (X2 number))
           ((X0 definite) = X1)
           ((X0 noun) = X2)
           (X0 = X3)))

(<NP> <==> (<N>)
          ((X0  = X1)))
```

Now the <N> in a straight forward way will refer to the lexical entry for that noun. Then the unification of the above LFG rules will be passed to <SIMPLE>. Sometimes there are more interpretations to the sentence, this is because it can satisfy more than one grammar rule. GLR produces interpretation of three F-structures for a sentence by default. The question is now how to select the best F-structure among this batch of F-structures? The answer to that question is connected to recent modification works by Lavie and Tomita [LT93], which includes a statistical disambiguation module in GLR*. To select the "best" parse from the set of parses returned by the parser, they used a scoring procedure that ranks each of the parses found, then selects the parse that was ranked best. Here in our system, the first F-structure list is going to be used and the rest ignored.[5]

[5]So far GLR* version has not distributed yet.

Chapter 4

The Arabic Computational Lexicon

4.1 Introduction

Reservations have been expressed as to whether the rule-based declarative approach is the most appropriate one for the description of Arabic morphology. Reference has been made to the possible use of Two-level models [Kos83], and Three-level models [NH92], on the one hand, and Functional Unification Grammars(FUG) [Kay85], Lexical Function Grammars [KB82], and Generalized Phrase Structure Grammars (GPSG) [GEGA85], on the other, to mention just a few of the alternative declarative and rule based approaches. Some research work has been done towards building an Arabic lexicon. Issues have included the combination of feature structures (unification), inclusion relations between them and the consequences that their use has for the nature of the parsing and generation processes. None of the previous research work has an Arabic computational lexicon based on feature structures, which have become increasingly important for any NLP or MT system. The feature-based system has permitted computational linguists to adopt very simple and compact grammatical rule systems at the cost of pushing almost all of the syntactic facts about the language into the lexicon [GEGA85]. Morphological rules become a matter of crucial concern, because there is no need to write a separate lexicon entry for each noun form. Typing every lexical rule by hand is a time consuming task and may allow unnecessary bugs to occur in the rules, in addition to the fact it reduces the size of the grammar. For simplicity, the system is divided into four modules as follows:

1. Noun Inflections Module.
2. Verb Inflections Module.
3. Irregular verbs such as the Verb "to be" and Verb "Not to be".
4. Pronouns and Miscellaneous Module.

4.2 The Lexical Functional Grammar Rules for Arabic words

The literal Arabic morphological system was well studied by Arab grammarians many centuries ago. The Arabic language has three types of word roots, namely, *tri-radical, quadri-radical and quinque-radical.* Verbs, nouns and adjectives are derived from the root, following regular patterns. These patterns are formed generally by a combination of the following changes:

- Inserting specific patterns of vowels,
- Prefixing,
- Suffixing,
- Adding medial consonant.

None, one or more of the above combination may be required to form an Arabic word (see Figure 4.1). Verbs are derived from tri and quadri-radical roots only, while nouns are also found with a quinque-radical root. Tri-radical roots are most frequently used in Arabic and form more than 63% of Arabic roots [Mou88]. Each pattern usually implies something about its meaning and its grammatical function.

Word formation in Arabic involves three concepts: the concept of root, the concept of pattern and the concept of form. Although Arabic is poor in verb tenses, it is rich in derived verb forms which extend or modify the meaning of the root form of the verb, giving many exact shades of meaning. This is a common feature of Semitic languages. Affixation of a word can be handled by writing Context-Free Phrase Structure Rules. A lexicon can be defined as a set of lexical entries. This lexicon requires only a set of root forms, and the morphological rules which will define all the possible affixations.

Taking advantage of the Universal Parser feature[NCTG92], the morphological rules can be written in the same formalism as the syntactic and semantic rules. For instance, the pattern of a given noun is parsed by the lexical rule for the root definition, and then the equation portion for

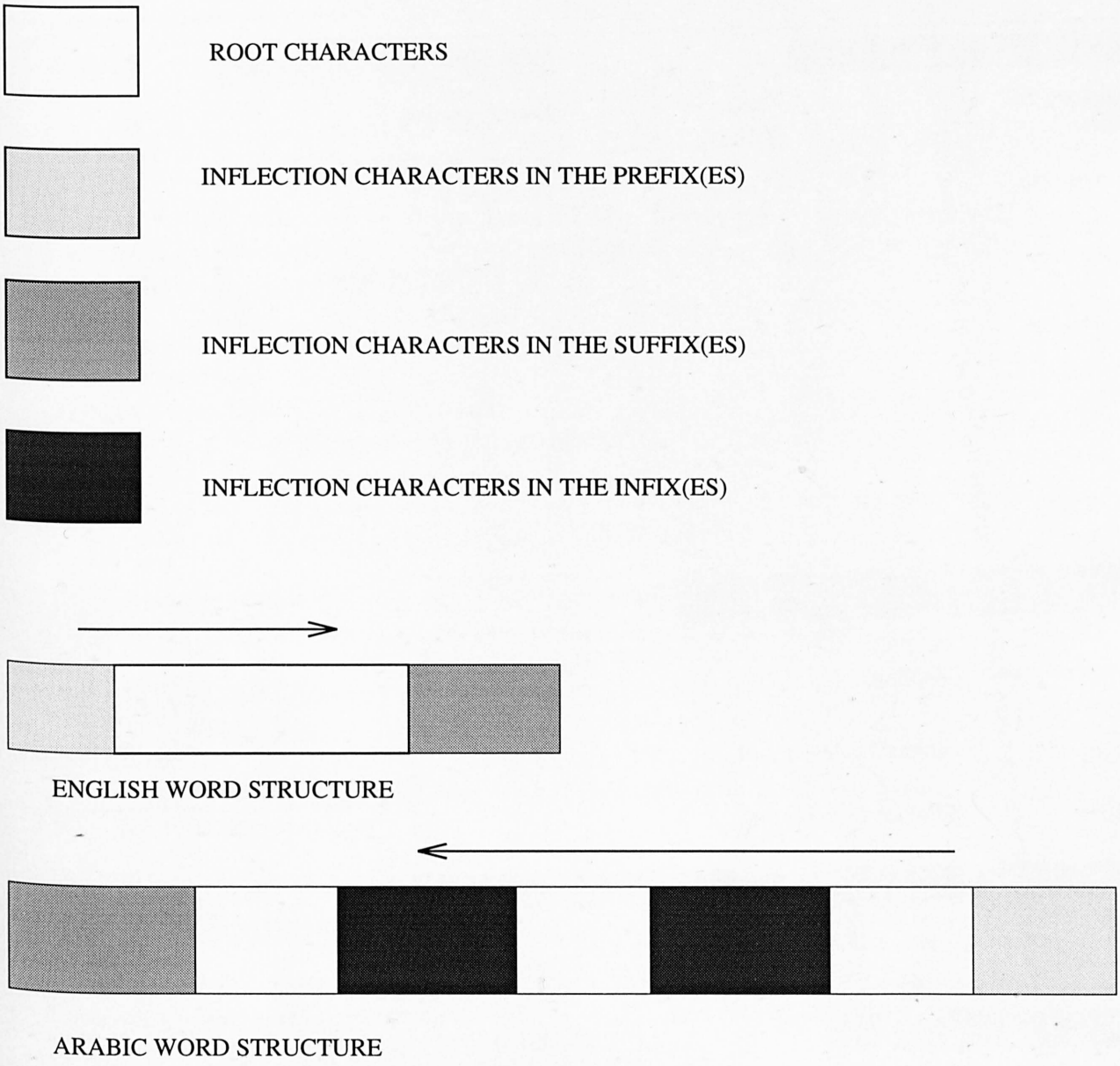


Figure 4.1: **The Word Structure**

any affixation rule will assign more features to the noun. For example, the noun مُعَلِّمٌ *muʿallimun* which means a teacher, will have the following LFG equations:

```
;Lexical rule:
(<NROOT>  --> (m u a l l i m)
              (((X0 root) = muallim)
               ((X0 count) = yes)
               ((X0 definite = yes)))
```

The above rule will be the main part of the F-structure later, and will be used together with any new morphological rule. The following rule derives the plural and the feminine noun.

```
;Deriving the plural feminine form:
;Case Nominative
(<NROOT>  --> (<NROOT> a a T u n)
               (((X0 pl-form) = aaTun)
                ((X0 a-an) =  al-)
                ((X0 nom) = + )
                ((X0 gender) = feminine)))
```

A set of morphological rules can be created depending on the above root rule. One of the generated patterns of that noun is as follows:

```
<cl> (p"muallim aaTun")

>muallim aaTun
 1 (1) interpretation found and took 1.444 seconds of real time
;**** interpretation 1 ***

((GENDER FEMININE) (NOM +) (A-AN AL-) (PL-FORM AATUN) (DEFINITE YES)
(COUNT YES) (ROOT MUALLIM))
```

The above F-structure is generated from the root noun form using the above morphological rule. Many feature equations can be used to prevent a rule from being applied in an undesirable order.

4.3 Nouns Inflections Module

In considering the design of this module, a number of issues had to be addressed. There are two kinds of nouns: "solid nouns", which are not

derived from verbs (such as أَسَدٌ *'asadun*: lion), and "unsolid nouns", which are derived from verbs (e.g. the noun مَكتُوبٌ *maktuwbun* :a letter[1] is derived from the verb كَتَبَ *kataba* (write). Each noun in Arabic is formulated depending on a number of the following factors:

- Declension of the noun (Cases). There are three cases indicated merely by changing the vowel after the final consonant (except in the dual and sound masculine plural endings). The special sound "uN" which is called (Tanwin) occurs after the final vowel in all three cases. The cases are:

  1. Nominative (NOM), vowelled with "damma" mark.
  2. Accusative (ACC), vowelled with "fatha" mark.
  3. Genitive (GEN), vowelled with "kasra" mark.

  The nominative case is used as the subject for a sentence, the accusative indefinite as "state" or "condition", and the genitive is used for "possession" or after a preposition.

- Number, which is Singular, Dual or Plural.

- Gender, which is Feminine or Masculine.

- Definite/Indefinite articles. To give a noun a definite status in Arabic the symbol (أَل *'al*) is added to the noun (prefix). By adding this, it will effect the vowelization of the end letter. In the definite noun removing of the original end marks and changing them to a nominative noun, is called "TANWIN" (e.g. the word مُعَلِّمٌ *mu'allimun*: a teacher, but in the definite will be written as أَلْمُعَلِّمُ *'al-mu'allimu*).

However, in the noun module, to derive the plural nouns, a set of singular nouns must be added with either a countable flag *:YES* or a flag *:NO* for the uncountable nouns. We assume the root nouns are *masculine gender* form in the beginning. Each word will have a set of default features in LFG. The discussion is focused on the following sections:

1. Regular Plural Masculine Noun Derivation.
2. Dual Masculine Noun Derivation.

[1]A letter is one of the meaning to the word.

3. Singular Feminine Noun Derivation.
4. Dual Feminine Noun Derivation.
5. Regular Plural Feminine Noun Derivation
6. Irregular Plural Noun Forms.
7. Noun for Both Singular and Plural.
8. Noun Inflection with Attached Pronouns.

These sub-modules are described in the following subsections.

4.3.1 Regular Plural Masculine Noun Derivation

Two more suffixes are needed: ونَ *wna*[2] for nominative type and إينَ *iyna* for oblique. Since the singular noun has its own ending vowel (tanwin) "uN", then modification needs to be done before attaching the suffixes. In order to show the correct plural suffixes and vowels, a strip function[3], is needed before applying the rule. The following is a example which comes from the noun مُعَلِّمٌ *mu'allimun*: teacher.

(مُعَلِّمُونَ *mu'allimuwna*) Plural-Nominative-Masculine (NOM)

(مُعَلِّمِينَ *mu'allimiyna*) Plural-Accusative-Masculine (ACC)
The following is a parsing interpretation for both nouns.[4]

```
<cl> (p"muallimuwna")
>muallimuwna

 1 (1) interpretation found and took 0.79 seconds of real time
;**** interpretation 1 ***

((GENDER MASCULINE) (A-AN AL-) (COUNT YES) (NOM +) (DEFINITE YES)
(COUNT YES)(PL-FORM UWNA) (ROOT MUALLIM))
```

And the parsing of the Accausative noun is:-

[2]The actual suffix is "wna", the LaTeX default whenever the Arabic vowels such as "a" or "u" are used in the beginning of the word, then interpretation is an Arabic "Alif" letter; but if used in a word it shows a perfect diacritic mark.

[3]This function removes the tanwin end mark from the word, i.e rmoving the "uN" from the root before applying the pattern "uwna"

[4]The letter ع *'a* is ignored in the coding, because it has a different meaning in Lisp.

```
<cl> (p"muallimiyna")
>muallimyina
 1 (1) interpretation found and took 0.815  seconds of real time
;**** interpretation 1 ***

((GENDER MASCULINE) (ACC +) (A-AN AL-) (COUNT YES) (DEFINITE YES)
(COUNT YES)(PL-FORM IYNA) (ROOT MUALLIM))
```

Those rules will not allow the noun to have unacceptable endings or vowels. An attempt to parse illegal noun inflections follows:

```
<cl> (p"muallimina")
>muallimina
 failed
 parsed --> (M U A L L I M)
 rest   --> (I N A $)

OR
<cl> (p"muallimema")

>muallimema
 failed
 parsed --> (M U A L L I M)
 rest   --> (E M A $)
```

Now let us see how the rule is failed by using the debug function when parsing an illegal plural noun form, as shown in the following example:

```
<cl> (p"muallimuwna iyna")
>muallimuwna iyna
M   -
U
A
L
E
M
 rule #   1 PL2F-4    <NROOT>(6) --> MUALLIM
U
W
N
A
 rule #   6 PL2F-24   <NROOT>(11) --> <NROOT> (6) UWNA
Y
```

```
I
N
A
 killed - rule # 7 PL2F-28  <NROOT> --> <NROOT>(11) I(13) Y(12)
                                        N(14) A(15)

 failed
 parsed --> (M U A L L I M U W N A I Y N A)
 rest   --> ($)
```

Here, the parser checks on a character by character basis, therefore, the system will reject any misspelled nouns or illegal vowelisation in the same lexical rules[5]. This would make the rules very useful in parsing sentences.

4.3.2 Dual Masculine Noun Derivation

For the dual masculine nouns, there are two suffixes used; first is انِ *āni*[6] in the nominative case and second اَينِ *ayni* in the other cases. The following are the lexical rules for this:

```
;RULE NO. 3 - dual-form nominative
(<NROOT>  --> (<NROOT>  a a n i)
                (((X0 dual-form) = aani)
                ((X0 count) = yes)
                ((X0 a-an) =  al-)
                ((X0 nom) = +)
                ((X0 gender) = masculine)))

;RULE NO. 4 - dual-form accusative
(<NROOT>  --> (<NROOT> a y n i)
                (((X0 dual-form) = ayni)
                 ((X0 count) = yes)
                 ((X0 a-an) =  al-)
                 ((X0 acc) = +)
                 ((X0 gender) = masculine)))
```

It is extremely difficult to derive a dual *feminine* noun from the singular masculine noun. This will include deriving the singular feminine noun first. But the next derivation will solve this problem.

[5]The parser can't stop at node no. 6, that is because there is some more input to parse and the pasrer has already checked node(6).

[6]The actual suffix being used is "aani".

4.3.3 Singular Feminine Noun Derivation

From the root form a singular feminine noun can be derived by adding the suffix (ة -*t*) which is called "ة المَربُوطَة -*t almarbuwuṭata*" will make the noun feminine. This derivation has to be in three case endings, as follows:

مُعَلِّمَةٌ *mu'allimatun* Singular -Feminine NOM. (Teacher (female))

مُعَلِّمَةٍ *mu'allimatin* Singular - Feminine GEN. (Teacher (female))

مُعَلِّمَةً *mu'allimatan* Singular - Feminine ACC. (Teacher (female))

```
;RULE NO 7- Singular feminine noun   has three cases.
; -Case Nominative.
 (<NROOT>  --> (<NROOT> a T u N)
               (((X0 pl-form) = aTuN)
                ((X0 count) = yes)
                ((X0 a-an) =  al-)
                ((X0 nom) = + )
                ((X0 gender) = feminine)))

; -Case Accusative.
 (<NROOT>  --> (<NROOT> a T i N)
               (((X0 pl-form) = aTiN)
                ((X0 count) = yes)
                ((X0 a-an) =  al-)
                ((X0 acc) = - )
                ((X0 gender) = feminine)))

; -Case Genitive
 (<NROOT>  --> (<NROOT> a T a N)
               (((X0 pl-form) = aTaN)
                ((X0 count) = yes)
                ((X0 a-an) =  al-)
                ((X0 gen) = + )
                ((X0 gender) = feminine)))
```

4.3.4 Dual Feminine Noun Derivation

Now, to derive the dual feminine nouns, a unification of both rules are needed. The actual singular noun rule without the tanwin vowel first, then the dual feminine rule second. This rule will add the suffix تَانِ *tāni* in the NOM case and تَينِ *tayni* in the other cases. Then it will be unified

in one F-structure. The following rule is used for the nominative case ending:

```
;RULE NO. 8 - Derive the dual form for the feminine noun
 (<NROOT> --> (<NROOT> t a a n i)
              (((X0 dual-form) = taani)
               ((X0 count) = yes)
              ((X0 a-an) =  al-)
              ((X0 nom) = + )
              ((X0 gender) = feminine)))
```

4.3.5 Regular Plural Feminine Noun Derivation

This form will have the suffix اتٌ *-ātun*, to the root noun for three case endings. For a complete list of inflections of the noun module see, Table 1, Appendix B.

4.3.6 Irregular Plural Noun Forms

For some nouns, the above lexical rules cannot be used, because of the existence of irregular plural form nouns called *BROKEN PLURALS* in Arabic. These nouns must be dealt with by special grammar rules. Since a flag countable "YES" is used in the beginning, now by setting that flag to "NO" it will not generate the plural form of that noun and will use the supplied plural form instead. But after that it can use all the other lexical rules in the same way as mentioned before. (e.g. مَطَر *maṭar* :rain, أَمطَار *ʾamṭa-ʾr* :rains). Both the noun and its plural form are needed in the definition. See the folowing parse:

```
<cl> (p"matar")
>matar
 1 (1) interpretation found and took 0.596 seconds of real time
;**** interpretation 1 ***

((AGR 3SG) (NUM ((ROOT SINGULAR)))(ROOT MATAR))

<cl> (p"amtar")
>amtar
 1 (1) interpretation found and took 0.620 seconds of real time
;**** interpretation 1 ***

((AGR 3SG) (NUM ((ROOT PLURAL))) (ROOT MATAR))
```

For a full derivation of the irregular noun plural, see the noun module, Table 2, Appendix B.

4.3.7 Noun for Both Singular and Plural

There are some Arabic nouns that can be used as singular and plural in the same way as some English words, e.g. water[7] مَاءٌ *mā'un*, air هَوَاءٌ *hawā'un*. In this case, different lexical rules will allow that form to be applied and eventually will accept all other inflections.

```
<cl> (p"maa")
>maa
 1 (1) interpretation found and took 0.593  seconds of real time
;**** interpretation 1 ***

((AGR 3SG) (PL-FORM SAME) (COUNT NO) (ROOT MAA))
```

For a full derivation list, see the noun module, Table 3, Appendix B.

4.3.8 Noun Inflection with Attached Pronouns

There are two types of pronoun in Arabic "Detached and the Attached pronouns". The detached pronouns will be discussed in the next module, but now some lexical rules of attached pronouns are needed to retain the correct derivation with the suitable F-structure for attached pronouns which combine with nouns to produce noun phrases.

```
;RULE NO. 12 - Pronoun for dual Masculine and Feminine second person.
(<NROOT>  --> (<NROOT> u k u m a a)
              (((X0 pro-form) = ukumaa)
               ((X0 person) = 2)
               ((X0 agr ) = 3sg)
               ((X0 case) = nom)
               (*or* (((X0 num root) = plural))
                  (((X0 num root) = dual)))
                   ((X0 human) = +)
                  ((X0 gender) =  both)))
```

[7]Sometimes the word water has a countable form in English when saying "Territorial waters".

Notice, the above rule can generate two interpretations having two parse trees when it parses a noun. The example below shows two interpretations for مُعَلِّمُكُمَا *mu'allimukumā* :your teacher, one for the dual and the other for the plural.

```
 <cl> (p" muallimukumaa")
> muallimukumaa
 2 (2) interpretations found and took 1.768 seconds of real time
;**** interpretation 1 ***

((GENDER BOTH) (HUMAN +) (NUM ((ROOT DUAL))) (CASE NOM)
 (AGR 3SG) (PERSON 2)(PRO-FORM UKUMAA) (ROOT MUALLIMUN))

;**** interpretation 2 ***

((GENDER BOTH) (HUMAN +) (NUM ((ROOT PLURAL))) (CASE NOM) (AGR 3SG)
 (PERSON 2) (PRO-FORM UKUMAA) (ROOT MUALLIMUN))
```

Finally, the noun inflections for the attached pronoun require about fourteen LFG rules that cover all possible declension types. (See the Noun Module, Tables 4, 5, 6 and 7, Appendix B). It is worth mentioning that most of the above nouns with attached pronoun rules are going to be used to attach them to some acceptable verbs inflections as well; more details in next section. This of course will not force us to have redundant rules in the system.

4.4 Verb Inflections Module

The morphological rules which will define all the possible verb affixations are obtained by applying one of the patterns generally numbered by Europeans from No. II upwards, while no. I is used for the root form. The exact number of derived forms is open to dispute as this number can be increased. For the verb derivation, however, nine of the derived forms are in common use from II to X, the remaining forms are rare [HN62]. (See the verb module, Table 1, Appendix C).<—

There is no single verb root which has a base form for all the derived stems. Some only give one or two forms. There is often a good deal of overlapping of meaning between the forms.

The system can accept the root features and pass them to other rules for possible inflections and generating a new parse tree. Very long verbs can be generated from the root, then the verb can have acceptable affixes

and attached pronouns, provided that there are rules for it. See the example below of morphological and lexical rules for general Arabic verb root type (I):

```
;1-lexical rule for the stem verb Type (I) MASCULINE.
(<STEM> --> (<r o o t>)
              (((X0 root) = root)
               ((X0 tense) = past)
               ((X0 person) = sing)
               ((X0 number) = 1)
               ((X0 gender) = masculine)))

;10-The morphological rules for the perfect verb FEMININE
; type IV (Af'alat).
(<STEM> --> (a  <STEM> t)
              (((X1 tense) = perfect)
               ((X1 person) = sing)
               ((X1 number) = 1)
               ((X0 gender) = feminine)))
```

Suppose the verb (كَسَرَ *kasara*) which means "break or smash" is taken as an example, then the above morphological rules can parse the following verb form: (أَكسَرَت *'aksarat*).

```
[1] <cl> (p"aksarat")
>a kasara t
A
K
A   -
S
A
R
 rule #   9 VERBSF-36       <STEM>(6) --> KASAR
A
 rule #  10 VERBSF-40       <STEM>(8) --> <STEM>(6) A
 rule #  18 VERBSF-72       <STEM>(9) --> A<STEM>(6) A
T
 killed - rule # 14 VERBSF-56  <STEM> --> <STEM>(9) T(12)
 killed - rule # 14 VERBSF-56  <STEM> --> <STEM>(8) T(11)
 rule #  20 VERBSF-80       <STEM>(13) --> A<STEM>(8) T
 rule #  73 VERBSF-319      <STEM>(14) --> <STEM>(6) AT
 rule #  59 VERBSF-236      <STEM>(15) --> <STEM>(6) AT
```

```
 1 (1) Interpretation found and took 3.801 seconds of real time
node # 13
;**** Interpretation 1 ***

((GENDER FEMININE)(NUMBER 1)(PERSON SING)(TENSE PERFECT)(ROOT KASAR))
```

The parser start parsing the stem first letter by letter. It checks if it satisfy any rule, if not then stop. The parsing process will parse the diacritical marks then the pronoun (Rule #8). If the parser find that some rules can't be applied for such a string then the parser (killed) their effect. The parser makes the necessary unification then accumulate them in a feature structure list.

Notice that in the last F-structure, the gender in type IV has overruled the gender in type I ; and the same has been done for the tense. Due to the complexity and diversity of the Arabic verb forms, this module derives the forms in three groups as follows:

1. Group 1: Forms II, III, V, VII, IX, X.
2. Group 2: The Passive Pattern.
3. Group 3: Deriving the Attached Pronouns to the Verb.

4.4.1 Group 1: Forms II, III, V, VII, IX, X

It will help to consider these patterns as one group. To derive these inflections, the system starts with the root form (I). The pattern used for deriving these forms which was named by early Arab grammarians, is the tri-radical verb template (فَعَلَ *fa‘ala*). One verb form will be used for the demonstration, and the rest of the inflections can be found in the verb module, Appendix B. Each case will have two tenses one for the perfect and the second for the imperfect and each will have two types, one for masculine and one for feminine. For each imperfect verb three case endings such as *Indicative, Subjunctive and Jussive* are defined. Detail of each form appear in the following sections.

Form (II): pattern (فَعَّلَ *fa‘‘ala* يُفَعِّلُ *yufa‘‘ilu*).

(1)- The Perfect Masculine: Pattern (فَعَّلَ *fa‘‘ala*). This pattern shows the masculine, 3rd person, singular form. Following is the standard lexical rule which is necessary to derive this pattern:

```
;1-Standard lexical rule for the stem verb perfect type (II) MASC.
(<STEM> --> (<STEM> a)
              (((X0 tense) = perfect)
               ((X0 person) = 3)
               ((X0 number) = sing)
               ((X0 gender) = masculine)))
```

Parsing the perfect feminine verb kassara: which means "broke":-

```
<cl> (p"kassara")
>kasara
 1 (1) Interpretation found and took 1.593 seconds of real time
node # 8

;**** Interpretation 1 ***
((GENDER MASCULINE) (NUMBER 3) (PERSON SING) (TENSE PERFECT) (ROOT KASAR))
```

The verb stem is (كَسَرَ *kasara*) and the derived form type (II) is (كَسَّرَ *kassara*)

(2)- The Perfect Feminine : pattern (فَعَلَت *fa'alat*).
This form is obtained by adding the suffix (ت *t*) to the stem. This shows the perfect feminine, 3rd person, singular as in the following parse of the verb (كَسَّرَت *kassarat*):

```
<cl> (p"kassarat")
>kassarta

 1 (1) interpretation found and took 3.229 seconds of real time
;**** interpretation 1 ***

((GENDER FEMININE) (NUMBER SING) (PERSON 3)(TENSE PERFECT) (ROOT KASAR))
```

(3)- The Imperfect Masculine: Pattern (يُفَعِّلُ *yufa''ilu*).
There are three types of case endings for the imperfect masculine as in the following rules:

```
;Case (1) Imperfect- Indicative
(<STEM> --> (y u <STEM> u)
              (((X0 tense) = imperf)
               ((X0 present) = indic)
               ((X0 subcat) = trans)
```

```
                ((X0 person) = 3sg)
                ((X0 gender) = masculine)))

;Case (2) Imperfect - Subjunctive
(<STEM> --> (y u <STEM> a)
               (((X0 tense) = imperf)
                ((X0 present) = subj)
                ((X0 subcat) = trans)
                ((X0 person) = 3)
                ((X0 gender) = masculine)))

;Case (3) Imperfect - Jussive
(<STEM> --> (y u <STEM> o)
               (((X0 tense) = imperf)
                ((X0 present) = jussive)
                ((X0 subcat) = trans)
                ((X0 person) = 3)
                ((X0 gender) = masculine)))
 The Parse of one case is following:
<cl> (p"yukassiru")
>yukassiru
 1 (1) interpretation found and took 0.938 seconds of real time
;**** interpretation 1 ***

((GENDER MASCULINE)(PERSON 3)(SUBCAT TRANS)(PRESENT INDIC)(TENSE IMPERF))
```

The above LFG rules derive the verbs in the following forms:
(e.g. يُكَسِّرُ *yukassiru* : Indicative, يُكَسِّرَ *yukassira*: Subjunctive, يُكَسِّر *yukassir* : Jussive)

(4)- The Imperfect Feminine: pattern (تُفَعِّل *tufa‘‘il.*)
Again this pattern will have three case endings. Only the inflected verb form will be given.

Case 1 Imperfect - Indicative (e.g. تُكَسِّرُ *tukassiru*).

Case 2 Imperfect - Subjunctive (e.g. تُكَسِّرَ) *(tukassira*).

Case 3 Imperfect - Jussive (e.g. تُكَسِّر *tukassir*).

Form (IV) : (أَفعَلَ *›af‹ala*, يُفعِلُ *yuf‹ilu*).

Two verb tenses can be derived from this form and in two gender cases.
(1)- The Perfect Masculine type (IV): Pattern (أَفعَلَ *›af‹ala*).
This form is causative: it makes intransitive verbs transitive, and transitive verbs doubly transitive. e.g. verb root (جَلَسَ *ğalasa*) : "to sit"; the derived form: (أَجلَسَ *›ağlasa*): means to seat. Often form type II and IV can have the same verb meaning, with perhaps a slight difference, e.g. verb root (صَلَحَ *ṣalaḥa*): "to repair"; The derived form: (أَصلَحَ *›aṣlaḥa*): to rectify or reform), and in transitive verbs of this form. e.g. verb root (سَلَّمَ *sallama*): say hello, the derived form is : (أَسلَمَ *›aslama*): to become a Muslim, which has a completely different meaning. See the parsing of the verb (أَجلَسَ *›ağlasa*) as follows:

```
<cl> (p"aglasa")
>aglasa
 1 (1) interpretation found and took 2.55 seconds of real time
;**** interpretation 1 ***

((GENDER MASCULINE)(NUMBER SING)(PERSON 1)(TENSE PERFECT)(ROOT GALASA))
```

(2)- The Perfect Feminine type (IV): Pattern (أَفعَلَت *›af‹alat*).
This form is similar to the perfect masculine except that the gender requires the suffix (تَ *ta*)[8] (e.g. أَكسَرَت *›aksarat*). (It is important to mention that if the root verb has an 'alif as the first consonant (e.g. أَكَلَ *›akala*), then it must not derive it as أأَكَل *››akal* for example, which is not an acceptable form. Therefore a special procedure is needed that will allow only the form أَكَلَ *›akala* to be generated and parsed).

(3)- The Imperfect Masculine type IV: : Pattern (يُفعِلُ *yuf‹ilu*).

This form derives three case endings as follows:
Case 1 Imperfect - Indicative (e.g. يُكسِرُ *yuksiru*).
Case 2 Imperfect - Subjunctive (e.g. يُكسِرَ *yuksira*).

[8] Because of LaTeX default, the letter "a" is shown as "Alif" letter in Arabic script, but if used in a word it shows a diacritic mark instead.

Case 3 Imperfect - Jussive (e.g. يُكسِر *yuksir*).

(4)- The Imperfect Feminine type IV: pattern (تُفعِلَ *tuf'ila*).

Three case endings can be derived as follows:
Case 1 Imperfect - Indicative (e.g. تُكسِرُ *tuksiru*).
Case 2 Imperfect - Subjunctive (e.g. تُكسِرَ *tuksira*).
Case 3 Imperfect - Jussive (e.g. تُكسِر *tuksir*).

Form V: Pattern (تَفَعَّلَ *tafa''ala*, يَتَفَعَّلُ *yatafa''alu*).

(1)- The Perfect Masculine : Pattern (تَفَعَّلَ *tafa''ala*).

Derived forms type V tend to be reflexives of form II, from which they are formed by prefixing (تَ *ta*). The meaning of this form indicates that something has acted on itself, i.e. the verb is reflexive, e.g. verb root (كَسَرَ *kasara*) : break, gives the form (تَكَسَّرَ *takassara*) :to break by itself.
(2)- The Perfect Feminine : Pattern (تَفَعَّلَت *tafa''alat*).

This form can be derived by adding the suffix (تَ *ta*) and prefix (تَ *ta*) to the root, e.g. (تَكَسَّرَت *takassarat*), and the F-structure when parsed is:

```
<cl> (p"takassarat")
>takassarat
 1 (1) interpretation found and took 1.550 seconds of real time
;***** interpretation 1 ***

((GENDER FEMININE) (PERSON 3SG) (SUBCAT TRANS) (PRESENT INDICATIVE)
 (TENSE IMPERF) (ROOT KASAR))
```

(3)- The Imperfect Masculine : Pattern (يَتَفَعَّلُ *yatafa''alu*).
Three case endings are derived which are:
يَتَكَسَّرُ *yatakassaru*, يَتَكَسَّرَ *yatakassara*, يَتَكَسَّر *yatakassar*

(4)- The Imperfect Feminine : Pattern (تَتَفَعَّلُ *tatafa''alu*).
Three case endings are derived which are:
تَتَكَسَّرُ *tatakassaru*, تَتَكَسَّرَ *tatakassara*, تَتَكَسَّر *tatakassar*

Form VII : Pattern (إنفَعَلَ *infa‹ala*, يَنفَعِلُ *yanfa‹ilu*)

(1)- The Perfect Masculine : Pattern (إنفَعَلَ *infa‹ala*)

Originally the reflexive of the root, this is form by adding the prefix إ *ī* to the root. e.g. verb root (كَسَرَ *kasara* :to break; the derived form is (إنكَسَرَ *inkasara*): to break (intransitive). See the following parsed inflection.

```
<cl> (p"inkasara")
>inkasara
 1 (1) interpretation found and took 1.155 seconds of real time
;**** interpretation 1 ***

((GENDER MASCULINE)(NUMBER 1)(PERSON SING)(TRANS INTRANS)(TENSE PERFECT))
```

(2)- The Perfect Feminine : Pattern (إنفَعَلَت *infa‹alat*).

This form has the prefix إن *ā'n* and the suffix ت *t*, e.g. verb root (كَسَرَ *kasara*): to break; the derived form is (إنكَسَرَت *inkasarat*) : to break by its self.

(3)- The Imperfect Masculine : Pattern (يَنفَعِلُ *yanfa‹ilu*).

This type has only the prefix يَن *yan* in the beginning of the root. There are three case endings generated by this verb:

e.g. يَنكَسِرُ *yankasiru*, يَنكَسِرَ *yankasira*, يَنكَسِر *yankasir*

(4)- The Imperfect Feminine : Pattern (تَنفَعِلُ *tanfa‹ilu*).

Again, the prefix is تَن *tan* for deriving the three case endings words.

e.g. تَنكَسِرُ *tankasiru*, تَنكَسِرَ *tankasira*, تَنكَسِر *tankasir*.

Form IX : Pattern (إفعَلَّ *ā'f‹alla*, يَفعَلُّ *yaf‹allu*).

(1)- The Perfect Masculine : Pattern (إفعَلَّ *if‹alla*).

This type seems similar to type IV (أَفعَلَ *'af‹ala*, يُفعَلُّ *yuf‹allu*), but is characterised by a prefixed ('alf) with (إ *ī*) and a doubling of the final letter. In certain parts, however, the doubled letter is written as two separate letters. This form is only used for colours and defects, and therefore a corresponding adjective will also be found (إفعَلَّ *ā'f‹alla*) e.g. (أَحمَر *aḥmar*): the red colour; but the derived form (إحمَرَّ *ā'ḥmarra*): to be or become red (or reddish) etc.)

(2)- The Perfect Feminine: Pattern (إِفعَلَّت *if'allat*).
The same as the above pattern but with the suffix (ت *t*).

(3)- The Imperfect Masculine : Pattern (يُفعَلُّ *yaf'allu*).
This pattern have three case endings as in the example:
يُحمَرُّ *yaḥmarru* يُحمَرَّ *yaḥmarra* يُحمَرِّ *yaḥmarir*.

(4)- The Imperfect Feminine : Pattern (تُفعَلُّ *taf'allu*).
This pattern have three case endings as in the example:
تُحمَرُّ *taḥmarru* تُحمَرَّ *taḥmarra* تُحمَرِّ *taḥmarir*. Notice the double letter for the Jussive form.

Form X : Pattern (إِستَفعَلَ *istaf'ala* يَستَفعِلُ *yastaf'ilu*).

This form is particularly rich in various extensions of meaning from the root, which cannot be classified [HN62], e.g. the verb root (قَبَلَ *qabala*): to receive, accept; but the form (إِستَقبَلَ *istaqbala*) : to welcome or receive a person. Two tenses are derived having different genders.
(For a full conjugation of group 1 see the verb module, Table 2, Appendix C).

4.4.2 Group 2 : The Passive Pattern

There is a special group, used to generate a new and widely used group of verbs. It can be used with all the previous verbs and their patterns as well. These passive participles are easily constructed by using all patterns prefixed with (م *m*) and vowelled with damma (مُ *mu-*). See the following table.-

Form Type	The perfect	The Participle
II	فَعَّلَ *fa''ala*	مُفَعِّلٌ *mufa''alun*
IV	أَفعَلَ *'af'ala*	مُفعِلٌ *muf'alun*
V	تَفَعَّلَ *tafa''ala*	مُتَفَعِّلٌ *mutafa''alun*
VII	إِنفَعَلَ *infa'ala*	مُنفَعِلٌ *munfa'alun*
IX	إِفعَلَّ *if'alla*	
X	إِستَفعَلَ *'istaf'ala*	مُستَفعِلٌ *mustaf'alun*

As we can see, form types II and IV have the same roots but different diacritics. The following table shows an example verb كَسَرَ *kasara*: break, with all the pattern forms.

II	مُكَسِّرٌ *mukassarun*
IV	مُكسِرٌ *muksarun*
V	مُتَكَسِّرٌ *mutakassarun*
VII	مُنكَسِرٌ *munkasarun*
X	مُستَكسِرٌ *mustaksarun*

4.4.3 Group 3: Deriving the Attached Pronouns to the Verb

The attached pronouns for a verb are classified according to the tenses. Type (I) contains the perfect stem and the imperfect with the three case endings. The attached pronouns are expressed either by *prefixes* or *suffixes* according to their number, person, and gender.

The Perfect Stem Attached Pronouns

The perfect stem, as we have seen, is obtained by cutting the last vowel of the 3rd person singular masculine perfect, and the perfect is conjugated by adding to the stem. (See the Verb Module, Table 2 which shows that ending). There are thirteen LFG rules used to generate the above cases. The following example will show the lexical rule for one of them.

```
;Rule12, 2nd-Feminine Plural Pronoun (tunna).
```

```
;e.g. You (women) have  written. (katabatunna)
(<STEM>  --> (<STEM> a t u n n a)
               (((X0 tense) = perfect)
                ((X0 person) = 2)
                ((X0 number) = plural)
                ((X0 gender) = feminine)))
```

(See the Verb Module, Tables 3, Appendix C).

The Imperfect Stem with Attached Pronouns

In this section, we discuss three case endings to the word as follows:

1. The Imperfect Indicative. Whereas in the perfect, the different persons were expressed by suffixes, the imperfect has prefixes. It also has some suffixes to denote number and gender. The prefixes and suffixes are shown in the verb module, Table 4, Appendix C).
2. The Imperfect Subjunctive.
   (See the conjugation in the verb module, Table 5, Appendix C).
3. The Imperfect Jussive.
   The Jussive Mood has the same forms as the Subjunctive except that where the quinque-radical is the last letter, it takes "Jazma (Sukun)". (See the conjugation in the verb module, Table 6, Appendix C).
4. The Imperative.
   The Imperative is formed from the Jussive, of which it may be considered a modification, by taking away the pronominal prefix, and replacing it by an ا *ā*. The parts of the Imperative, naturally all second person, are as follows:

   Sing. masc. أُكْتُب *ʾuktub*

   Sing. fem. أُكْتُبِ *ʾuktubi*

   Dual masc. and fem. أُكْتُبَا *ʾuktubā*

   Plur. masc. أُكْتُبُوَا *ʾuktubuwā*

   Plur. fem. أُكْتُبْنَ *ʾuktubna*

A duplication of the grammar rule is not used to show the gender (for example one for the feminine and other for the masculine), because the system computes both cases and gives two parse trees to show that.

4.5 Irregular Verbs such as Verb "To Be" and Verb "Not To Be"

The designed module is capable of dealing with the Arabic verb "to be" (كَانَ *kāna*) and the verb "Not to be" (لَيسَ *laysa*). These verbs play an important role in the usage of the language. See the example below of morphological and lexical rules for the Arabic verb (كَانَ *kāna*)

```
;;General Morphological Rules for the Verb (kana).
;3-per-MASC  Singular (e.g. kana) which means: He was.
(<IRR-STEM> --> (<IRR-STEM> a)
                (((X0 tense) = perfect)
                 ((X0 person) = 3)
                 ((X0 number) = sing)
                 ((X0 gender) = masculine)))
```

From the above morphological rule, the other verb inflections can be derived. This module consists of two sub-modules, as follows:
(i) Verb "To Be" (كَانَ *kāna*),
(ii) Verb "Not To Be" (لَيسَ *laysa*,
Detail of each type is shown in the following sections.

4.5.1 Verb "To Be" (كَانَ *kāna*)

This module deals with the verb "to be" (كَانَ *kāna*). The morphological rules are used for only one stem verb "to be" (كَانَ *kāna*); the system will derive the remaining inflections of that type. There are three types of verb "to be" which are as follows:
(i) The perfect كَانَ *kāna*,
(ii) The imperfect يَكُوْنُ *yakuwunu*,
(iii) The imperative كُن *kun*.
(i)- The perfect verb (كَانَ *kāna*) is used with the perfect of another verb to make a full sentence, the subject being normally placed between the two verbs, e.g. (كَانَ مَازِنٌ قَد كَتَبَ *kāna māzinun qad kataba*) which means "Mazin

had written", the actual verb being "written" [HN62]. All the inflections to that verb are derived with the characteristic of each of them. See the full inflection forms in the Irregular Verbs Module, Table 1, Appendix D.

4.5.2 Verb "Not To Be" (لَيسَ *laysa*)

This module deals with the inflections of the Verb "Not To Be" (لَيسَ *laysa*). This verb will give the negation of the sentence. Only the perfect occurs, and when used it has the meaning of the imperfect. This verb has several inflections depending on the gender, number and person. See the following F-structure for the word (لَستُمَا *lastumā*):-

```
<cl> (p"lastumA")
>lastumA
 1 (1) interpretation found and took 0.990 seconds of real time
;**** interpretation 1 ***
((GENDER MASCULINE)(NUMBER 2)(PERSON DUAL)(TENSE PERFECT)(ROOT LAYSA))
```

The rest of verb inflections are found in Table 2, Appendix D).

4.6 The Pronouns and Miscellaneous Module

This module covers the following pronoun types, the Separated Personal Pronouns, the Relative Pronouns (اَلَّذِي *alladī*), the Demonstrative Pronouns. Detail of one type will be discussed in the following section.

4.6.1 The Personal Pronouns

The attachment of pronouns to the verb is specified by the lexical equation parameters. Here we will see how the separate personal pronouns are used in the sentence. Some of them can be used for the masculine and others for the feminine and some can not have a distinction between masculine and feminine forms. The LFG rule for one of them (e.g. نَحنُ *naḥnu*) which means "we" is given.

```
;;The Plural form "nahnu" which means "We".
 (<PRONOUN>  --> (n a h n u)
                 (((X0 root) = nahnu)
                  ((X0 gender) = both)
                  ((X0 cat) = noun)
```

```
((X0 person) =  1st)
((X0 form) =  personal)))
```

Other pronoun types will have their individual grammar rules that derive their inflection forms. There are several pronoun types that can be defined in the system, but only the common pronouns are chosen such as; the Demonstrative pronoun, the Separated personal and the Relative pronouns. See these pronoun types in Appendix E.

There are many particles that can be created in this lexicon, but details of one set of particles is defined. This set of particles is called "Anna and its sisters" in Arabic which are declined as nouns in the sentences. See Appendix E for more detail. [9]

4.7 Conclusion

Since the system is a character-based one, it allows for the generation or analysis of a complicated word that contains the tense, subject, object, and pronouns all at the same time, forming a complete sentence (e.g. the word سَنُلَاعِبُهُم *sa-nu-lāʿibuhum* means "We will play with them").

The noun module has 47 lexical rules that generate the necessary inflections, so for a small domain with 20 entries a total of 940 words with their internal representation (i.e. Syntactic, Morphological and Semantic) are derived[AM95].

The verb module is very important in generating and analyzing Arabic verbs from the Tri-radical verb pattern, which is widely used. Quadri-radical and Quinque-radical, verbs have not been dealt with, but it is possible to add them later. The idea of attaching more pronouns to the inflected verb is possible, (e.g. أَضرِبُهُ *ʾaḍribuhu*, means I (masculine) strike him), this will need more lexical rules to be added. There are three forms which are not dealt with so far in this module (i.e. type (III), type (VI) and form type (VIII) because they need more rules and functions to break up the stem into two stems for each form, by doing that the system could have more undesirable inflections. These forms will be the subject of future investigation.

The verb module contains 125 lexical rules that do the necessary inflections, so for a small domain with 10 entries a total of 1250 words with their internal representation are derived. The total number excludes forms used for feminine and masculine as they are covered by one rule.

[9] More details can be found in [AM96b].

The irregular verb module creates the lexical feature entries for many Arabic verbs, such as, the verb "to be" (كَانَ *kāna*) and the verb "Not to be" (لَيسَ *laysa*) inflections. All of them are covered by 58 LFG rules. (i.e. for only one verb, the system can parse and generate 58 derived forms). The pronouns module contains 48 LFG rules covering most of the pronoun forms. Also the system has covered some miscellanous particles which are widly used, like "Ann and its sisters".

The Arabic computational system[AM96a] can be extended to cover more aspects of the Arabic language, such as *The Conditional conjunctions (Likely and Unlikely), Prepositions and their Particles, and the Cardinal Numbers.* By using this approach, it is always possible to add more features and/or words, moving towards a complete Auto-Analyzer and Generator Lexicon for the Arabic language. A selection of options are provided in the lexicon menu, whenever the system is started. This module could be useful as a CAL tool for teaching Arabic words to a novice.

Chapter 5

Semantic Specification

5.1 Introduction

This chapter presents a basic grammar for Arabic that covers some linguistic phenomena. This approach is used for the Verbal sentence which is a Verb-Subject-Object (VSO) order in Arabic, but it is straightforward and can be used for other word orders by changing the generation grammar rules. It is designed to be flexible for the different possible word orders in the language. The designed model contains a set of semantic grammar rules governing the morphological inflection requirement and selection of Arabic word sense. The semantics somehow have to be presented during the translation process. However, this brings up an issue of modularity in the monolingual lexicons. If the monolingual lexicons are to be developed independently of each other, possible by monolingual speakers as seems desirable, then it makes sense to have semantic primitives corresponding to the rules of the natural language in question. There is one further key point involved with the semantic: before generation, the semantics of the terms have to be grounded. First, instantiate the variables[1] in the F-structure with unique non-variable "legal" semantic rules. Second, when the transfer rule puts these variables into correspondence, all that is necessary is to unify these variables into new Arabic F-structure.

[1]These semantic variables are either a part of the lexical entry written in LFG or can be added during the transfer phase.

5.2 The Semantic Rules of Arabic in LFG

The semantic features of the grammatical subject or object of the verb can usually determine how the verb is being used. As an example, for the Arabic verb "akala :eat", such constraints could be; the agent must have the marker animate and masculine, the object must have the marker edible. The transfer rules can map an English root into corresponding Arabic forms and unify them with other Arabic semantic conditions. To illustrate what happens to the word sense in the sentence, two sentences are examined carefully e.g. "The boy plays football", and "The boy plays the piano". All possible combinations of these sentences have been considered. Here, these sentences contain the verb "play" used in a different sense, each one depending on the object being used. Thus, two semantic signs are needed. We start with the following entries, which show the semantic values.

```
(<OBJ> --> (f o o t b a l l)
        (((X0 root) = football)
         ((X0 number) = sg)
         ((X0 cat) = noun)
         ((X0 instrument) = game)))

(<OBJ> --> (p i a n o)
        (((X0 root) = piano)
         ((X0 number) = sg)
         ((X0 cat) = noun)
         ((X0 instrument) = music)))
```

These lexical entries are used as a part of the English parser. Then, after compiling the English grammar rules for these sentences, and issue the parsing command, the following F-structure is produced for each of them.

```
<cl> (p"the boy plays football")
>the boy plays football
 1 (1) Interpretations found and took 13.441 seconds of real time
node # 45
;**** Interpretation 1 ***

((CASE *NOMINATIVE)
 (AGENT ((AGREEMENT 3SG) (GENDER MASCULINE) (HUMAN YES)
 (NUMBER SG) (ROOT BOY)))
 (PATIENT ((INSTRUMENT GAME) (NUMBER SG) (ROOT FOOTBALL)))
```

```
 (ACTION ((TENSE PRESENT) (DEFINITENESS DEF) (AGREEMENT 3SG)
 (ROOT PLAY))))

 <cl> (p"the boy plays the piano")
>the boy plays the piano
1 (1) Interpretation found and took 14.336 seconds of real time
node # 47
;**** Interpretation 1 ***

((CASE *NOMINATIVE)
 (AGENT ((AGREEMENT 3SG) (GENDER MASCULINE) (HUMAN YES)
 (NUMBER SG) (ROOT BOY)))
 (PATIENT ((INSTRUMENT MUSIC) (NUMBER SG) (ROOT PIANO)))
 (ACTION ((TENSE PRESENT) (DEFINITENESS DEF) (AGREEMENT 3SG)
 (ROOT PLAY)))
 (NUMBER SG) (DEFINITE DEF) (ROOT THE))
```

These interpretations look like each other, but they are different in the semantic variable "INSTRUMENT" which is constrained to "MUSIC" and "GAME". The actual verb (PLAY) never changes. The above F-structure, together with the monolingual syntactic information about the target language is enough to generate the correct meaning. As long as the object refers to a different meaning (i.e. GAME and MUSIC), this constraint is important in deciding which word is going to be selected in the Arabic language. It is left to the transfer rule to check the semantic values for the patient as follows:[2]

```
(<N-SEM> --> (% %)
          ((*EOR*
           (((X0 patient instrument) =c music)
            ((X1 value) = "al"))
           (((X0  patient instrument) =c game)
             ((X1 value) = "-")))
           ((x2 value) = (X0 patient Arabic)))))
```

The above rule means that if the semantic value for the INSTRUMENT is restricted to "MUSIC" then this allows the Arabic word to be preceded by the determinate (X1 value = "al-") (ال *a-*), in the nominative case ending, otherwise it selects the Arabic word "GAME" without the determinate value which is represented here as "-".

[2]The % symbol means a global search which satisfy the equation condition.

5.2.1 Handling the Word Sense

To generate an Arabic sentence, more semantic rules are needed to cover the subject, verb and the object. More semantic rules means more features, such as one value to indicate the feminine and one for the masculine subject (gender). These values are used to modify the verb in order to obtain a special feminine or masculine form.
Now, the verb sense depends on both semantic features like (AGENT HUMAN =C YES) and (PATIENT INSTRUMENT =C GAME or MUSIC) as shown by the following rules:

```
;;Rule for selecting the VERB-SENSE depending on the source structure
(<V-ROOT-SEM> --> (%)
              ((*EOR*
               (((X0 agent human) =c yes)
                (*OR*
                 (((X0  patient instrument) =c game)
                  ((X1 value) =  "layab"))
                 (((X0 patient instrument) =c music)
                  ((X1 value) = "azaf")))))))
```

This rule will check the Agent and the Patient, then it decides which Arabic verb is selected in the Arabic form. Let us see how the system distinguishes the semantic variables: In this system only the (VSO) word order is designed and implemented. To start with, see the following grammar rule:

```
(<SEMANTICS-ST> ==> (<VP-OUT> <NP-OUT> <OBJ-P>)
     -                ((X1 = X0)
                       (x2 = X0)
                       (x3 = X0)))

(<VP-OUT> --> (<PRE-V> <V-ROOT-SEM> <SUFF-V>)
                  ((X1 = X0)
                   (x2 = X0)
                   (x3 = X0)))

(<NP-OUT> --> (al <N-ROOT-OUT> <SUFF-N>)
                 ((x2 = X0)
                  (x3 = X0)))

;Rule for deciding the masculine and feminine prefix to the verb,
;according to the tense.
```

```
(<PRE-V> --> (%)
            ((*EOR*
             (((X0 action tense) =c past)
              ((X1 value) = "-"))
             (((X0  action tense) =c present)
                  (*case* (X0  agent gender)
                     (masculine  ((X1 value) = "ya-" ))
                     (feminine ((X1 value) = "ta-")))))))
```

In order to distinguish the other meaning of the verb in the sentence, the system needs to observe the semantic rules that check for the ((PATIENT INSTRUMENT) =C MUSIC). If so, then the verb is replaced by the other verb sense for "PLAY" in Arabic, which is (azafa) in the past tense, and the process derives the morphological rules for the present tense. The surface verb or noun can be connected with other semantic values, as in the following format:

Prefix+word+suffix[3]

Here, when connecting the verb to the correct prefix or suffix, more rules are needed to check these values. The value of the tense variable (present) in the F-structure indicates the prefix ("ya" is issued if masculine and ("ta") if feminine). However, for the sake of clarity, the prefixes and the suffixes, in the following examples are separated by the symbol "+". Then the sequence of the constituents will be:

The	boy	plays	football.
al	walad(nom)	layaba(past)	bi-kurati alqademi.
al+walad+u		ya+layab+u	bi-kurati alqademi.

Therefore, the Arabic sentence in VSO order, where the verb precedes the subject.[4] Then the text is accumulated in right-left direction as follows:

yala'abu alwaladu bi-kurati alqademi.

يَلعَبُ الوَلَدُ بِكُرَةِ القَدَم *ya-l'abu 'lwalad-u bikurati 'lqadami*

For the second case, the verb needs to be replaced by the Arabic verb "azaf" (i.e. the other sense of play) and according to the rule it must replace the object, which is "piano" into the Arabic object "payaanuwu". Then the final sentence will be as follows:

[3]Some times the word could have a multiple of these prefixes (or/and) suffixes but this is out of the scope of this research.

[4]That is because of Lisp specification, the arabic letter ع ' is converted to "y" which is not possible to keep it in the text'

يَعزِفُ الوَلَدُ عَلَى البِيَانُو *ya‘zifu 'lwaladu ‘alā 'lpayānuwu.*

Discussion of inserting the best preposition is in the next section.

5.2.2 Rules for Handling the Subject and the Verb Tense

For the past tense, the subject needs to have an attached pronoun to indicate the gender of the suffix. This does not mean ignoring the actual subject (the boy or girl). For example: "the boy played football", will have the Arabic past verb "layab+a" with a diacritical mark ("a") at the end, this is called an "absent pronoun". However; that pronoun is very obvious when using the feminine subject "at" which is ("layiba+ta"). Such pronouns have to be immediately connected to the verb, and the verb with the pronoun, is written as a single word by using the following transfer rule:

```
;Rule to derive the suffix to the verb.
(<SUFF-V> --> (%)
            ((*EOR*
             (((X0 action tense) =c past)
              (*case* (X0  agent gender)
                     (masculine  ((X1 value) = "a-" ))
                     (feminine ((X1 value) = "aT-"))))
              ((X0 action tense) =c present)
                ((X1 value) = "u")))))
```

For the present tense and feminine subject, the verb is preceded by the attached pronoun (singular feminine form) "ta" as in (ta+layab+u) when the PATIENT INSTRUMENT =C GAME and (ta+azaf+u) when the PATIENT INSTRUMENT =C MUSIC by using the following transfer rule:

```
;Rule for selecting the VERB-SENSE depending on the source structure
(<V-ROOT-SEM> --> (%)
              ((*EOR*
               (((X0 agent human) =c yes)
                (*OR*
                 (((X0  patient instrument) =c game)
                  ((X1 value) =  "layab"))
                 (((X0 patient instrument) =c music)
                  ((X1 value) = "azaf")))))))
```

Another semantic attribute must be considered when adding the preposition. The variable value for the object will show which preposition

should be used. For example: plays football will have an Arabic preposition ("bi"), but for "music", another preposition is used which is ("alY"). See the following preposition rule:

```
;;Rule for selecting the preposition.
(<PP> --> (%)
        ((*case* (X0  patient instrument)
            (game  ((X1 value) = "bi-" ))
            (music ((X1 value) = "alY")))))
```

See the full combination of the singular feminine and masculine in Table 1 Appendix F.

5.3 Semantic Attributes for the Dual Forms

Arabic is different from English in the existence of dual form sentences. In the Arabic dual form, the gender is used to modify the noun and the verb as well. By knowing these attributes, the transfer rules change the subject into a dual form and make some changes to both the noun and verb. See the following dual form sentences:
The two boys play football.
The two girls play the piano.
The parser recognises the word ("two") for the dual which indicates the changes for the subject and verb to be in the dual form. There are some changes that need to be dealt with before transferring the sentence to the surface. These changes will affect the subject and gender. Consider the following parsing sentence:

```
<cl> (p"the two boys play football")
>the two boys play football
 1 (1) Interpretation found and took 11.11 seconds of real time
node # 37
;**** Interpretation 1 ***

((CASE *NOMINATIVE)
 (AGENT
  ((AGREEMENT 3SG) (GENDER MASCULINE) (HUMAN YES) (NUMBER DUAL)
   (ROOT BOY)))
 (CONCEPT ((NUMERICAL DUAL) (ROOT TWO)))
 (PATIENT ((INSTRUMENT GAME) (NUMBER SG) (ROOT FOOTBALL)))
 (ACTION ((TENSE PRESENT) (DEFINITENESS DEF) (AGREEMENT 3SG)
   (ROOT PLAY))))
```

The feature (AGREEMENT 3SG) of the subject is taken into account in deciding the new dual subject. Also, the verb must be modified according to the gender of the subject (i.e. modified to verb feminine or verb masculine.). The following semantic rules have been used for processing the dual and the plural forms:

```
;Dealing with the feminine nouns:-
(<N-ROOT-OUT> --> (%)
              ((*EOR* (((X0  agent gender) =c feminine)
                  (*EOR*
                     (((X0 concept numerical) =c  dual)
                      ((X1 value) = "bent-aan") )
                     (((X0 agent number) =c plural)
                     ((X1 value) = "benAAt"))
              (((X0 agent number) =c sg)
                ((X1 value) = (X0  agent Arabic)))))))

;Dealing with the  masculine nouns:-
(<N-ROOT-OUT> --> (%)
              ((*EOR* (((X0  agent gender) =c masculine)
                  (*EOR*
                    (((X0 concept numerical) =c  dual)
                     ((X1 value) = "walad-aan"))
                    (((X0 agent number) =c plural)
                    ((X1 value) = "awalaad"))
              (((X0 agent number) =c sg)
                 ((X1 value) = (X0  agent Arabic)))))))
```

The transfer sequence can be considered as follows:

the	two	boys	play	football.
al	ethnan(dual)	walad (3sg)	layaba (past)	bi–kuraTi alqadami.

Then the VSO order will be:

ya+layaba+u (present) al+waladaan+i (dual) bi–kuraTi alqademi.
يَلعَبُ الوَلَدَانِ بِكُرَةِ القَدَم *ya-l‹abu 'lwaladān-i bi-kurati 'lqadami*

For the feminine, it is necessary to check for gender first and then attach different prefixes for the present tense, as follows:

ta+layaba+u al+bent+an bi–kuraTi alqadami.

تَلعَبُ البِنتَانِ بِكُرَةِ القَدَمِ *ta-lʿabu 'lbintān-i bi-kurati 'lqadami.*

The combination of the dual sentences are found in Table 2 Appendix F.

5.4 Semantic Attributes for the Plural Forms

For the plural form, a modification is effected by the semantic sign (NUMBER PLURAL) in the F-structure. This modification will apply to the verb and noun (feminine or masculine) and according to the word order as before. The same semantic rules as in the previous section are used for extracting the feminine and the masculine nouns:

```
<cl>  (p"The girls played the piano")
>The girls played the piano
 1 (1) Interpretation found and took 23.326 seconds of real time
node # 59
;**** Interpretation 1 ***

((CASE *NOMINATIVE)
 (AGENT
  ((AGREEMENT 3SG)(GENDER FEMININE)(HUMAN YES)(NUMBER PLURAL)(ROOT GIRL)))
 (PATIENT ((INSTRUMENT MUSIC) (NUMBER SG) (ROOT PIANO)))
 (ACTION ((TENSE PAST) (DEFINITENESS DEF) (AGREEMENT 3SG) (ROOT PLAY)))
 (NUMBER SG) (DEFINITE DEF) (ROOT THE))
```

Finally, the plural sentences are translated into following:

1. Plural feminine in the past tense
   عَزَفَت البَنَاتُ عَلَى البِيَانُوُ *ʿazafat albanātu ʿalā 'lpayānuwu*
2. Plural masculine in the past tense
   عَزَفَ ألأَوَلَادُ عَلَى البِيَانُوُ *ʿazafa ›l›awalādu ʿalā 'lpayānuwu*

See the combination of the plural sentences in Table 3 Appendix F.

5.5 Conclusion

In this chapter, the solving of some of the linguistic problems is dealt with. As it has been shown from the tables in Appendix F, a large number of semantics features are tackled and generated. The use of the same formalism for other linguistic problems will enhance this module.

Chapter 6

General Pragmatics

6.1 What is Pragmatics

When we compare the language produced by people to the text produced by existing language generation programs, one thing becomes clear immediately: People can say the same thing in various ways to achieve various effects, and generators can not. It is straightforward to write a language generation program that produces impressive text by associating a sentence template (or some equivalent general grammatical form) with each representational item and then using a grammar to realise the template into surface form. Such a program, however, is not sensitive to anything but the input items, and therefore produces the same output to all readers in all circumstances [Hov88]. Since different realisations of a topic convey different pragmatic effects, the pragmatic aspects of communication must help to control the choices facing the generator. But, though there has been much discussion about what pragmatics as a field of inquiry is all about, no generally accepted scheme has yet emerged. After a review of some relevant literature, a classification is presented of the kinds of pragmatic information that text can convey. However, pragmatic goals are invariably too general to support rules that directly control generator decisions. The general problem is that no clear distinction exists between pragmatics and semantics. Though attempts to establish the distinction were made [Kat80] and were countered [Jac81], the question is not yet resolved. For example, Jackendoff (1981, p. 105), says:

"... the distinction between "semantic" rules of linguistic inference and pragmatic rules of linguistic interaction with general knowledge is less marked than is often supposed".

In AI research on language generation, most work has been done not

on the general *ways* in which pragmatic information can be conveyed, but on what types of pragmatic information is used by speakers and what additional information speakers can convey in the text? In order to begin to study how pragmatics is used in the generator, a number of rather crude assumptions must be made about plausible types of source goals and the relevant characteristics of the target language setting. In MEANA MT system, the pragmatic sentence interpretations are governed by a set of transfer rules. These rules convey the following categorisations:

1. Acknowledgements.
2. Addresses.
3. Proper Names, Places and Titles.
4. Dates and Telephone Numbers.

For parsing the above phrases, different description grammar rules are needed. These rules must be dependent on the pragmatic aspects (on one hand), and must interact with the generation process in order to produce text that serves the Arabic goals (on the other hand).

6.2 Acknowledgements Translation

The techniques used to incorporate the internal representation of the source to determine the same choice of topic (i.e. by including appropriate stress words, adverbs and adjectives and by choosing verbs and nouns with appropriate effect). All these decisions are based upon the affect rule. Following is the set of Acknowledgement rules used in our system:

```
(<SENTENCE> --> (<ACKNOWLEDGEMENT>) ; non-sentences like "OK"
              (((X0 mood) = *acknowledgement)
               (X0 = X1)))
(<SENTENCE> --> (<THANKYOU>) ; non-sentences like "Thank you"
              (((X0 mood) = *acknowledgement)
               (X0 = X1)))
(<SENTENCE> --> (<WELCOME>) ; non-sentences like "you're welcome"
              (((X0 mood) = *acknowledgement)
                (X0 = X1)))
```

These kinds of translations are generated according to the Arabic "Template". (i.e. Sometimes one English word is explained by many Arabic words.) The set for non-sentence acknowledgements like "OK" are: *OK,*

Okay, sure, fine, good, not yet, alright, yes, no, that is all right. The set of the non-sentence acknowledgements like "Thank you" are *Thank you, Thanks.* The set for non-sentence acknowledgement like "you are welcome" are: *You are welcome, you+re welcome and not at all.* Each of these must have further rules as follows:

```
(<THANKYOU>  --> (<THANKYOU> <ADVP>)
                 (((X0 root) = (X1  root))
                  ((X0 modifier) =  X2)))

(<THANKYOU>  --> (t h a n k y o u)
                  (((X0 root) = *thank-you)))

(<THANKYOU>  --> (t h a n k s)
                  (((X0 root) = *thank-you)))
```

Parsing this set will give different F-structures, but will have the same feature value "mood". Let us parse one of the phrases (i.e. "That +s all right"), then we see the following F-structure:

```
<cl> (p"That +s all right")
>That +s all right
 1 (1) Interpretation found and took 2.729 seconds of real time
;**** Interpretation 1 ***

((MOOD *ACKNOWLEDGEMENT) (ROOT *OKAY))
```

The question addressed is how should such language be represented in the generator? In particular where should the information required by the generator reside? What is the relation between the information concepts the generator must express, and the pragmatic and stylistic considerations? The argument presented here is that all the structural aspects of language-rules of grammar, phrases, word patterns should be represented together in the transfer grammar rules in the form of phrases and features of words; this is so that it is closely bound to the system's concept representation and formalism. Then the selection of stylistically appropriate forms of expression is facilitated. This is achieved by accepting the English internal representation F-structure and by passing them to the transfer rules for corresponding Arabic F-structure.values. After finding the Arabic word then by applying the generation rules; the system will produce appropriate Arabic surface structure. Detail of all pragmatic styles are described in the following sections. Table 1, Appendix G shows the Acknowledgement set which have been used to test this module.

6.3 Translation of Addresses

Before dealing with the address phrases, there is a question that must be asked first. Is it necessary to translate the address into Arabic language script or not. Translating proper names for streets and roads into Arabic may be used mainly for spoken language, as well as for written communication. Another possibility is to keep the address in the source language mixed with the target language (i.e. in English form). Since the aim is to build a MT that will do a translation into Arabic form, then the approach adopted here is to generate an address in to the Arabic form. The English grammar rules concerning the phrasal constituents of the address must be based on what is called "the order of environments" (i.e. House No., Street name, Town, Country and State). Each of which contains a specific condition. An example of address environment rules for parsing is:

```
(<ADDRESS>  --> (<STREET-ADDRESS> <CITY-ADDRESS>)
             (((X0 root) = *personal-address)
              ((X0 house-number) = (X1 number value))
              ((X0 street name) = (X1 street-name value))
              ((X0 street type) = (X1 street-name type)) ;For Lane, Ave.
              ((X0 city) = (X2 city-name value))
              ((X0 state) = (X2 state-name value))
              ((X0 code) = (X2 code number-value))))

(<STREET-ADDRESS>  --> (<ADDRESS-NUMBER> <HOUSE-TYPE>) ;For A or B.
   -              (((X0 street-type) = X2)
                   ((X0 number) = X1)))

(<CITY-ADDRESS>  --> (<CITY-NAME> <STATE-NAME> <POST-CODE>)
                 (((X0 city-name) = X1)
                  ((X0 state-name) = X2)
                  ((X0 post-code) = X3)))
```

Some of the above symbols either have a terminal word or are analysed by further rules. Now in order to parse the address, the number must be written in words rather than digits see the parsing of the following address phrase[1]:

```
<cl>(p'(The address is thirty two  Leicester walk Sheffield south
     Yorkshire S zero three seven H U England  $))
```

[1]The $ sign will be used in this module to indicate the end of input sentence.

```
 1 (1) Interpretation found and took 49.38 seconds of real time
node # 53
;**** Interpretation 1 ***

((TYPE ADDRESS) (COUNTRY ENGLAND) (CODE "S03-7HU") (STATE SOUTH-YORKSHIRE)
 (CITY SHEFFIELD) (STREET ((ROOT WALK) (NAME LEICESTER))) (HOUSE-NUMBER 32))
```

Then this F-structure will be passed to the Transfer and Generation Phase in order to have the translated form in Arabic.

6.4 The Proper Names, Places and Titles

There are several grammar rules to define English proper names, each of which has its own F-structure. These rules are created according to the way they are used in general. A selection of titles are permitted (e.g. Miss, Mr, Mister, Mrs, Ms, Doc, and Doctor). Most of these rules will have their own terminal entry. See the following forms:

```
(<PERSON-NAME>  --> (<FIRST-NAME> <LAST-NAME>)
                 (((X0 root) =  person-name)
                  ((X0 first) = (X1 value))
                  ((X0 last) = (X2 value))))

(<PERSON-NAME>  --> (<FIRST-NAME>)
                 (((X0 root) =  person-name)
                  ((X0 first) = (X1 value))))

; Two cases of names proceeded by title.
(<PERSON-NAME>  --> (<TITLE> <LAST-NAME>)
                 (((X0 root) =  person-name)
                  ((X0 title) = X1)
                  ((X0 last) = (X2 value))))

(<PERSON-NAME>  --> (<TITLE> <FIRST-NAME> <LAST-NAME>)
                 (((X0 root) =  person-name)
                  ((X0 title) = X1)
                  ((X0 first) = (X2 value))
                  ((X0 last) = (X3 value))))

;Special case of name who came from a business company.
```

```
(<PERSON-NAME>  --> (<PERSON-NAME> from <CORPORATE-NAME>)
                    (((X0 root) =  person-name)
                     ((X0 name) = (X1 value))
                     ((X0 affiliation) = X3)))
```

Titles from the Arabic point of view, are important, as they indicate the subject gender of the sentence like Miss, Mr, Mrs and Ms have clear gender. But a title like doctor has two genders in Arabic. Therefore, a special semantic rule is needed to show correct meaning. Because of that, a new title for the lady doctor is introduced called "Docf" to show the gender being used. See the parse of the following sentence:

```
<c> (p'(My name is  Docf  Christine McGregor $))
 1 (1) Interpretation found and took 9.546 seconds of real time
node # 16
;**** Interpretation 1 ***

((PROPER PERSON-NAME) (TYPE NOUN-3) (ROOT NAME)
 (TITLE ((ROOT DOCF) (GENDER FEMININE))) (FIRST CHRISTINE) (LAST MCGREGOR))
```

Now for a sentence stating the name of a company such as: Docf Christine McGregor from the Computer Science Department, another lexical rule is needed which defines the place as a terminal word like:

```
(<PERSON-NAME>  --> (<PERSON-NAME> from <CORPORATE-NAME>)
                    (((X0 proper) = person-name)
                     ((X0 name) = X1)
                     (X0 = X3)
                     ((X0 Type) = affiliation)))
```

6.5 Translation of Dates

In natural language, talking about dates and telephone numbers is very common. These are miscellaneous pragmatic phrases, that need special treatment. In order to parse each one, a full description and values will be included in the parsing. See the rules below:

```
(<SENTENCE>  ==> (<DATE>)
                 (((X0 date) = year)
                  (X0 = X1)))
```

```
(<DATE>  --> (<MONTH-DES> <YEAR>)
             (((X0 day-month) = X1)
              ((X0 Year) = X2)))

(<MONTH-DES>  --> (<MONTH> <CARDINAL>)
                  (((X0 month) = X1)
                   ((X0 day) = X2)))
```

For parsing the date phrase, the date must be written in a words starting with the month first, then the day and the year. See the parse of the following date sentence:

```
<cl> (p '(The eighth of December nineteen hundred and ninety four $))
 1 (1) Interpretation found and took 7.89 seconds of real time
node # 19
;**** Interpretation 1 ***

((DATE YEAR) (YEAR ((VALUE "1994") (ROOT YEAR)))
 (DAY-MONTH ((MONTH ((ROOT MONTH) (VALUE 12))) (DAY ((VALUE 8))))))
```

The Muslim's calendar is based on the "Hijra year", which started from the day when the prophet Mohammed (صَلعَم *ṣal'am*)[2], left Mecca for Medina on 16th July, 622 AD. Dates have the word (هِجرِيَّه *hiğriyyah*) in brackets after them, or simply (ه *ha*). As the year has only 354 days, the Muslim year progressively outstrips the Christian year. In the modern Islamic world one seldom sees the Higriyya date alone. Newspapers, for instance, always show the Christian date, which is also operative in commerce and official pronouncements. To indicate the date, the ordinal numbers are usually employed. After the ordinal is put the name of the month, with or without the word شَهر *šahar* (month) inserted before it, and after that, the number of the year with or without the word سَنَةٌ *sanatun* (year) in the genitive or accusative.

There is one problem in generating an Arabic month's name. The Arabic world has three kinds of month name, first using well known Arabic month name[3], second, pure Islamic month names, and the third, in some Arab countries (like Egypt, Morocco, Algeria, Libya, Tunisia, Lebanon and Syria) which were influenced by the French occupation, and use the French or English word for the month literally. In this research the first type will be used, which are recognised by most Arab countries. (i.e well

[2]Means the god upon him.

[3]Like Kanun Althniy, Shobat, I'thar ... etc.

known Arabic month names). See the date example in Table 2, Appendix G.

6.6 Translation of a Telephone Number

A phrase containing a telephone number, again must be written with words rather than digits. The following is the parsing rules for such a phrase:-

```
(<SENTENCE>  ==> (<NP-TEL>)
                 (((X0 mood) = phone-no)
                  (X0 = X1)))

(<NP-TEL>  --> (<AREA-CODE> - <PHONE-NUMBER>)
               (((X0 tel-no) =  X3)
                ((X0 area-code) = X1)
                ((X0 root) = tel-number)))
```

See the parsing of the my telephone number as follows:

```
<cl>(p '(zero one one four - two seven three nine zero one five $))

1 (1) Interpretation found and took 73.821 seconds of real time
node # 26
;**** Interpretation 1 ***

((MOOD PHONE-NO) (ROOT TEL-NUMBER) (AREA-CODE ((VALUE "0114")))
 (TEL-NO ((VALUE "2739015"))))
```

The parser can parse telephone numbers without the code as well as with the code and gives the same F-structure. A full list of pragmatic sentences and phrases are found in Tables 1 and 2, Appendix G.

Chapter 7

The Transformation Phase

7.1 Introduction

The translation process itself happens in three steps: recognition of a structure for the input text, selection of the structure for the target text, (i.e. the representation which will yield the best translation equivalence), and production of output text. The set-up requires two language-dependent representations of the input and the output text. The transfer then becomes a mapping between the two representations which gives translation equivalents among the classes of strings which they characterise. Here, the transfer grammar is expressed in the same formalism as in the Generalised LR parser/Compiler. [TMMK88]. The purpose of this phase is to Add/Remove features to the English F-structure. It only requires the English F-structure which has produced from the parsing phase, and some transfer rules for converting into an Arabic F-structure. This process needs to consult a Bilingual Lexicon in order to find the Arabic words. The final output will be passed to the Generation phase. In this chapter a description of the algorithm used and some examples are given.

7.2 The Transfer Algorithm

The algorithm which is used in this module to convert the English F-structure representation into Arabic F-structure representation is as follows:

For each sublist in English F-structure do:-

- Change the English F-structure list into values.

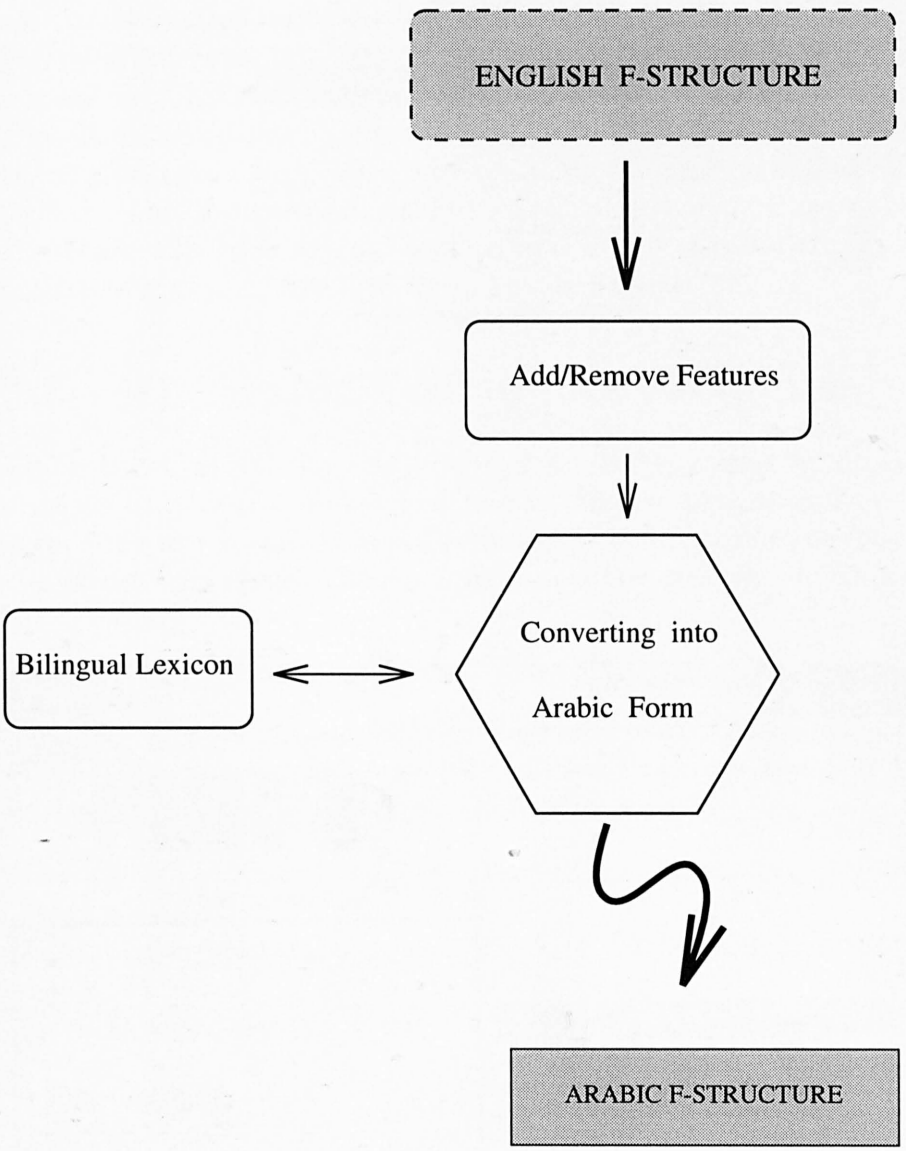


Figure 7.1: **The Transformation Phase**

- Add /Remove feature according to the rule applied.
- Consult the Bilingual Lexicon for the Arabic words.
- Unify these features into Arabic F-structure representation.
- Transfer the Arabic F-structure list to the Generation phase.

The Adding procedure will be adding the Arabic word and some connected articles required in the generation phase (like "_tumma": after). The Removing is considered for unnecessary morphological values features and some auxiliary verbs. In fact, these features do not take any part in the generation phase. The algorithm is repeated until all F-structure sub-lists are exhausted and a new Arabic F-structure representation is created. See the diagram in Figure 7.1.

7.3 An Arabic F-structure Examples

Since the system has been tested on three types of domains, three examples of transfer rules are used. Below the mapping into Arabic F-structure representation is illustrated by one sentence from each domain, the children's story, the semantic and the pragmatic domain.

7.3.1 An Example From The Children's Story

Consider the following parse to the sentence "The mouse cried and clasped her hands begging for mercy". In this example the English F-Structure first then the Arabic F-Structure is followed.

```
<cl> (g"The mouse cried and clasped her hands begging for mercy")

 1 (1) Interpretation found and took 22.993 seconds of real time
node # 83
;**** Interpretation 1 ***

((MOOD *MERCY)
 (S-CONJ
  ((ADVP
    ((ROOT BEG) (TENSE PRESENT) (CAT VERB) (SUBCAT TRANS) (TIME PROGRESS)
     (N ((CAT NOUN) (NUMBER SG) (ROOT MERCY)))
     (PP
      (*OR* ((CAT PREP) (LOCATION OVER) (ROOT FOR))
       ((LOCATION ATTACHE-TO) (CAT PREP) (ROOT FOR))))))
   (CONJ-P
```

```
    ((ROOT CLASP) (TENSE PAST) (CAT VERB) (SUBCAT TRANS)
     (NOUN ((CAT NOUN) (NUMBER DUAL) (ROOT HAND)))
     (PATIENT ((NUMBER 1SG) (CAT PRON) (ROOT HER)))))
   (CONJ ((ROOT AND)))))
 (SUBJECT
  ((NOUN
    ((AGREEMENT 3SG) (CAT NOUN) (GENDER FEMININE) (NUMBER SG) (ROOT MOUSE)))
   (DEFINITE ((NUMBER SG) (DEFINITE DEF) (ROOT THE)))))
 (ACTION ((SUBCAT TRANS) (CAT VERB) (TENSE PAST) (ROOT CRY))))

***** THE TRANSFER PHASE AND THE ARABIC F-STRUCTURE FOLLOWS *****

(((MOOD *MERCY)
  (S-CONJ
   ((ADVP
     ((ARABIC "tawwasala") (TENSE PRESENT) (CAT VERB) (SUBCAT TRANS)
      (TIME PROGRESS) (N ((CAT NOUN) (NUMBER SG) (ARABIC "ra.hamaT")))
      (PP
       (*OR* ((CAT PREP) (LOCATION OVER) (ARABIC "al_A"))
        ((LOCATION ATTACHE-TO) (CAT PREP) (ARABIC "al_A"))))))
    (CONJ-P
     ((ARABIC ".tabbqa") (TENSE PAST) (CAT VERB) (SUBCAT TRANS)
      (NOUN ((CAT NOUN) (NUMBER DUAL) (ARABIC "ya-da")))
      (PATIENT ((NUMBER 1SG) (CAT PRON) (ARABIC "ha")))))
    (CONJ ((ARABIC "wa")))))
  (SUBJECT
   ((NOUN
     ((AGREEMENT 3SG) (CAT NOUN) (GENDER FEMININE) (NUMBER SG)
      (ARABIC "faaraT")))
    (DEFINITE ((NUMBER SG) (DEFINITE DEF) (ARABIC "al-")))))
  (ACTION ((SUBCAT TRANS) (CAT VERB) (TENSE PAST) (ARABIC "baka")))))
```

7.3.2 An Example From The Semantic Domain

```
 <cl>(g"the two  girls play football")

 1 (1) Interpretation found and took 7.284 seconds of real time
node # 38
;**** Interpretation 1 ***

((CASE *NOMINATIVE)
```

```
 (AGENT
  ((AGREEMENT 3SG)
   (GENDER FEMININE) (HUMAN YES)(NUMBER PLURAL)(ROOT GIRL)))
 (CONCEPT ((NUMERICAL DUAL) (ROOT TWO)))
 (PATIENT ((INSTRUMENT GAME) (NUMBER SG) (ROOT FOOTBALL)))
 (ACTION ((TENSE PRESENT) (DEFINITENESS DEF) (AGREEMENT 3SG) (ROOT PLAY))))

***** THE TRANSFER PHASE AND THE ARABIC F-STRUCTURE FOLLOWS *****

(((CASE *NOMINATIVE)
  (AGENT
   ((AGREEMENT 3SG) (GENDER FEMININE) (HUMAN YES) (NUMBER PLURAL)
    (ARABIC "benT")))
  (CONCEPT ((NUMERICAL DUAL) (ARABIC "ethnan")))
  (PATIENT ((INSTRUMENT GAME) (NUMBER SG) (ARABIC "kuraTi-alqademi")))
  (ACTION ((TENSE PRESENT) (DEFINITENESS DEF) (AGREEMENT 3SG)
   (ARABIC "layab")))))
```

7.3.3 An Example From The Pragmatic Domain

```
<cl> (g"you+re welcome")

 1 (1) Interpretation found and took 3.322 seconds of real time
node # 16
;**** Interpretation 1 ***

((MOOD *ACKNOWLEDGEMENT) (ROOT YOU+RE-WELCOME))

***** THE TRANSFER PHASE AND THE ARABIC F-STRUCTURE FOLLOWS *****

(((MOOD *ACKNOWLEDGEMENT) (ARABIC "mar.haba'aN")))
```

Thus, this list will be loaded to the Arabic Generator phase to produce the Arabic language. Detail of the Arabic Generator Grammar rules is in the next chapter.

Chapter 8

The Arabic Generation Grammar

8.1 Introduction

This chapter considers the nature of the generation task, focusing on computational aspects; that is, what a generator needs to do in order to generate well. A generator needs access to a considerable amount of information in order to make correct word choices, such as, selecting the right gender, even though it may not always use it all. The specific target language may not be sensitive to some of it but the information must be there just in case. It is true that there is currently no application which requires (or even allows) a generator to deal with such rich input. In this chapter, a description of a relatively uncontroversial Arabic Generation grammar is presented. Detail of generation of three types of domain are given. The differences between the English and Arabic language features are discussed, and the characteristics of the Arabic Generation Grammar Rules are described in detail.

8.2 Arabic Language Features

Most of the features of the Arabic language have been given in the Arabic Computational Lexicon in chapter 3. In this section further description of the Arabic F-structure that is related to the generation aspects will be given. Arabic, like Latin, is a synthetic, or inflectional language rather than a language like English which is predominantly analytic. This means that the syntactical relationship of nouns is indicated by case endings, and that verbs are inflected by means of prefixes, infixes, and suffixes to indicate the various persons, numbers, genders, derived form, moods, and tenses, in contrast to English where, for example, a separate word

(noun or pronoun) is required to indicate the person. Arabic grammarians recognise only three parts of speech: verbs, nouns, and particles. The concept of verb is the same in Arabic as in English; but adjectives, adverbs, and pronouns are classified as noun; particles include conjunctions, prepositions, and interjections. Arabic has its own unique alphabet system. Arabic script has another distinguishing feature; it is written from right to left.

8.2.1 Arabic Sentence Types

The main features of Arabic grammar will be the word order. The Arabic sentence has two main orders, the first being Subject-Verb-Object (SVO) (sometimes called "Nominative sentence" because it starts with noun at the beginning) and the second Verb-Subject-Object (VSO) (sometimes called "Verbal sentence" because it starts with a verb). The Arabic word and its inflections carry information about the tense, the person, number of the subject and the gender. The subject does not need to be explicitly present, as long as the content and the form of the word makes its reference clear. Although Arabic has a canonical word order, as explained before, the ordering of Arabic constituents appears to be fairly free. The different orders will convey emphasis or focus on different elements of the sentence, but the ordering is still very free. For example:

My father (S) bought (V) a house (O).

This can be seen in different Arabic constituent orders as follows:

1) VSO اِشتَرَى وَالِدِي بَيتًا *ištarā wālediy baytan.*

2) SVO وَالِدِي اشتَرَى بَيتًا *wālediyi 'štarā baytan.*

3) VOS اِشتَرَى بَيتًا وَالِدِي *ištarā baytan wālediy.*

4) OVS بَيتًا اشتَرَى وَالِدِي *baytan ištarā wāledi.*

5) OSV أَلبَيتُ وَالِدِي اِشتَرَى *›albaytu wālidiy ā'štarā.*

Although all these combinations are possible and correct in modern Arabic, the sentences no. 1 and 2 are most frequently used, and the other types of sentences are used for different purposes. The difference between these types is in the different way of reading the words.

8.2.2 Correspondences

One other practical issue that arises is what to do when two sentences to be translated have a different number of words. Up to now, the bilingual

lexicon has been described as a process which establishes a one-to-one correspondence between English F-structure and Arabic F-structure. It is clearly not always the case that two corresponding expressions have the same number of words.

Two basic cases may arise. The first possibility is that a lexical entry corresponds to nothing at all. This can be seen in the translation of auxiliary verbs and some of the prepositions, which have no equivalent in Arabic. One solution is to put constraints in the generator grammar that translate it to an empty string. If there are words that do not contribute anything to the semantics, how can this be shown in the sentence? Alternatively a safe operation to carry that out is to *remove*[1], these words from the English F-structure and allow only the semantic set of words in the Arabic F-structure. See the following grammar rules for removing some of the attribute values from the F-structure:

```
(<ARABIC>  ==> (*WORD)
          (((X1 det def) = *indef)
           ((X1 a-an) = *a)
           ((X1 concept) = *any-time)
           ((X1 det) = *remove*)
           ((X1 anim) = *remove*)
           ((X1 aux) = *remove*)
           ((X1 root) = *remove*)
           (X0 = X1)))
```

The first three equations are used for adding features to the word and the rest are for removing these features from the F-structure. Then these must be unified on the list. (i.e removing these features from the Arabic F-structure). This rule is added to the Transformation rules used prior to the generation phase.

There are cases in which we have to consider one or more words in the source mapped to a different number of words in the target language. (e.g. the expression "happen to come by" in the story: "a mouse happened to come by"), which has been translated in to a single Arabic word, صُدفَةٌ *ṣudfatun*. In the current categorial approaches we say that there is an entry for "come" which subcategorises for "by", the translation of which is صُدفَةٌ *ṣudfatun*, and that a possible translation for "by" is an empty string. This is analogous to an entry in a standard dictionary, where "come by" would be under the "come" list.

[1]In GLR a *remove* function is used, to delete any value or path from the F-structure.

8.3 Writing Generation Grammar Rules

The grammar formalism for the Generation Kit is called the Pseudo Unification Grammar, which is the same formalism as in the Generalized LR Parser/Compiler [TMMK88], also developed at CMU. The pseudo equations are used to check certain attribute values, such as verb form and person agreement. In parsing, these rules are used to combine one or more constituents into a higher constituent. In generation, on the other hand, these rules are used to disassemble a constituent (left hand side) into several lower constituents (right hand side). The current implementation of the generator compiler involves creating a set of LISP functions which represents the grammar of the target language. The diagram in Figure (8.1) shows the compilation process for the Generation grammar rules which have been used to generate the Arabic language. Each function, will be given a name of the form GG-X (where X is any syntactic category), and implements all rewrite rules from the grammar whose left hand symbol is <X>. When GG-X is called with the Arabic F-structure representation of a source language string, if that string can be generated by expansion of the non-terminal <X>, the GG-X returns the representative Arabic language string. More detail about the Generation Kit can be found in [TMMK88]. The first step in writing the generation grammar involves creating the sentence order to be generated. Since any MT system is limited to the number of sentences which are capable of satisfying the generation grammar rules, here this grammar covers the children's story, semantics and the pragmatic sentences. Any generation system which wants to make full use of the features of natural language needs to process both syntax and semantics, therefore, a build up of the sentence is carry out from the Arabic F-structure they represent. The key problem in generation is computing the appropriateness of words given the concepts to the expressed. The task of word choice is considered in terms of how it affects the sentence.

This idea extends naturally to generation as a process in time and computes the appropriateness of each word for the current time. By doing this for each successive word choice it produces an appropriate utterance word by word. This process continues until a successful generation is found, or until all of the rules are exhausted. The algorithm and the whole generation process are given in the next section. Let us consider parsing a sentence as an example. Below the parsing grammar rules which have been used to parse the following sentence: "He seized the mouse in one of his huge paws"

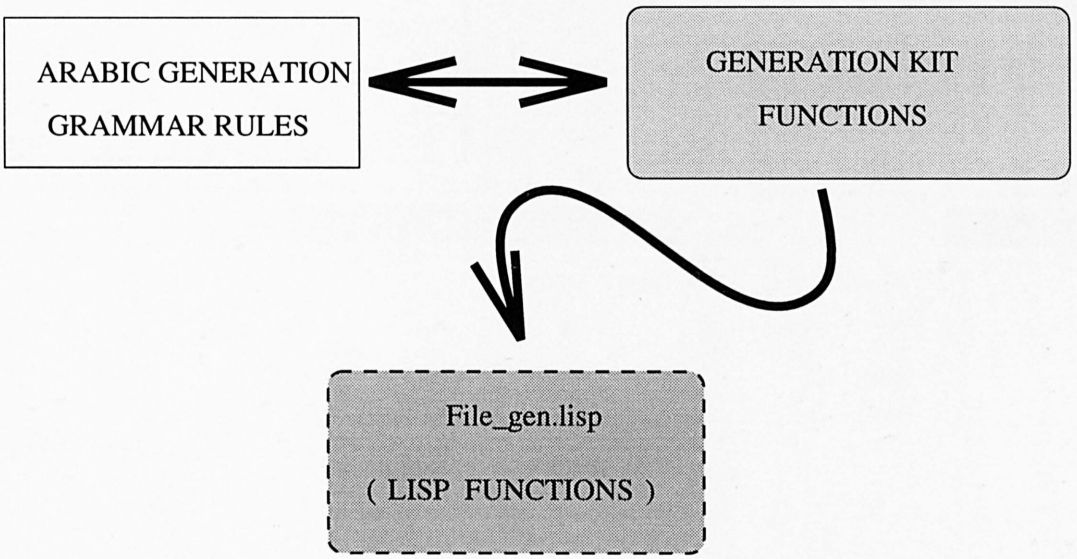


Figure 8.1: **The Compilation Process of the Generation Grammar Rules**

```
(<SENTENCE> ==> (<DECL>)
             ((X0 = X1)
              ((X0 mood) = *declaration)
              ((X0 connect) = "\_tumma")
              ((X0 def) =  "al-")))

(<DECL> ==> (<SIMPLE> <OBJ-P>)
             ((X0  = X1)
              (X0 = X2)))

(<SIMPLE>  ==> (<NP> <VP1>)
            (((X0 action) = X2)
             ((X0 subject) = X1)))

(<OBJ-P>  ==> (<NP> <PREP> <NP> <PP-NP> )
           (((X0 object) = X1)
            ((X0 subj2) = X2)
            ((X0 prep-np) = X3)
            (X0 = X4)))

(<PP-NP>  ==> (<PREP> <POSSIVE> <ADJ> <NP>)
           (((X0 prep) = X1)
            ((X0 patient) = X2)
```

```
            ((X0 adj) = X3)
            ((X0 noun) = X4)))

(<NP>  ==> (<DET> <N>)
          (((X1 number) = (X2 number))
           ((X0 definite) = X1)
           ((X0 noun) = X2) ))

(<NP>  ==> (<N> )
          ((X0 = X1) ))

(<VP1>  ==> (<VP>)
          ((X0 = X1)))

(<VP>  ==> (<V>)
          ((X0 = X1)))
```

Let us call the above grammar "parse-T5". We need a different set of pseudo equations for the generation phase.

8.4 Writing the Arabic Generation Grammar Rules

As the computational grammar formalisms have been mostly used for writing grammars of English, we will concentrate mainly on the differences between Arabic and English. The main features of the Arabic generation grammar will be processing according to sentence word order, subject selection, diacritics and word endings. The Arabic verb inflections carry information about the tense, the person and the number of the subject. The subject does not need to be explicitly present, as long as the discourse context and the form of the verb makes its referent clear. The conditions are similar to those that licence the presence of a subject pronoun in English. The following rules are the generation rules which correspond to the above sentence:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Gen-T5 "He seized the mouse in one of his huge paws"  ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(<VERBAL-ST> ==> (<ARTICLE> <VPHRASE> <NPHRASE> <PRE-PHRASE>)
            ((X1 = X0)
             (X2 = X0)
             (X3 == (X0 object))
```

```
            ((X0 object) = *remove*)
            (X4 = X0)))

(<ARTICLE> --> (%)
          (((X1 value) = (X0 connect))))

(<VPHRASE> --> ( <STEM2> <SUFFIX>)
          ((X1 = X0)
           (X2 = X0)))

(<STEM2> --> (%)
          (((X1 value) = (X0 action  Arabic))))

(<SUFFIX> --> (%)
          (((X1 value) = A)))

(<NPHRASE> --> (<DEF> <NOUN>)
          ((X1 = X0)
           (X2 = X0)))

(<DEF> --> (%)
          (((X1 value) = (X0  definite Arabic))))

(<PRE-PHRASE> ==> ( <ORDINAL> <EPITHET>)
           ((X1 == (X0 prep-np))
            (X2 = X0)))

(<PRE> --> (<PRE> %)
           ((X1 == (X0 subj2) )
            ((X2 value) = (X0 Arabic))))

(<ORDINAL> --> (BIA %)
           ((X1 = X0) ((X2 value) = (X0 Arabic))))

(<EPITHET> ==> (<NOUN-PRON> <DEF-ADJ>)
           ((X1 = X0)
           (X2 = X0)))

(<NOUN-PRON> --> (<NOUN> <PRON1>)
            ((X1 = X0)
             (X2 = X0)))
```

```
(<NOUN> --> (%)
          (((X1 value) = (X0  noun Arabic))))

(<PRON1> --> (%)
          (((X1 value) = (X0  patient Arabic))))

(<DEF-ADJ> --> (% <ADJ1> t)
          (((X1 value) = (X0  def))
           (X2 = X0)))

(<ADJ1> --> (%)
          (((X1 value) = (X0  adj Arabic))))
```

Let us call the above Arabic generation grammar "gen-T5". The Generation Kit takes grammar rules like these and generates a LISP program that implements a run-time sentence generation. In order to make the generation grammar usable, a compilation process needs to be done first[2]. Figure 8.1 shows the compilation process. The syntactic structures produced by the parser and the syntactic structures accepted as input by the generation are identical in form [TMMK88]. The process of generating a surface utterance (i.e a string of words) from an Arabic F-structure is basically the reverse of parsing. In the parser grammar rule, a constraint equation places the information from <NP> inside the subject of the <SIMPLE> during parsing; in generation, the F-structure for a <VERBAL-ST> will be broken up into its constituent parts; each of which will be generated by further recursive applications of grammar rules. In this case, the embedded F-structure that fills the subject slot of the verbal sentence F-structure will be used as input to all of the rules that can possibly generate an <NP>. The generator follows a top-down, depth-first strategy for applying rules during generation. If the current search path fails, the generator backs up to the next application rule. This process continues until a successful generation is found, or until all of the rules are exhausted.

8.5 Arabic Sentence Generation

Figure 8.2, shows that the generator uses the compiled grammar rules in "file_gen.lisp" as guiding rules for the generation process. Whenever,

[2]The command for compiling the generation grammar and converting it into a Lisp function is (compgen"gen-T5").

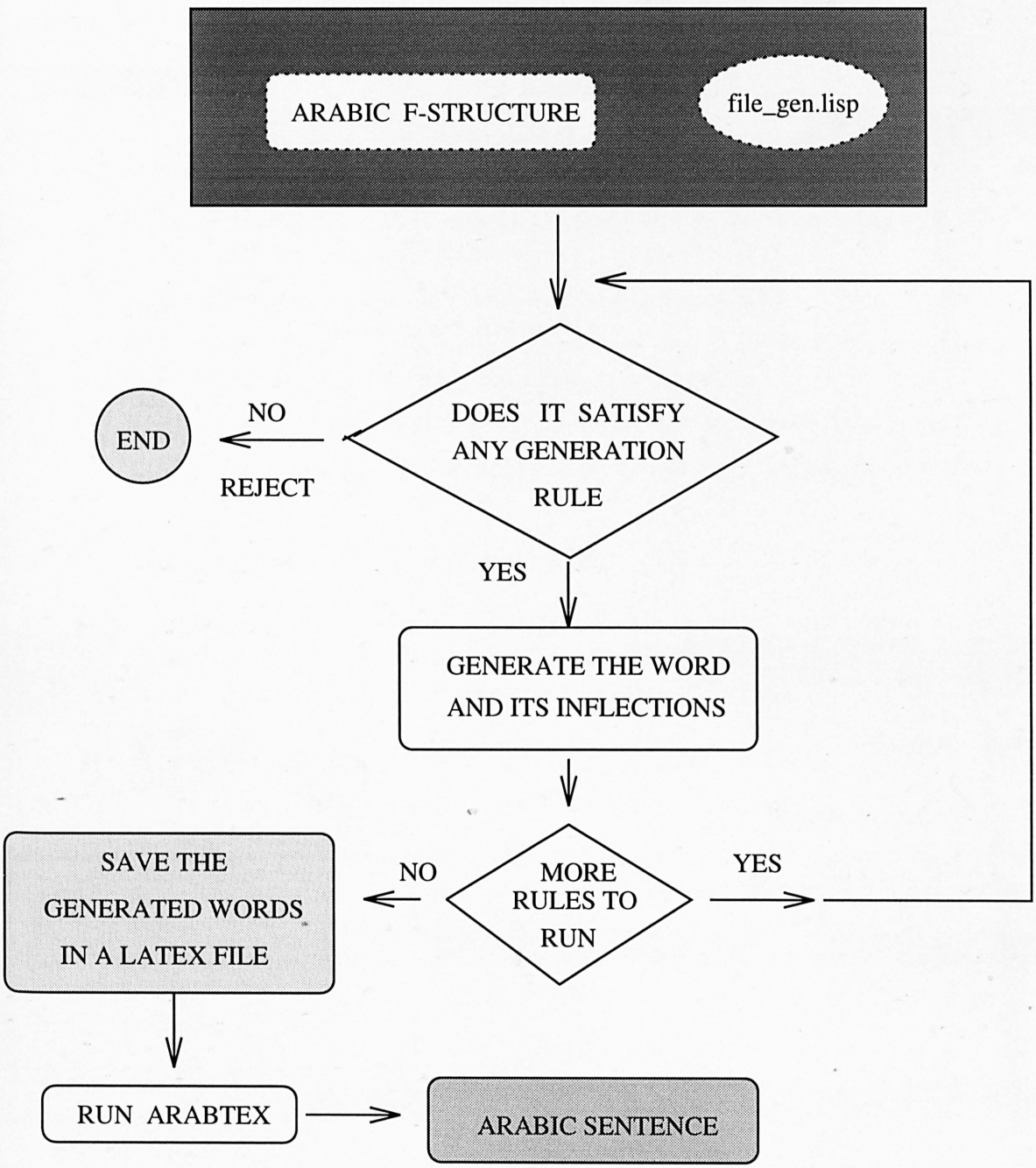


Figure 8.2: **The Generation Phase**

the parsing and the transfer processes are finished, it passes the internal representation (i.e the Arabic F-structure) to the generation phase and the following algorithm will start running:

For each sublist in the Arabic F-structure do:

1. Search for appropriate sentence order rules based on the mood feature.
2. Does the Arabic F-structure satisfy any generation rule. If NOT then END, otherwise;
3. Generate the Arabic word and its inflections.
4. If more rules are waiting to run and the Arabic F-structure list is NOT finished then goto step 2, otherwise
5. Show the translated Arabic sentence in a transliterated form.
6. Save the produced list of Arabic words in a LaTeX file format.

8.6 Complete MT Examples

In order to demonstrate the capability of this system, one example will be used. It shows the complete process of MEANA Machine Translation from English to Arabic, (i.e. Parsing, Transferring and Generation phases).

8.6.1 An Example From The Children's Story

Below is an example that shows the complete stages of translating one sentence which has been chosen from the children's story. This sentence is "He seized the mouse in one of his huge paws". The parsing file "pars-T5" must be compiled first in the normal way; see below:-

```
<cl> (compgra"parse-t5")

***** Start compiling parse-T5.gra
 - Reading parse-T5
 - pars-T5 read
*** Grammar Pre-processor started
*** Grammar Pre-processor done
*** LFG Compiler started
LFG [ 20]
```

```
*** LFG Compiler done
*** LR Table Compiler started
 - converting grammar
 - there were 20                    rules
 - there were 20 really different rules
 - there were 30 symbols
 - there were 16 terminal symbols
 - there were 14 non terminal symbols
 - making augmented grammar
 - making all items
 - 75 items made
 - collecting all items
LR [  0]
LR [  1]
LR [  2]
LR [  3]
LR [  4]
LR [  5]
LR [  6]
LR [  7]
LR [  8]
LR [  9]
LR [ 10]
LR [ 11]
LR [ 12]
LR [ 13]
LR [ 14]
LR [ 15]
LR [ 16]
LR [ 17]
LR [ 18]
LR [ 19]
LR [ 20]
LR [ 40]
 - the number of states is 54
 - generating parsing table
LR'[  0]
LR'[ 20]
LR'[ 40]
 - reforming goto table
*** LR Table Compiler done
```

```
 - Writing File pars-T5.tab
 - File pars-T5.tab written
 - Writing File pars-T5.fun
 - File pars-T5.fun written
 - Writing File pars-T5.funload
 - File pars-T5.funload written
***** Setting up the runtime parser
  Loading File - pars-T5.tab
; Loading /tmp_mnt/home/mash/ahmed/mt/ArabMT/parse-T5.tab.
  Loading File - parse-T5.fun
; Loading /tmp_mnt/home/mash/ahmed/mt/ArabMT/parse-T5.fun.
  Parser Ready for parse-T5
```

The second stage is creating an Arabic F-structure, this can be done by loading the transfer file that will do the mapping into an Arabic words. <cl> (load "transfer")[3]

```
; Loading /tmp_mnt/home/mash/ahmed/mt/ArabMT/transfer.lisp
```

The third stage is compiling the Arabic Generation grammar rules, this can be done by using the GLR command (compgen"gen-T5") in order to create the LISP functions. Now the parser and the generator are ready to accomplish the translation. A command of one letter (i.e (g"sentence")) is used to do the three stages of processing. Below are the three phases of MT (Parsing, Transferring and Generating) in the debugging mode:

```
<cl>  (g"He seized the mouse in one of his huge paws")
>He seized the mouse in one of his huge paws
H
E
 rule #  13 PARS-T5F-52      <PRON>(2) --> HE
 rule #   8 PARSE-T5F-32      <NP>(3) --> <PRON>(2)
S
E
I
Z
E
D
 rule #  11 PARSE-T5F-44      <V>(10) --> SEIZED
 rule #  10 PARSE-T5F-40      <VP>(11) --> <V>(10)
 rule #   9 PARSE-T5F-36      <VP1>(12) --> <VP>(11)
```

[3]All the files are loaded from my working directory.

```
 rule #   3 PARSE-T5F-12      <SIMPLE>(13) --> <NP>(3) <VP1>(12)
T
H
E
 rule #  18 PARSE-T5F-72      <DET>(17) --> THE
M
O
U
S
E
 rule #  17 PARSE-T5F-68      <N>(23) --> MOUSE
 rule #   6 PARSE-T5F-24      <NP>(24) --> <DET>(17) <N>(23)
I
N
 rule #  19 PARSE-T5F-76      <PREP>(27) --> IN
O
N
E
 rule #  16 PARSE-T5F-64      <N>(31) --> ONE
 rule #   7 PARSE-T5F-28      <NP>(32) --> <N>(31)
O
F
 rule #  20 PARSE-T5F-80      <PREP>(35) --> OF
H
I
S
 rule #  14 PARSE-T5F-56      <POSSIVE>(39) --> HIS
H
U
G
E
 rule #  12 PARSE-T5F-48      <ADJ>(45) --> HUGE
P
A
W
S
 rule #  15 PARSE-T5F-60      <N>(50) --> PAWS
 rule #   7 PARSE-T5F-28      <NP>(51) --> <N>(50)
 rule #   5 PARSE-T5F-20      <PP-NP>(52) --> <PREP>(35) <NP>(40)
 <ADJ>(45) <NP>(51)
 rule #   4 PARSE-T5F-16      <OBJ-P>(53) --> <NP>(24) <PREP>(27)
```

```
 <NP>(32) <PP-NP>(52)
 rule #   2 PARSE-T5F-8       <DECL>(54) --> <SIMPLE>(13) <OBJ-P>(53)
 rule #   1 PARSE-T5F-4       <SENTENCE>(55) --> <DECL>(54)

1 (1) Interpretation found and took 7.428 seconds of real time
node # 55
;**** Interpretation 1 ***

((DEF "al-") (CONNECT "_tumma") (MOOD *DECLARATION)
 (ACTION ((TENSE PAST) (CAT VERB) (ROOT SEIZ)))
 (SUBJECT ((NUMBER SG) (CAT PRON) (ROOT HE)))
 (OBJECT
  ((NOUN
    ((AGREEMENT 3SG) (CAT NOUN) (GENDER FEMININE) (NUMBER SG) (ROOT MOUSE)))
   (DEFINITE ((NUMBER SG) (DEFINITE DEF) (ROOT THE)))))
 (SUBJ2 ((CAT PREP) (ROOT IN)))
 (PREP-NP ((CAT NOUN) (SUBCAT TRANS) (COUNTABLE NUMERIC) (ROOT ONE)))
 (NOUN ((CAT NOUN) (NUMBER PLURAL) (ROOT PAW)))
 (ADJ ((GENDER FEMININE) (CAT ADJ) (ROOT HUGE)))
 (PATIENT ((NUMBER 1SG) (CAT PRON) (GENDER MASCULINE) (ROOT HIS)))
 (PREP ((CAT PREP) (ROOT OF))))

***** THE TRANSFER PHASE AND THE ARABIC F-STRUCTURE FOLLOWS *****

(((DEF "al-") (CONNECT "_tumma") (MOOD *DECLARATION)
  (ACTION ((TENSE PAST) (CAT VERB) (ARABIC ".ha.sar")))
  (SUBJECT ((NUMBER SG) (CAT PRON) (ARABIC "hw")))
  (OBJECT
   ((NOUN
     ((AGREEMENT 3SG) (CAT NOUN) (GENDER FEMININE) (NUMBER SG)
      (ARABIC "faaraTa")))
    (DEFINITE ((NUMBER SG) (DEFINITE DEF) (ARABIC "al-")))))
  (SUBJ2 ((CAT PREP) (ARABIC "'lY")))
  (PREP-NP ((CAT NOUN) (SUBCAT TRANS) (COUNTABLE NUMERIC) (ARABIC "a.hdaa")))
  (NOUN ((CAT NOUN) (NUMBER PLURAL) (ARABIC "ma_haalib")))
  (ADJ ((GENDER FEMININE) (CAT ADJ) (ARABIC ".da_hmati")))
  (PATIENT ((NUMBER 1SG) (CAT PRON) (GENDER MASCULINE) (ARABIC "yhi")))
  (PREP ((CAT PREP) (ARABIC "al_A")))))

;******* THE GENERATION PHASE STARTS NOW *****
```

```
Generator> GG-SENTENCE called
Generator>   GG-VERBAL-ST called
Generator>     GG-ART called
Generator>       Rule 1 for <ART> returns _tumma
Generator>     GG-ART returns _tumma
Generator>     GG-VPHRASE called
Generator>       GG-STEM2 called
Generator>         Rule 1 for <STEM2> returns .ha.sar
Generator>       GG-STEM2 returns .ha.sar
Generator>       GG-SUFFIX called
Generator>         Rule 1 for <SUFFIX> returns A
Generator>       GG-SUFFIX returns A
Generator>       Rule 1 for <VPHRASE> returns .ha.sarA
Generator>     GG-VPHRASE returns .ha.sarA
Generator>     GG-NPHRASE called
Generator>       GG-DEF called
Generator>         Rule 1 for <DEF> returns al-
Generator>       GG-DEF returns al-
Generator>       GG-NOUN called
Generator>         Rule 1 for <NOUN> returns faaraTa
Generator>       GG-NOUN returns faaraTa
Generator>       Rule 1 for <NPHRASE> returns al-faaraTa
Generator>     GG-NPHRASE returns al-faaraTa
Generator>     GG-PRE-PHRASE called
Generator>       GG-ORDINAL called
Generator>         Rule 1 for <ORDINAL> returns BIAa.hdaa
Generator>       GG-ORDINAL returns BIAa.hdaa
Generator>       GG-EPITHET called
Generator>         GG-NOUN-PRON called
Generator>           GG-NOUN called
Generator>             Rule 1 for <NOUN> returns ma_haalib
Generator>           GG-NOUN returns ma_haalib
Generator>           GG-PRON1 called
Generator>             Rule 1 for <PRON1> returns yhi
Generator>           GG-PRON1 returns yhi
Generator>           Rule 1 for <NOUN-PRON> returns ma_haalibyhi
Generator>         GG-NOUN-PRON returns ma_haalibyhi
Generator>         GG-DEF-ADJ called
Generator>           GG-ADJ1 called
Generator>             Rule 1 for <ADJ1> returns .da_hmati
Generator>           GG-ADJ1 returns .da_hmati
```

```
Generator>           Rule 1 for <DEF-ADJ> returns al-.da_hmati
Generator>         GG-DEF-ADJ returns al-.da_hmati
Generator>         Rule 1 for <EPITHET> returns ma_haalibyhi al-.da_hmati
Generator>       GG-EPITHET returns ma_haalibyhi al-.da_hmati
Generator>       Rule 1 for <PRE-PHRASE> returns BIAa.hdaa
                     ma_haalibyhi al-.da_hmati
Generator>     GG-PRE-PHRASE returns BIAa.hdaa ma_haalibyhi
                     al-.da_hmati
Generator>     Rule 1 for <VERBAL-ST> returns _tumma .ha.sarA
                     al-faaraTa BIAa.hdaa ma_haalibyhi al-.da_hmati
Generator>   GG-VERBAL-ST returns _tumma .ha.sarA al-faaraTa
                     BIAa.hdaa ma_haalibyhi al-.da_hmati
Generator>   Rule 1 for <SENTENCE> returns _tumma .ha.sarA al-faaraTa
                     BIAa.hdaa ma_haalibyhi al-.da_hmati
Generator> GG-SENTENCE returns _tumma .ha.sarA al-faaraTa
                     BIAa.hdaa ma_haalibyhi al-.da_hmati

"YOUR ENGLISH SENTENCE: "
He seized the mouse in one of his huge paws

"IS TRANSLATED INTO FOLLOWING ARABIC SENTENCE "

"_tumma .ha.sarA al-faaraTa BIAa.hdaa ma_haalibyhi al-.da_hmati"

YOUR SENTENCE IS TRANSLATED INTO  ARABIC AND STORED IN A LaTex
FORMAT IN THE FILE CALLED MT-LaTeX.tex.
IT SHOWS THE FOLLOWING CONTENTS :
\documentstyle[12pt, arabtex,atrans,nashbf]{article}
\input atrans.sty
\sloppy
\parskip =5mm
\frenchspacing
\pagestyle{empty}
\setarab
\vocalize
\transtrue
\arabtrue
\spreadtrue

\begin{document}
\setnashbf \Large
```

```
\begin{arabtext}
\setnash
_tumma .ha.sara al-faarata biaa.hdaa ma_haalibyhi al-.da_hmati.

\end{arabtext}
\end{document}
```

The above file which is in LaTeX format, has embedded commands within the text, these are LaTeX and ArabTex commands.[4] For a complete translation of the story in Arabic script see Appendix K.

8.6.2 An Example Of The Semantics Domain

Below is a full MT to the example taken from the semantics sentences set, but now, without the debugging mode.

```
<cl> (g"The two girls play the piano")

 1 (1) Interpretation found and took 18.34 seconds of real time
node # 39
;**** Interpretation 1 ***

((CASE *NOMINATIVE)
 (AGENT
  ((AGREEMENT 3SG)(GENDER FEMININE)(HUMAN YES)(NUMBER PLURAL)(ROOT GIRL)))
 (CONCEPT ((NUMERICAL DUAL) (ROOT TWO)))
 (PATIENT ((INSTRUMENT MUSIC) (NUMBER SG) (ROOT PIANO)))
 (ACTION ((TENSE PRESENT) (DEFINITENESS DEF) (AGREEMENT 3SG) (ROOT PLAY)))
 (NUMBER SG) (DEFINITE DEF) (ROOT THE))

***** THE TRANSFER PHASE AND THE ARABIC F-STRUCTURE FOLLOWS *****

(((CASE *NOMINATIVE)
  (AGENT
   ((AGREEMENT 3SG) (GENDER FEMININE) (HUMAN YES) (NUMBER PLURAL)
    (ARABIC "benT")))
  (CONCEPT ((NUMERICAL DUAL) (ARABIC "ethnan")))
  (PATIENT ((INSTRUMENT MUSIC) (NUMBER SG) (ARABIC "payaanuw")))
  (ACTION ((TENSE PRESENT) (DEFINITENESS DEF) (AGREEMENT 3SG)
           (ARABIC "layab")))
```

[4]Lisp has stored the sentence in that file in the lower case by default. By using a Lisp converting function will not help as it will convert all the text into upper case.

```
  (NUMBER SG) (DEFINITE DEF) (ARABIC "al-")))

;******* THE GENERATION PHASE STARTS NOW *****

"YOUR ENGLISH SENTENCE: "
The two girls play the piano

"IS TRANSLATED INTO FOLLOWING ARABIC SENTENCE "

"TA-AZAFU ALBENT-AAN-I ALAA AL-payaanuw"

YOUR SENTENCE IS TRANSLATED INTO  ARABIC AND STORED IN  A LaTex
FORMAT IN THE FILE CALLED MT-LaTeX.tex.
IT SHOWS THE FOLLOWING CONTENTS :
\documentstyle[12pt, arabtex,atrans,nashbf]{article}
\input atrans.sty
\sloppy
\parskip =5mm
\frenchspacing
\pagestyle{empty}
\setarab
\vocalize
\transtrue
\arabtrue
\spreadtrue

\begin{document}
\setnashbf \Large

\begin{arabtext}
\setnash
ta-azafu albent-aan-i alaa al-payaanuw.

\end{arabtext}
\end{document}
```

For a complete parsing and generating of different translation list, see Tables 1, 2 and 3 Appendix F.

8.6.3 An Example Of The Pragmatic Domain

Below is an example taken from one of the pragmatic sentences set (i.e the date). The command used is (gg '(phrase or sentence) $).

```
<cl> (gg '(The eighth of december  nineteen hundred and ninety four $))

 1 (1) Interpretation found and took 14.284 seconds of real time
node # 19
;**** Interpretation 1 ***

((DATE YEAR) (YEAR ((VALUE "1994") (ROOT YEAR)))
 (DAY-MONTH ((MONTH ((ROOT MONTH) (VALUE 12))) (DAY ((VALUE 8))))))

***** THE TRANSFER PHASE AND THE ARABIC F-STRUCTURE FOLLOWS *****

(((DATE YEAR) (YEAR ((VALUE "1994") (ARABIC "lisenaTi")))
  (DAY-MONTH ((MONTH ((ARABIC "^sahar") (VALUE 12))) (DAY ((VALUE 8)))))))

;******* THE GENERATION PHASE STARTS NOW *****

"YOUR ENGLISH SENTENCE: "
(THE EIGHTH OF DECEMBER NINETEEN HUNDREDS NINETY FOUR $)
"IS TRANSLATED INTO FOLLOWING ARABIC SENTENCE "

"ALYWUM AL_TAAMEN MIN ^sahar KAANWONA-ALAAWWAL lisanaTi 1994"

YOUR SENTENCE IS TRANSLATED INTO  ARABIC AND STORED IN A LaTex
FORMAT IN THE FILE CLLED MT-LaTeX.tex.
IT SHOWS THE FOLLOWING CONTENTS :
\documentstyle[12pt, arabtex,atrans,nashbf]{article}
\input atrans.sty
\sloppy
\parskip =5mm
\frenchspacing
\pagestyle{empty}
\setarab
\vocalize
\transtrue
\arabtrue
\spreadtrue
```

```
\begin{document}
\setnashbf \Large
\begin{arabtext}
\setnash

alywum al_taamen min ^sahar kaanwona-alaawwal lisanati 1994.

\end{arabtext}
\end{document}
```

See Tables 1 and 2 Appendix G for a complete pragmatic phrases. Detail of converting the LaTeX file into Arabic script is given in the next chapter.

Chapter 9

Converting the Text into Arabic Script

Generally speaking, MT systems are not fully automatic; they require human assistance in some way. This assistance can be categorised as follows[Hut89]:

1. Rejection: Sentence which can not be handled by the system.
2. Pre-editing: Source texts are edited by humans to make them fit the syntax and vocabulary which the MT system can handle.
3. Post-editing: The MT systems takes un-edited source texts and output target texts which require substantial human post-editing (this is the most common procedure in MT).
4. Interactive method: The system requires neither 2 or 3; instead, it requires interactive assistance during the translation.

From the above categories, MEANA MT system needs a little post-editing. In this system the Arabic scripts are converted by using the Arabic supported package within in the LaTeX software (ArabTex V 2.07b) [Lag93]. The ArabTex[1] uses the Latin letters and transliterates them to the corresponding Arabic scripts.

In this system, the last phase generates the words and diacritics in the Latin script; which are saved in a LaTeX file format. More detail about LaTeX can be found in the LaTeX manual [TM88]. Running the LaTeX command generates a Device Independent (dvi) file. This (dvi) file can be reviewed on the screen or converted into a postscript file which could be printed to read the Arabic script.

[1] One of the packages which used to support different natural languages.

9.1 The Post-editing

The actual Arabic transliteration which is produced from the system could be read without post-editing. If we run the LaTeX typesetting for the generated sentence which is in the file MT-LaTeX.tex, we see the Arabic script shown below for the last three examples:-

ṯumma ḥaṣara 'l-fāratu bā'ḥdā maḫālibhu 'l-ḍaḫmati.

ثُمَّ حَصَرَ الفَارَتُ بَااحدَا مخَالِبهُ الضَخمَتِ·

ta-'zafu 'lbent-ān-i 'lā 'lpayānuw.

تَارَفُ البِنتانِ الَا البِيَانُو·

alywum alṯāmen min šahary kānwona-'lāwal lisenati 1994.

اَليوُم الثَامِن مِن شَهَرِي كَانوُنَالَاوَل لِسِنَتِ ١٩٩٤·

These sentences need a little post-editing. These post-editings have come from LaTeX interpretation in generating some Arabic letters. Some Arabic letters have no equivalent in English, therefore, I used the most convenient letter to make the Arabic text readable. However, post editing is necessary to generate some difficult letter and word endings. LaTeX needs some letters to be preceded by specific characters like (_, ; ; ÷.. etc) before converting into Arabic script. (See ArabTeX requirements in Appendix I). The following represents the Arabic script after the post-editing.

ṯumma ḥaṣara 'l-fāratu biāḥdā maḫālibayhi 'l-ḍaḫmati.

ثُمَّ حَصَرَ الفَارَةُ بِاحدَا مخَالِبِيه الضَخمَةِ·

ta-ʿzifu 'lbintaān-i ʿalā 'lpayānuw

تَعزِفُ البِنتَانِ عَلَى البِيَانُو

alyawum alṯāmin min šahr kānwwona 'lāwwal lisanati 1994.

اليَوُم الثَامِن مِن شَهر كَانوُنَ الاوَّل لِسَنَةِ ١٩٩٤·

Perhaps if we use ân Arabised terminal that contains an interface for the input/output of Arabic script, this would be simplified better if the output is transferred into an Arabic word processor which will make the editing easy.

There are some limitations which exist in the ArabTex.package. One of these limitations is the necessity to convert some of the lower case letters into upper case in order to show the Arabic letter or diacritic. Not all upper case letters could produce a script, some of them produce undesirable characters or none. Another limitation is found when representing some of Arabic characters. The ArabTex package uses the dot before the Latin letter in order to generate a specific Arabic letter. Some

of these codes which have been used to generate the Arabic letters and vowels like ض ، ط ، أ ، ع ، ‘ , ’ , ṭ, ḍ... etc. need a special attention. The small "o" in the above Arabic script represent the Arabic Sukun, which indicates the absence of vowel. Such a dot or comma can't be coded in the Lisp environment in the generation phase because Lisp interprets these marks in a different way. These limitation make some of the post-editing unavoidable. Detail of ArabTex can be found in the Appendix I. Of course, this part may be skipped if the Arabic scripts are not required and the user depends on the actual Latin letters (e.g. in the case of using the Arabic Computational lexicon separately).

Chapter 10

Conclusion

This thesis has described the design and implementation of a new prototype MT system. The recent development of the micro computer has introduced new possibilities for processing by computer accordingly computational linguistics has become an important field for manipulating the data of language to determine the patterns underlying the variation and complexity of details found in human language activity. The work has some important implications which will be discussed below.
This model is capable of generating messages of varying complexity. The output text is quite readable and displays cohesive features. More importantly, the model attempts to cover all aspects of MT requirements. That is, it uses a unique grammar formalism from the parsing phase to the generation phase. Unlike some previous attempts, the program decides the setting of the semantics values before the generation phase. The implementation of my program stands as a direct personal MT system. The prototype generates quite readable sentence. However, only a limited range of linguistic features was incorporated and these determine the type of domain in which the program can operate. Any other linguistic features could be dealt with, and added, using the same style.

10.1 MEANA MT Advantages

I have presented a prototype which generates sentences as specified by the grammar. The Arabic generated text is quite readable. The post-editing was enforced by the LaTeX software, in the absence of a convenient Arabic word processor in the department. Compared with other researches in Arabic MT, MEANA MT is a direct translation from the source sentence as input and finishing with the Arabic sentence as output. This system is simple and elegant even although it has been tested on one children's

story, but it is possible to modify the grammar to cover other stories. In addition, the MEANA MT system has many advantages such as:

1. Unique formalism grammar for Parsing, Translation and Generation is used in the MEANA MT system.
2. New fundamental approach of a personal MT system for small documents or messages.
3. An Arabic Computational lexicon has been created. The principles of unification features that concern the diacritical marks are covered. This module offers a very good resource for Computer Aided Learning (CAL) of Arabic words.
4. This system solves some semantic problems concerning the morphological aspect and word sense. The semantic grammar rules are very reliable and prevent undesirable words or sentences.
5. The Arabic generation grammar rules cover the language cases and types dealing with the prefix, suffix and infix marks in the system, and deal with a variety of words and conditions through an intelligent method.

In chapter one, I discussed the choice of the Generalized LR (GLR) Parser/Compiler, developed by Tomita [Tom86]. I have used the GLR tool mainly for compiling the grammars. The grammars were the Context-Free Phrase Structure Grammar for the English parser and the Arabic generator grammar. These grammars are consulting other transfer grammar rules and some other Lisp functions written for this purpose. So all the grammars for MEANA MT are compiled in the GLR and Common Lisp environments. The English lexicon and the bilingual lexicon have been created to support the system. The required lexicon for the two languages has 265 entries for the Children's story alone, but there are a further 230 entries for the semantic and pragmatic domain. The Arabic computation lexicon is different as it uses a small set of entries and produces many inflection words in Arabic. Details of their size has been covered in chapter four.
Since, the GLR* (the new version) can now handle two particular problems that are typical of parsing spontaneous speech (i.e. noise contamination and limited grammar coverage), then a spoken language input could be parsed by such a system, and Arabic output could be generated.

10.2 Future Research

Future directions include enhancing the linguistic generation and allowing for more sophisticated planning as follows:

- Different research could be done in the area of Arabic parsers. An experiment of building an Arabic parser dealing with the reverse direction has already been tried during this study, and was found to work perfectly. A sample of Arabic grammar for parsing has been written and compiled during the same test and showed good results. Other research has been done in this parser field like Arabic Language Interpretation (ALI) [Meh87], which used the gulf war domain sentences, or the ARABTN system from Roochnik[Roo89]. Building a new Arabic parser would be much faster and more efficient if it is based on this thesis.

- Writing a "Reverse Grammar" for both parsing and generation at the same time. In this field (i.e. a grammar) some work was carried out at the Centre of MT at CMU for a different project. An evaluation of their work would be useful before carrying out any research in the field of Reverse Grammar.

- Personal MT: the aim of this kind of MT will be dominated by imposing certain constraints. Commercial and business applications: need a specific Personal MT which is suitable for their requirements. I believe such systems will be necessary in the future. A study of specific business field requirements and the building of an individual Arabic generator will be beneficial to business. A Personal MT needs to be further investigated. Such MT would have to deal with many unstructured phrases, sentences or business terminology in a field requiring special attention.

All of the aims and implementations discussed above are based on the ideas presented in this thesis.

APPENDICES

Appendix A

INPUT DATA AND THE PARSING GRAMMAR RULES

(1)THE CHILDREN'S STORY[1]
A mouse's gratitude

A lion was taking a nap in the grassland of Africa when a mouse happened to come by.
Mistaking the lion's mane for grass the mouse rustled about in it.
Who is disturbing my nap.
He seized the mouse in one of his huge paws.
What it is only a mouse.
You are just what I would like for my lunch.
The mouse cried and clasped her hands begging for mercy.
Mr lion please spare my life.
My seven children are waiting for my return.
You are a mother of seven.
Yes I am.
A tiny mouse like me will not be enough to satisfy your hunger.
If you will kindly spare my life.
I promise to repay you without fail.

Below is the story input to MEANA MT:-

```
(g"A mouse +s gratitude")
```

[1]The command "g" is created to represent the parsing and the generation processes.

```
(g"A lion was taking a nap in the grassland of Africa when a mouse
   happened to come by")
(g"Mistaking the lion +s mane for grass the mouse rustled about in it")
(g"Who is disturbing my nag")
(g"He seized the mouse in one of his huge paws")
(g"What it is only a  mouse")
(g"You are just what I would like for my lunch")
(g"The mouse cried and clasped her hands begging for mercy")
(g"Mr lion please spare my life")
(g"My seven children are waiting for my return")
(g"You are a mother of seven")
(g"Yes I am")
(g"A tiny mouse like me will not be enough to  satisfy your hunger")
(g"If you will kindly spare my life")
(g"I promise to repay you without fail")
```

(2) THE PARSER GRAMMAR RULES FOR THE ABOVE STORY WRITTEN IN LFG

The grammar rules which have been used in parsing the English story sentences. Each rule will call the appropriate grammar rules.

```
(<SENTENCE> ==> (<SIMPLE>)
                (((X0 mood) = *nominative)
                 (X0 = X1)))

(<SENTENCE> ==> (<RELATION-ST>)
                ((X0 = X1)
                 ((X0 prep) = "\'l-y")
                 ((X0 mood) = *relation)))

(<SENTENCE> ==> (<DECLARATION>)
                (((X0 mood) = *declaration)
                 ((X0 def) = "al-")
                 ((X0 connect) = "fa-")
                 (X0 = X1)))

(<SENTENCE> ==> (<WHQ> )
                (((X0 mood) = *question)
                 (X0 = X1)))

(<SENTENCE> ==> (<DECL>)
                ((X0 = X1)
                 ((X0 mood) = *declaration)
                 ((X0 connect) = "\_tumma")
                 ((X0 def) = "al-")))

(<SENTENCE> ==> (<SWHQ>)
                (((X0 mood) = *wondering)
                 ((X0 connect) = "'an")
                 (X0 = X1)))

(<SENTENCE> ==> (<STATE>)
                (((X0 mood) = *action)
                 ((X0 relative) = "maA")
                 (X0 = X1)))
```

```
(<SENTENCE> ==> (<AND-ST>)
                (((X0 mood) = *mercy)
                 (X0 = X1)))

(<SENTENCE> ==> (<REQUEST-ST>)
                (((X0 mood) = *request)
                 (X0 = X1)))

(<SENTENCE> ==> (<REPLY>)
                ((X0 = X1)
                 ((X0 mood) = *reply)))

(<SENTENCE> ==> (<DECL2>)
                ((X0 = X1)
                 ((X0 def) = "al-")
                 ((X0 mood) = *declaration)))

(<SENTENCE> ==> (<QUESTION>)
                ((X0 = X1)
                 ((X0 aux-pron) = "i")
                 ((X0 mood) = *question)))

(<SENTENCE> ==> (<CONDITION>)
                ((X0 = X1)
                 ((X0 and) = "w")
                 ((X0 imperfect-pron) = "t-t")
                 ((X0 mood) = *condition)))

(<SENTENCE> ==> (<PROMISE>)
                ((X0 = X1)
                 ((X0 noun-An) = "an")
                 ((X0 mood) = *promise)))
```

Appendix B

THE NOUN INFLECTIONS

Table (No.1) Shows the Regular Nouns Inflections Forms

1-	مُعَلِّمٌ *mu'allimun*	Singular-NOM- Masculine. (Teacher)
2-	مُعَلِّمُونَ *mu'allimuwna*	Plural-NOM-Masculine. (Teachers)
3-	مُعَلِّمِينَ *mu'allimiyna*	Plural-GEN/ACC-Masculine. (Teachers)
4-	مُعَلِّمَانِ *mu'allimāni*	Dual-NOM-Masculine. (Two teachers)
5-	مُعَلِّمَيْنِ *mu'allimayni*	Dual-GEN/ACC-Masculine. (Two teachers)
6-	مُعَلِّمَةٌ *mu'allimatun*	Singular-NOM. (Teacher (Feminine))
7-	مُعَلِّمَةٍ *mu'allimatin*	Singular-GEN. (Teacher (Feminine))
8-	مُعَلِّمَةً *mu'allimatan*	Singular-ACC. (Teacher (Feminine))
9-	مُعَلِّمَتَانِ *mu'allimatāni*	Dual-NOM. (Two teachers (feminine))
10-	مُعَلِّمَتَيْنِ *mu'allimatyni*	Dual-ACC. and GEN. (Two teachers (feminine))
11-	مُعَلِّمَاتٌ *mu'allimātun*	Plural-NOM. (Teachers (Feminine))
12-	مُعَلِّمَاتٍ *mu'allimātin*	Plural-GEN/ACC. (Teachers (Feminine))

Table (No.2) Shows an Irregular Plural Nouns (Broken)

1- مَطَرٌ *maṭarun* Singular-Masculine. (Rain)

2- أَمطَارٌ *ʾamṭārun* Plural-NOM-Masculine. (Rain)

3- أَمطَارٍ *ʾamṭārin* Plural-GEN-Masculine. (Rain.)

4- مَطَرَانِ *maṭarāni* Dual-NOM-Masculine. (Two rain)

5- مَطَرَيْنِ *maṭarayni* Dual-GEN and ACC-Masculine. (Two rain)

6- مَطَرَةٌ *maṭaratun* Singular-NOM. (Rain (Feminine))

7- مَطَرَةٍ *maṭaratin* Singular-GEN. (Rain (Feminine))

8- مَطَرةً *maṭartan* Singular-GEN. (Rain (Feminine))

9- مَطَرَتَانِ *maṭaratāni* Dual-NOM-Feminine. (Two rain drops (feminine))

10- مَطَرَتَيْنِ *maṭaratayni* Dual-ACC and GEN-Feminine (Two rain (feminine))

11- Failed to parse أَمَطَرَاتٌ *amaṭaraātun* (because it does Not Exist)

12- Failed to parse أَمَطَرَاتٍ *amaṭaraātin* (because it does Not Exist)

Table (No.3) Shows the Acceptable and Unacceptable Noun inflections

ACCEPTABLE FORMS:

1- مَاءٌ *māʾun* Singular and Plural. (water)

2- مَاءَانِ *māʾāni* Dual -Masculine - NOM. (Two water)

3- مَاءَيْنِ *māʾayni* Dual-Masculine - GEN and ACC. (Two water)

FAILED FORMS TO GENERATE OR TO PARSE (UNACCEPTABLE INFLECTIONS):

4- مَاؤُونَ *māʾuwna* 5- مَاءِينَ *māʾyina* 6- مَاءةٌ *māʾtun* 7- مَاءةٍ *māʾtin*

8- مَاءةً *māʾtan* 9- مَاءَتَانِ *māʾataān* 10- مَاءَتَيْنِ *māʾatayn*

11- مَاءَاتٌ *māʾaātun* 12- مَاءَاتٍ *māʾaātin*

Table (No.4) Shows the Nouns Inflections with the Attached Pronouns

1- مُعَلِّمِي *mu'allimiy* 1st person - Singular Masculine and Feminine (My teacher)

2- مُعَلِّمُنَا *mu'allimunā* 1st person - Plural NOM Masculine and Feminine (Our teacher)

3- مُعَلِّمَكَ *mu'allimaka* 2nd person - Singular ACC Masculine only (Your teacher)

4- مُعَلِّمَكِ *mu'allimaki* 2nd person - Singular ACC Feminine only (Your teacher)

5- مُعَلِّمُكُمَا *mu'allimukumā* 2nd person - Dual NOM Masculine and Feminine (Your teacher)

6- مُعَلِّمُكُمْ *mu'allimukum* 2nd person - Plural NOM Masculine (Your teachers)

7- مُعَلِّمُكُنَّ *mu'allimukunna* 2nd person - Plural NOM Feminine (Your teacher)

8- مُعَلِّمَهُ *mu'allimahu* 3rd Person - Singular ACC Masculine (His teacher)

9- مُعَلِّمَهَا *mu'allimahā* 3rd person - Singular ACC Feminine (Her teacher)

10- مُعَلِّمَهُمَا *mu'allimahumā* 3rd person - Dual ACC Masculine (Their teacher)

11- مُعَلِّمُهُمَا *mu'allimuhumā* 3rd person - Dual NOM Feminine (Their teacher)

12- مُعَلِّمُهُمْ *mu'allimuhum* 3rd person - Plural NOM Masculine (Their teacher)

13- مُعَلِّمُهُنَّ *mu'allimuhunna* 3rd person - Plural NOM Feminine (Their teacher)

Table (No.5) Shows the Regular Singular Noun Inflections with the Attached Pronouns.

1- وَلَدِي *waladiy* 1st person - Singular Masculine and Feminine (My boy)
2- وَلَدُنَا *waladunā* 1st person - Plural NOM Masculine and Feminine (Our boy)
3- وَلَدُكَ *waladuka* 2nd person - Singular NOM Masculine only (Your boy)
4- وَلَدُكِ *waladuki* 2nd person - Singular NOM Feminine only (Your boy)
5- وَلَدُكُمَا *waladukumā* 2nd person - Dual Masculine and Feminine NOM (Your boy)
6- وَلَدُكُمْ *waladukum* 2nd person - Plural Masculine NOM (Your boy)
7- وَلَدُكُنَّ *waladukunna* 2nd person - Plural Feminine NOM (Your boy)
8- وَلَدُهُ *waladuhu* 3rd Person - Singular Masculine NOM (His boy)
9- وَلَدُهَا *waladuhā* 3rd person - Singular Feminine NOM (Her boy)
10- وَلَدُهُمَا *waladuhumā* 3rd person - Dual Masculine NOM (Their boy)
11- وَلَدُهُمَا *waladuhumā* 3rd person - Dual Feminine NOM (Their boy)
12- وَلَدُهُمْ *waladuhum* 3rd person - Plural Masculine NOM (Their boy)
13- وَلَدُهُنَّ *waladuhunna* 3rd person - Plural Feminine NOM (Their boy)

Table (No.6) Shows an Irregular Plural Noun with the Attached Pronouns.

1- أَولَادِي *ʾawlādiy* 1st person - Singular Masculine and Feminine (My boys)

2- أَولَادُنَا *ʾawlādunā* 1st person - Plural NOM Masculine and Feminine (Our boys)

3- أَولَادُكَ *ʾawlāduka* 2nd person - Singular Masculine NOM (Your boys)

4- أَولَادُكِ *ʾawlāduki* 2nd person - Singular Feminine NOM (Your boys)

5- أَولَادُكُمَا *ʾawlādukumā* 2nd person - Dual Masculine and Feminine NOM (Your boys)

6- أَولَادُكُمْ *ʾawlādukum* 2nd person - Plural Masculine NOM (Your boys)

7- أَولَادُكُنَّ *ʾawlādukunna* 2nd person - Plural Feminine NOM (Your boys)

8- أَولَادُهُ *ʾawlāduhu* 3rd Person - Singular Masculine NOM (His boys)

9- أَولَادُهَا *ʾawlāduhā* 3rd person - Singular Feminine NOM (Her boys)

10- أَولَادُهُمَا *ʾawlāduhumā* 3rd person - Dual Masculine NOM (Their boys)

11- أَولَادُهُمَا *ʾawlāduhumā* 3rd person - Dual Feminine NOM (Their boys)

12- أَولَادُهُمْ *ʾawlāduhum* 3rd person - Plural Masculine NOM(Their boys)

13- أَولَادُهُنَّ *ʾawlāduhunna* 3rd person - Plural Feminine NOM (Their boys)

Table (No.7) Shows the Noun for Both a Singular and a Plural with the Attached Pronouns

1- مَائِي *mā'iy* **1st person - Singular Masculine and Feminine (My water)**

2- مَاؤُنَا *mā'unā* **1st person - Plural Masculine and Feminine NOM. Our water)**

3- مَاؤُكَ *mā'uka* **2nd person - Singular Masculine only NOM (Your water)**

4- مَاؤُكِ *mā'uki* **2nd person - Singular Feminine only NOM (Your water)**

5- مَاؤُكُمَا *mā'ukumā* **2nd person - Dual Masculine and Feminine NOM (Your waters)**

6- مَاؤُكُمْ *mā'ukum* **2nd person - Plural Masculine NOM (Your waters)**

7- مَاؤُكُنَّ *mā'ukunna* **2nd person - Plural Feminine NOM (Your waters)**

8- مَاؤُهُ *mā'uhu* **3rd Person - Singular Masculine NOM (His water)**

9- مَاؤُهَا *mā'uhā* **3rd person - Singular Feminine NOM (Her water)**

10- مَاؤُهُمَا *mā'uhumā* **3rd person - Dual Masculine NOM (Their water)**

11- مَاؤُهُمَا *mā'uhumā* **3rd person - Dual Feminine NOM (Their water)**

12- مَاؤُهُمْ *mā'uhum* **3rd person - Plural Masculine NOM (Their water)**

13- مَاؤُهُنَّ *mā'uhunna* **3rd person - Plural Feminine (Their water)**

Appendix C

THE VERB INFLECTIONS

Table (No.1) Shows the Arabic Verb Patterns from II to X.

Form Type	The perfect	The Imperfect
II	فَعَّلَ *fa‘‘ala*	يُفَعِّلُ *yufa‘‘ilu*
III	فَاعَلَ *faā‘ala*	يُفَاعِلُ *yufā‘ilu*
IV	أَفعَلَ *›af‘ala*	يُفعِلُ *yuf‘ilu*
V	تَفَعَّلَ *tafa‘‘ala*	يَتَفَعَّلُ *yatafa‘‘alu*
VI	تَفَاعَلَ *tafā‘ala*	يَتَفَاعَلُ *yatafā‘alu*
VII	اِنفَعَلَ *infa‘ala*	يَنفَعِلُ *yanfa‘ilu*
VIII	اِفتَعَلَ *ifta‘ala*	يَفتَعِلُ *yafta‘ilu*
IX	اِفعَلَّ *if‘alla*	يَفعَلُّ *yaf‘allu*
X	اِستَفعَلَ *istaf‘ala*	يَستَفعِلُ *yastaf‘ilu*

Table (No.2) Shows the Conjugation of the Verb Patterns from II to X. (without patterns No. III, VI and VIII)

FORM TYPE	PERFECT	IMPERFECT THREE CASES
II (MASC)	كَسَّرَ *kassara*	يُكَسِّرُ *yukassiru* يُكَسِّرَ *yukassira* يُكَسِّر *yukassir*
II (FEM)	كَسَّرَت *kassarat*	تُكَسِّرُ *tukassiru* تُكَسِّرَ *tukassira* تُكَسِّر *tukassir*
IV (MASC)	أَكسَرَ *ʾaksara*	يُكسِرُ *yuksiru* يُكسِرَ *yuksira* يُكسِر *yuksir*
IV (FEM)	أَكسَرَت *ʾaksarat*	تُكسِرُ *tuksiru* تُكسِرَ *tuksira* تُكسِر *tuksir*
V (MASC)	تَكَسَّرَ *takassara*	يَتَكَسَّرُ *yatakassaru*, يَتَكَسَّرَ *yatakassara* يَتَكَسَّر *yatakassar*
V (FEM)	تَكَسَّرَت *takassarat*	تَتَكَسَّرُ *tatakassaru*, تَتَكَسَّرَ *tatakassara*, تَتَكَسَّر *tatakassar*
VII (MASC)	إِنكَسَرَ *inkasara*	يَنكَسِرُ *yankasiru*, يَنكَسِرَ *yankasira*, يَنكَسِر *yankasir*
VII (FEM)	إِنكَسَرَت *inkasarat*	تَنكَسِرُ *tankasiru*, تَنكَسِرَ *tankasira*, تَنكَسِر *tankasir*
IX (MASC)	إِحمَرَّ *iḥmarra*	يَحمَرُّ *yaḥmarru* يَحمَرَّ *yaḥmarra* يَحمَرِر *yaḥmarir*
IX (FEM)	إِحمَرَّت *iḥmarrat*	تَحمَرُّ *taḥmarru* تَحمَرَّ *taḥmarra* تَحمَرِر *taḥmarir*
X (MASC)	إِستَحضَرَ *istaḥḍara*	يَستَحضِرُ *yastaḥḍiru*, يَستَحضِرَ *yastaḥḍira*, يَستَحضِر *yastaḥḍir*
X (FEM)	إِستَحضَرَت *istaḥḍarat*	تَستَحضِرُ *tastaḥḍiru*, تَستَحضِرَ *tastaḥiḍira*, تَستَحضِر *tastaḥḍir*

Table (No.3) Shows The Conjugation of the Perfect Indicative Stem with Attached Pronouns to the Verb (كَتَبَ *kataba*)

SINGULAR

3 Person Masc. The ending is "a", e.g. (كَتَبَ *kataba*) He has written (or he wrote).

3 Person Fem. The ending is ت *t*, e.g. (كَتَبَت *katabat*) She has written (or she wrote).

2 Person Masc. The ending is تَ *ta*, e.g. (كَتَبتَ *katabta*) You (man) have written .

2 Person Fem. The ending is تِ *ti*, e.g. (كَتَبتِ *katabti*) You (woman) have written.

1 Person Masc. and Fem The ending is تُ *tu*, e.g. (كَتَبتُ *katabtu*) I have written.

DUAL

3 Person Masc. The ending is ا *ā*, e.g. (كَتَبَا *katabā*) The Two (men) have written.

3 Person Fem. The ending is تَا -*tā*, e.g. (كَتَبَتَا *katabatā*) The two (women) have written .

1 Person Masc. and Fem. The ending is تُمَا *tumā*, e.g. (كَتَبتُمَا *katabtumā*) You have written.

PLURAL

3 Person Masc. The ending is وُا *wuā*, e.g. (كَتَبُوا *katabuwā*) They have written.

3 Person Fem. The ending is نَ *na*, e.g. (كَتَبنَ *katabna*) They (women) have written.

2 Person Masc. The ending is تُم *tum*, e.g. (كَتَبتُم *katabtum*) You (men) have written.

2 Person Fem. The ending is تُنَّ *tunna*, e.g. (كَتَبتُنَّ *katabtunna*) You (women) have written .

1 Person Masc. and Fem. The ending is نَا *nā*, e.g. (كَتَبنَا *katabnā*) We have written.

Table (No.4) Shows The Conjugation of the Imperfect Indicative Stem with Attached Pronouns to the Verb (كَتَبَ *kataba*)

SINGULAR

١- يَكتُبُ *1- yaktubu* ٢- تَكتُبُ *2- taktubu* ٣- تَكتُبُ *3- taktubu*

٤- تَكتُبِينَ *4- taktubiyna* ٥- أَكتُبُ *5- ʾaktubu*

DUAL

٦- يَكتُبَانِ *6- yaktubāni* ٧- تَكتُبَانِ *7- taktubāni* ٨- تَكتُبَانِ *8- taktubāni*

PLURAL

٩- يَكتُبُونَ *9- yaktubuwna* ١٠- يَكتُبْنَ *10- yaktubna* ١١- تَكتُبُونَ *11- taktubuwna*

١٢- تَكتُبْنَ *12- taktubna* ١٣- نَكتُبُ *13- naktubu*

Table (No.5) Shows The Conjugation of the Imperfect Subjunctive Stem with the Attached Pronouns to the Verb (كَتَبَ *kataba*)

SINGULAR

١- يَكتُبَ *1- yaktuba* ٢- تَكتُبَ *2- taktuba* ٣- تَكتُبَ *3- taktuba*

٤- تَكتُبِي *4- taktubiy* ٥- أَكتُبَ *5- ʾaktuba*

DUAL

٦- يَكتُبَا *6- yaktubā* ٧- تَكتُبَا *7- taktubā* ٨- تَكتُبَا *8- taktubā*

PLURAL

٩- يَكتُبُوا *9- yaktubuwā* ١٠- يَكتُبْنَ *10- yaktubna* ١١- تَكتُبُوا *11- taktubuwā*

١٢- تَكتُبْنَ *12- taktubna* ١٣- نَكتُبَ *13- naktuba*

Table (No.6) Shows The Conjugation of the Imperfect Jussive Stem Attached Pronouns to the Verb كَتَبَ *kataba*

SINGULAR

١- يَكتُب *1- yaktub* ٢- تَكتُب *2- taktub* ٣- تَكتُب *3- taktub*

٤- تَكتُبِي *4- taktubiy* ٥- أَكتُب *5- ʾaktub*

DUAL

٦- يَكتُبَا *6- yaktubā* ٧- تَكتُبَا *7- taktubā* ٨- تَكتُبَا *8- taktubā*

PLURAL

٩- يَكتُبُوا *9- yaktubwā* ١٠- يَكتُبْنَ *10- yaktubna* ١١- تَكتُبُوا *11- taktubwā*

١٢- تَكتُبْنَ *12- taktubna* ١٣- نَكتُب *13- naktub*

Appendix D

IRREGULAR VERBS: VERB "TO BE" AND "NOT TO BE"

Table (No.1) Shows the Perfect of the Verb "to be" كَانَ *kāna*

Sing. 3. Masc. كَانَ *kāna* He was.
Sing. 3. Fem. كَانَت *kānat* She was.
Sing. 2. Masc. كُنتَ *kunta* You (Masc) were.
Sing. 2. Fem. كُنتِ *kunti* You (Fem) were.
Sing. 1. Masc. and Fem. كُنتُ *kuntu* I was.

Dual. 3. Masc.كَانَا *kānā* They two (Masc) were.
Dual. 3. Fem. كَانَتَا *kānatā* They two (Fem) were.
Dual. 2. Masc. and Fem كُنتُمَا *kuntumā* You two were.

Plur. 3. Masc. كَانُوا *kānuwā* They (Masc) were.
Plur. 3. Fem. كُنَّ *kunna* They (Fem) were.
Plur. 2. Masc. كُنتُم *kuntum* You (Masc) were.
Plur. 2. Fem. كُنتُنَّ *kuntunna* You (Fem) were.
Plur. 1. Masc. and Fem. كُنَّا *kunnā* We were.

Table (No.2) Shows the Perfect of the Verb (لَيسَ *laysa*).

Sing. 3. Masc. لَيسَ *laysa* He was.
Sing. 3. Fem. لَيسَت *laysat* She was.
Sing. 2. Masc. لَستَ *lasta* You (Masc) were.
Sing. 2. Fem. لَستِ *lasti* You (Fem) were.
Sing. 1. Masc. and Fem. لَستُ *lastu* I was.

Dual. 3. Masc. لَيسَا *laysā* They two (Masc) were.
Dual. 3. Fem. لَيسَتَا *laysatā* They two (Fem) were.
Dual. 2. Masc. and Fem لَستُمَا *lastumā* You two were.

Plur.3. Masc. لَيسُوَا *laysuwā* They (Masc) were.
Plur. 3. Fem. لَسنَ *lasna* They (Fem) were.
Plur. 2. Masc. لَستُم *lastum* You (Masc) were.
Plur. 2. Fem. لَستُنَّ *lastunna* You (Fem) were.
Plur. 1. Masc. and Fem.لَسنَا *lasnā* We were

Appendix E

THE PRONOUNS AND MISCELLANEOUS MODULE

(a) The Demonstrative pronouns

1- (this, these)

	Masculine	*Feminine*
Singular, all cases	هٰذَا *hādā*	هَذِهِ *haḏihi*
Dual Nominative	هَذَانِ *haḏāni*	هَاتَانِ *hātāni*
Accusative and Genitive	هَذَيْنِ *haḏayni*	هَاتَيْنِ *hātayni*
Plural, all cases, masc and fem is	هَؤُلَاءِ *haʾulāʾi*	

2- That, those.

	Masculine	*Feminine*
Singular	ذَاكَ *ḏāka*	تِيكَ *tiyka*
Dual Nominative	ذَانِكَ *ḏānika*	تَانِكَ *tānika*
Accusative and Genitive	ذَيْنِكَ *ḏaynika*	تَيْنِكَ *taynika*
Plural, all cases, masc and fem is	أُولَائِكَ *ʾuwlāʾika*	

(b) The personal pronouns are as follows:

أَنَا *ʾanā* means I
أَنتَ *ʾanta* means You (masculine)
أَنتِ *ʾanti* means you (feminine)
هُوَ *huwa* means he , it.
هِيَ *hiya* means she, it
أَنتُمَا *ʾantumā* Dual: means you (masculine)
نَحنُ *naḥnu* plural: means we (masculine and feminine)
أَنتُم *ʾantum* plural: means you (masculine)
أَنتُنَّ *ʾantunna* plural: means you (feminine)
هُمّ *humm* plural: means those (masculine)
هُنَّ *hunna* plural: means those (feminine)

(c) The Relative Pronouns set is expressed by اَلَّذِي *alladī*, which declined as follows:

	Singular	*Dual*	*Plural*
Masc. Nom.	اَلَّذِي *alladī*	اَللَّذَانِ *al-laḍāni*	اَلَّذِينَ *alladīna*
ACC. and Gen.	اَلَّذِي *alladī*	اَللَّذَينِ *alllaḍayni*	اَلَّذِينَ *alladiyna*
Fem. Nom.	اَلَّتِي *allatī*	اَللَّتَانِ *al-latāni*	اَللَّاتِي *al-lātī* or اَللَّوَاتِي *al-lawātī*
ACC. and Gen.	اَلَّتِي *allatī*	اَللَّتَينِ *al-latyni*	اَللَّاتِي *alllātī* or اَللَّوَاتِي *al-lawātī*

مَن *man* means who, whomsoever
مَا *mā* means that, which
مَاذَا *māḍā* means what
أَيٌّ *ʾyyun* means whichever and whatever

(d) The Particles "Ann and its Sisters" set as follows:

أَنَّ *ʾnn* **Means Verily**

إِنَّ *ʾnna* **means that**

لَكِنَّ *lakinna* **Means But**

لِأَنَّ *liʾnna* **Means because**

لَعَلَّ *laʿalla* **Means Perhaps**

Appendix F

WORD SENSE DISAMBIGUATION TABLES

Table no. 1 shows a Verbal sentence containing Singular, Feminine and Masculine. It includes changing the subject, verb and object as well. Parsing for the following input:

(g"The boy plays football")
(g"The boy plays the piano")
(g"The boy played football")
(g"The boy played the piano")
(g"The girl plays football")
(g"The girl plays the piano")
(g"The girl played football")
(g"The girl played the piano")

The Latin forms after the generation phrase:
ya-lyabu alwalad-u. bi- -kurati-alqadami.
ya-azafu alwalad-u alY alpayaanuw.
-layab-a alwalad-u bi- -kurati-alqadami.
-azaf-a alwalad-u alY alpayaanuw.
ta-layabu albent-u bi- -kurati-alqadami.
ta-azafu albent-u alY alpayaanuw.
-layabat albent-u bi- -kurati-alqadami.
-azafat albent-u alY alpayaanuw.

The final Arabic sentence after post-editing.
It includes converting some small letters to capital.[1]
ya-lʿabu alwalad-u bikurati alqadami

يَلعَبُ اَلوَلَدُ بِكُرَةِ اَلقَدَمِ

ya-ʿzifu alwalad-u ʿalā alpayānuw

يَعزِفُ اَلوَلَدُ عَلَى اَلپِيَانُو

laʿib-a alwalad-u bikurati alqadami

لَعِبَ اَلوَلَدُ بِكُرَةِ اَلقَدَمِ

ʿazafa alwalad-u ʿalā alpayānuw

عَزَفَ اَلوَلَدُ عَلَى اَلپِيَانُو

ta-lʿab-u albent-u bikurati alqadami

تَلعَبُ اَلبِنتُ بِكُرَةِ اَلقَدَمِ

ta-ʿzif-u albent-u ʿalā alpayānuw

تَعزِفُ اَلبِنتُ عَلَى اَلپِيَانُو

laʿibat albent-u bikurati alqadami

لَعِبَت اَلبِنتُ بِكُرَةِ اَلقَدَمِ

ʿazafat albent-u ʿalā alpayānuw

عَزَفَت اَلبِنتُ عَلَى اَلپِيَانُو

Table no. 2 generation of the verbal sentences with features Dual Feminine and Masculine. It has two semantic values for instrument (Piano) and game (ball). It includes changing the subject, verb and object as well. Parsing for the following input:

(g"The two boys play football")
(g"The two boys play the piano")
(g"The two girls play football")
(g"The two girls play the piano")
(g"The two boys played football")
(g"The two boys played the piano")
(g"The two girls played football")
(g"The two girls played the piano")

[1]**Mainly adding the letter (ع ʿ) which is not possible to keep it in the text; because of Lisp specification.**

The Latin forms after the generation phrase:

ya-lyabu alwalad-aan-i bi- -kurati-alqadami.
ya-azafu alwalad-aaan-i aly alpayaanuw.
ta-layabu albent-aaan-u bi- -kurati-alqadami.
ta-azafu albent-aaan-i aly alpayaanuw.
layab-a alwalad-aaan-i bi- -kurati-alqadami.
azaf-a alwalad-aaan-i aly alpayaanuw.
layabati albent-aaan-i bi- -kurati-alqadami.
azafati albent-aaan-i aly alpayaanuw.

The final Arabic sentence after post-editing.

ya-lʿabu alwaladaān-i bikurati alqadami

يَلعَبُ اَلوَلَدَانِ بِكُرَةِ اَلقَدَمِ

ya-ʿzifu alwaladaān-i ʿalā alpayānuw

يَعزِفُ اَلوَلَدَانِ عَلَى اَلبِيَانُو

ta-lʿabu albenitaāan-u bikurati alqadami

تَلعَبُ اَلبِنتَانُ بِكُرَةِ اَلقَدَمِ

ta-ʿzifu albentaān-i ʿalā alpayānuw

تَعزِفُ اَلبِنتَانِ عَلَى اَلبِيَانُو

laʿib-a alwaladaān-i bikurati alqadami

لَعِبَ اَلوَلَدَانِ بِكُرَةِ اَلقَدَمِ

ʿazaf-a alwaladaān-i ʿalā alpayānuw

عَزَفَ اَلوَلَدَانِ عَلَى اَلبِيَانُو

laʿibati albentaān-i bikurati alqadami

لَعِبَتِ اَلبِنتَانِ بِكُرَةِ اَلقَدَمِ

ʿazafati albentaān-i ʿalā alpayānuw

عَزَفَتِ اَلبِنتَانِ عَلَى اَلبِيَانُو

Table no. 3 Generation of the verbal sentences with features Plural (Feminine and Masculine). Parsing for the following input:

(g"The boys play football")
(g"The boys play the piano")
(g"The girls play football")
(g"The girls play the piano")
(g"The boys played football")

(g"The boys played the piano")
(g"The girls played football")
(g"The girls played the piano")

The Latin forms after the generation phrase:

ya-lyabu alawalaad-u bi- -kurati-alqadami.
ya-azafu alawalaad-u aly alpayaanuw.
ta-layabu albenaat-u bi- -kurati-alqadami.
ta-azafu albenaat-u aly alpayaanuw.
layab-a alawalaad-u bi- -kurati-alqadami.
azaf-a alawalaad-u aly alpayaanuw.
layabati albenaat-u bi- -kurati-alqadami.
azafati albenaat-u aly alpayaanuw.

The final Arabic sentence after post-editing.

ya-lʿabu alʾwalād-u bikurati alqadami	يَلعَبُ اَلأَوَلَادُ بِكُرَةِ اَلقَدَمِ
ya-ʿzifu alʾwalād-u ʿalā alpayānuw	يَعزِفُ اَلأَوَلَادُ عَلَى اَلبِيَانُو
ta-lʿabu albanāt-u bikurati alqadami	تَلعَبُ اَلبَنَاتُ بِكُرَةِ اَلقَدَمِ
ta-ʿzifu albanāt-u ʿalā alpayānuw	تَعزِفُ اَلبَنَاتُ عَلَى اَلبِيَانُو
la̅ʿib-a alʾwalād-u bikurati alqadami	لَعِبَ اَلأَوَلَادُ بِكُرَةِ اَلقَدَمِ
ʿazaf-a alʾwalād-u ʿalā alpayānuw	عَزَفَ اَلأَوَلَادُ عَلَى اَلبِيَانُو
laʿibati albanāt-u bikurati alqadami	لَعِبَتِ اَلبَنَاتُ بِكُرَةِ اَلقَدَمِ
ʿazafati albanāt-u ʿalā alpayānuw	عَزَفَتِ اَلبَنَاتُ عَلَى اَلبِيَانُو

Appendix G

THE PRAGMATIC PHRASES

Table no. 1: THE ACKNOWLEDGEMENTS

The Acknowledgement set used as follows:
(g"you+re welcome") (g"hello")(g"yes") (g"no")
(g"not yet") (g"thanks") (g"thank you") (g"goodbye")
(g"correct") (g"right") (g"may i help you") (g"okay") (g"I ve got it")
(g" let +s see") (g"I understand") (g"fine") (g"good") (g"sure")
(g"I am sorry") (g"thats all") (g"not at all") (g"I see") (g"oh")

The pragmatic phrases after parsing and generating into Arabic script as follows:

marḥaba›an

مَرحَبَاً

-

›ahlan

أَهلًا

naʿam

نَعَم

lā

لَا

laysa ›al›āna

لَيسَ أَلآنَ

šukran

شُكرًا

šukran laka

شُكرًا لَكَ

maʿa 'lsalāmati

مَعَ السَلَامَةِ

ṣaḥiyḥun

صَحِيحٌ

ṣaḥiyḥun

صَحِيحٌ

hal ʾastaṭiyʿu musāʿadataka

هَل أَستَطِيعُ مُسَاعَدَتَكَ

ḥasanan

حَسَنًا

ḥaṣaltu ʿalayhi

حَصَلتُ عَلَيهِ

daʿnā narā

دَعنَا نَرَى

ʾinnanī fāhimun

إِنَّنِي فَاهِمٌ

lā baʾsa

لَا بَأسَ

ǧayyidan

جَيِّدًا

mutaʾakkadun

مُتَأَكِّدٌ

ʾanā ʾāsifun

أَنَا آسِفٌ

haḏā kulluhu

هَذَا كُلُّهُ

kallā muṭlaqan

كَلَّا مُطلَقًا

fahamtu

فَهِمتُ

ʾwh

أوه

Table no. 2: OTHER PRAGMATIC PHRASES

The parsing and generating other pragmatic phrases such as Telephone number, the proper names with different titles (Masculine and Feminine), The address and The Date are follows:
(gg'(My NAME IS jim McGregor $))
(gg'(My NAME IS jim $))
(gg'(My NAME IS mr jim McGregor $))
(gg'(My NAME IS mister jim McGregor $))
(gg'(My NAME IS Doctor jim McGregor $))
(gg'(My NAME IS Doc jim McGregor $))
(gg'(My NAME IS mrs christine McGregor $))
(gg'(My NAME IS miss christine McGregor $))
(gg'(My NAME IS Docf christine McGregor $))
(gg '(my name is mr jim McGregor from computer science department $))
(gg'(The address is thirty two Leicester Walk Sheffield south Yorkshire S zero three seven H U England $))
(gg '(The eighth of December nineteen hundred and ninety four $))
(gg '(zero one one four - two seven three nine zero one five $))

The generated sentences without the post-editing:
Asmy ĝim magrygor.
Asmy ĝim.
Asmy sayed ĝim magrygor.
Asmy sayed ĝim magrygor.
Asmy dukatwur ĝim magrygor.
Asmy dukatwur ĝim magrygor.
Asmy sayedaTa CHRISTINE magrygor.
Asmy Aanasat CHRISTINE magrygor.
Asmy dukatwurat CHRISTINE magrygor.
Asmy sayed ĜIM MAGRYGOR MAN qasam 'alumu al.ha-si-bat.
ALEINWAAN ALRRAQAM 32 mamŝY LEICESTER MADYNT SHEFFIELD RAMAZ S03-7HU WALAAYT SOUTH-YORKSHIRE ENGLAND.
ALYAWUM AL_TAAMIN MIN ŝahar KAANWONA-ALAAWWAL lisenaTi 1994.

Now the translated sentences in Arabic script after post-editing.

ʾismiy ğim magariyigor.

إِسمِي جِم مَگرِيگُر.

ʾismiy ğim.

إِسمِي جِم.

ʾismiy sayyed ğim magariyigor.

إِسمِي سَيِّد جِم مَگرِيگُر.

ʾismiy sayyed ğim magariyigor.

إِسمِي سَيِّد جِم مَگرِيگُر.

ʾismiy dukutwur ğim magariyigor.

إِسمِي دُكتُور جِم مَگرِيگُر.

ʾismiy d. ğim magariyigor.

إِسمِي د. جِم مَگرِيگُر.

ʾismiy sayyidat karistiyin magariyigor.

إِسمِي سَيِّدَة كَرِستِين مَگرِيگُر.

ʾismiy ʾanisaha karistiyin magariyigor.

إِسمِي أَنِسَة كَرِستِين مَگرِيگُر.

ʾismiy dukatuwrata karistiyin magariyigor.

إِسمِي دُكتُورَة كَرِستِين مَگرِيگُر.

ʾismiy sayyed ğim magariyigor min qism ʿilmu ʾalḥāsi-bāti.

إِسمِي سَيِّد جِم مَگرِيگُر مِن قِسم عِلمُ أَلحَاسِبَاتِ.

alʿunwān alraqm 32 mamšy leyester madīnat šaffyeld ramz s03 7 h u wilāyt ʾinakaltarrha.

اَلعُنوَان الرَقم ٣٢ مَمشي لِيستِر مَدِينَة شَفِّيلد رَمز س٣٠ ٧ ٥ ا وِلَاية إِنكَلتَرّة.

alyawum alṯāmin min šahr kānwwona ʾlāwwal lisanati 1994.

اَليَوم الثَامِن مِن شَهر كَانوُنَ الاَوَّل لِسَنَةِ ١٩٩٤.

raqm altilifuwni 2739015 0114

رَقم التِلِفُونِ ٢٧٣٩٠١٥ ٠١١٤.

Appendix H

THE ARABIC GENERATION GRAMMAR RULES

```
;;=========================================================================
;;   Arabic Generation Grammar Rules (Arabic GenKit) in MEANA MT.
;;   The following grammar rules have been used in the generation phase.
;;   Each of which will call the appropriate grammar rules.
;;=========================================================================

(<START> -->  (<SENTENCE-LIST>)
              ((X1 = X0)))

(<SENTENCE-LIST> --> (<SENTENCE>)
               ((X1 = X0)))

(<SENTENCE> ==> (<SEMANTICS-ST>)
                (((X0 case) = *nominative)
                 (X1  == X0)))

; Grammar rules for Generating the Date.

(<SENTENCE> --> (<ARABIC-YEAR>)
                (((X0 date) =c year)
                 (X1 = X0)))

;Grammar rules for generating the Addresses.

(<SENTENCE> --> (<ADDRESS>)
                (((X0 type) =c address)
```

```
                (X1 = X0)))

; Grammar rule for Generating the Proper names and Titles.

(<SENTENCE> --> (<PERSON-NAME>)
                (((X0 proper) = person-name)
                 (X1 = X0)))

; Grammar rules for generating the Phone Number.

(<SENTENCE> --> (<PHONE-NUMBER>)
                (((X0 mood) =c phone-no)
                 (X1 = X0)))

; Grammar rules for generating the Children's Story sentences.

(<SENTENCE> --> (<WHQ>)
                (((X0 mood) = *question)
                 (X1 = X0)))

(<SENTENCE> --> (<NOM-ST>)
                (((X0 mood) = *nominative)
                 (X1 = X0)))

(<SENTENCE> --> (<NOM-ST-past>)
                (((X0 MOOD) = *declaration)
                 (X1 = X0)))

(<SENTENCE> --> (<NOM-ST-REL>)
                ((X1 = X0)))

(<SENTENCE> --> (<VERBAL-ST>)
                (((X0 mood) = *declaration)
                 (X1 = X0)))

(<SENTENCE> --> (<DECLARATION>)
                (((X0 mood) = *declaration)
                 (X1 = X0)))

(<SENTENCE> --> (<WHQ-ST>)
                (((X0 mood) = *wondering)
```

```
                (X1 = X0)))

(<SENTENCE> --> (<STATEMENT>)
                (((X0 mood) = *action)
                 (X1 = X0)))

(<SENTENCE> --> (<CONJ-ST>)
                (((X0 mood) = *mercy)
                 (X1 = X0)))

(<SENTENCE> --> (<REQUEST-ST>)
                (((X0 mood) = *request)
                 (X1 = X0)))

(<SENTENCE> --> (<DECL2>)
                (((X0 mood) = *declaration)
                 (X1 = X0)))

(<SENTENCE> --> (<QUESTION>)
                (((X0 mood) = *question)
                 (X1 = X0)))

(<SENTENCE> --> (<REPLY>)
                (((X0 mood) = *reply)
                 (X1 = X0)))

(<SENTENCE> --> (<CONDITION>)
                (((X0 mood) = *condition)
                 (X1 = X0)))

(<SENTENCE> --> (<PROMISE>)
                (((X0 mood) = *promise)
                 (X1 = X0)))

;Non-sentences like "ok".

(<SENTENCE>  --> (<ACKNOWLEDGEMENT>)
                 (((X0 mood) = *acknowledgement)
                  (X1 = X0)))
```

The complete list of the generation grammar rules could be obtained from the author.

Appendix I

A r a b T e X Document

This document was presented here by permission
from the author Klaus Lagally.

arabtex.doc 05.11.1992

A r a b T e X Version 2
==========

What is ArabTeX?
==========
ArabTeX is a package extending the capabilities of TeX/LaTeX to generate the Arabic writing from an ASCII transliteration for texts in several languages using the Arabic script.

It consists of a TeX macro package and an Arabic font in several sizes, presently only available in the Naskhi style. ArabTeX will run with Plain TeX and also with LaTeX; other additions to TeX have not been tried.

ArabTeX is primarily intended for generating the Arabic writing, but the scientific transcription can be also easily generated. For other languages using the Arabic script limited support is available.

Input coding:
==========
The ASCII input notation for Arabic text is modelled closely after the transliteration standards ISO/R 233 and DIN 31 635. As these standards do not guarantee unique re-transliteration and are also not ASCII compatible, some modifications were necessary. These follow the general rules:

- if the transliteration uses a single letter, code that letter;

- if the transliteration uses a letter with a diacritical mark, put a special character similar to the diacritical mark BEFORE the letter.

Standard Arabic and persian characters:

b	bah	d	dal	.s	ssad	f	fah	h	hah	'	hamza
t	tah	_d	dhal	.d	ddad	q	qaf	w	waw	N	tanween
_t	thah	r	rah	.t	ttah	k	kaf	y	yah	Y	alif maqsoura
^g	geem	z	zay	.z	tthah	l	lam	g	gaf	_A	alif maqsoura
.h	hhah	s	seen	‘	‘ain	m	meem	p	pah	T	tah marbouta
_h	khah	^s	sheen	.g	ghain	n	noon	v	vah	W	waw (see below)

Additional characters generally available:

c	hhah with hamza
^c	gim with three dots (below)
,c	khah with three dots (above)
^z	zay with three dots (above)
~n	kaf with three dots (Ottoman)
~l	law with a bow accent (Kurdish)
~r	rah with two bows (Kurdish)

See also "Urdu" and "Pashto" below.

Additional coding rules:

- For long vowels, use capital letters <A>, <I>, <U>, or <_a>, <_i>, <_u>.

- As the transliteration is ambiguous, use <T> for <tah marbouta>, <N> for <tanween>, <Y> or <_A> for <alif maqsoura>.

- Short vowels <a>, <i>, <u> need not generally be written except in the following cases:
  - at the beginning of a word where they generate <alif>,
  - adjacent to <hamza> where they will influence the carrier,
  - when the transcription is wanted,
  - in \fullvocalize mode.

- <hamza> is denoted by a single RIGHT quote; its carrier will be determined from the context according to the rules for writing Arabic words. If that is not wanted, "quote" it (see below).

- <'ain> is a single LEFT quote, don't confuse it with <hamza>!

- <madda> is generated by a right quote (<hamza>) before <A>: <'A>.

- The "invisible letter" <|> may be inserted in order to break unwanted ligatures and to influence the <hamza> writing. It will not show in the Arabic output or in the transcription.

- <tashdid> is indicated by doubling the appropriate letter.

- The article is always written <al-> (with hyphen!).

- Hyphens <-> may be used freely, and generally do not change the writing, but will show up in the transcription. At the beginning and the end of a word they enforce the use of the connection form of the adjacent letter (if it exists), like e.g. in the date <1400 h->.

- A double hyphen <--> between two otherwise joining letters will break any ligature and will insert a horizontal stroke (<tatweel>, <kashida>) without appearing in the transcription. It may be used repeatedly.

Quoting:
========
A double quote <"> will modify the meaning of the following character as follows:

- if a short vowel follows, the appropriate diacritical mark <fatha>, <kasra>, <damma> will be put on the preceding character even if the vocalization is off otherwise. If <N> follows the short vowel, the appropriate form of <tanween> will be generated instead. At the beginning of a word, <alif> is assumed as the first character. If the previous word ended with a vowel, <wasla> is generated instead of the vowel indicator.

- if the following character is a single right quote, a <hamza> mark will

be put on the preceding character even if in conflict with the <hamza> rules.

- if the following character is the "invisible letter" <|>, the connection between the adjacent letters will be broken and a small space inserted.

- otherwise: a <sukun> will be put on the preceding character. The following character will be processed again.

The double quote will not show up in the transcription.

Ligatures:
==========
There is no way to explicitly indicate ligatures as a large number of them are generated automatically. Any unwanted ligature can be suppressed by interposing the invisible letter <|> between the two letters otherwise combined into a ligature. After "\ligsfalse" ligatures in the middle of a word will not normally be produced; for some texts this looks better. You can return to the normal strategy by "\ligstrue".

Vocalization:
=============
There are three modes of rendering short vowels:

\fullvocalize:
- every short vowel will generate the corresponding diacritic mark <fatha>, <kasra>, <damma>.

- If <N> follows a short vowel, the corresponding form of <tanween> is generated instead.

- <_a> will produce a <qur'an alif> accent instead of an explicit <alif> character which is coded <A>.

- if a long vowel follows a consonant, the corresponding short vowel is implied. The long vowel itself carries no diacritical mark.

- if no vowel is given after a consonant, <sukun> will be generated except if a double "sun letter" follows <lam>.

- <alif> at the beginning of a word carries <wasla> instead of the vowel

indicator if the preceding word ended with a vowel.

\vocalize: as above, but <sukun> and <wasla> will not be generated except if explicitly indicated by "quoting".

\novocalize: no diacritics will be generated except if explicitly asked for by "quoting".

In all modes, a double consonant will generate <tashdid>, and <'A> always generates <madda> on <alif>. After <aN> the silent <alif> character is generated if necessary. The silent <alif> may also be explicitly indicated by <aNa> or <aNA>, or coded literally as <A> in \novocalize mode. If a silent <alif maqsoura> is wanted instead, write <aNY>, <aN_A>, <Y> or <_A>. A silent <alif> after <waw> is indicated by <Ua>, <UA> or <Wa>, <WA> (with a capital <W>!).

Appendix J

MEANA-MT SYSTEM GUIDE

In order to run MEANA MT, first load lisp, then the initial file.cl will be executed. The following menu will be displayed which is easy to follow.

```
Allegro CL 3.1.13.1 [Sun4] (0/0/0)
Copyright (C) 1985-1990, Franz Inc., Berkeley, CA, USA
; Loading /export/home/ahmed/mt/ArabMT/glr-meanaMT.lisp.

"************************************************************"
"** English  -  Arabic  Machine  Translation  System      **"
"**                  ( MEANA MT)                           **"
"** MEANA includes the following packages:                 **"
"** English Parser  and Lexicon.                           **"
"** Arabic Computational Lexicon.                          **"
"** Arabic GenKit.                                         **"
"**                                                        **"
"** Using The Generalized LR Parser/Compiler/Interpreter **"
"**             and GENKIT and TRANSKIT                    **"
"**                 version 8-5                            **"
"**         Center for Machine Translation                 **"
"**           Carnegie Mellon University                   **"
"**     (c) 1986, 1987, 1988 All rights reserved           **"
"**                                                        **"
"** Type (HELP) for more GLR description.                  **"
"** Type (GLR-ACKNOWLEDGE) for GRL acknowledgements.       **"
"** Type (MEANA-HELP) for MEANA-MT Description.            **"
"**                                                        **"
"************************************************************"

"THERE ARE TWO MENUS FOR RUNNING MEANA MT SYSTEM"
```

```
"*********************************************************"

"Type (MEANA-MT) FOR LOADING MEANA MT SYSTEM"
"Type (MEANA-LEX) FOR RUNNING THE ARABIC COMPUTATIONAL LEXICON"
; Loading /export/home/ahmed/mt/ArabMT/menu.lisp.

<cl> (meana-mt)
 LOADING MEANA'S PARSER, TRANSFER AND THE GENERATOR GRAMMAR
 PLEASE WAIT...

; Loading /export/home/ahmed/mt/ArabMT/openfile.lisp.
  Loading File - meana.tab
; Loading /export/home/ahmed/mt/ArabMT/meana.tab.
  Loading File - meana.fun
; Loading /export/home/ahmed/mt/ArabMT/meana.fun.
  Parser Ready for meana
; Loading /export/home/ahmed/mt/ArabMT/transfer.lisp.
***** Start Reading generat.gra
***** End Reading generat.gra
***** Start Writing "generat_gen.lisp"
***** End Writing "generat_gen.lisp"
***** Start Loading "generat_gen.lisp"
; Loading /export/home/ahmed/mt/ArabMT/generat_gen.lisp.
***** End Loading "generat_gen.lisp"

_" MEANA-Machine Translation is READY NOW"

<cl> (meana-help)
"
MEANA's Functions:
(P "word or sentence" )  - Parse a word or sentence.
(G "sentence" )          - For parsing an English sentence or
                           Acknowledgements and generating the Arabic
                           sentence.

(GG '(sentence $))       - For parsing an English Date, Address, Names,
                           and the Telephone Number and generating the
                           Arabic sentence.

(DMODE n)                - n=1: show symbols parsed
                         - n=2: and show rules applied.
```

```
(setq *trace-rules* t)    - Shows the Arabic generated words with the rules.
(MEANA-MT)                - For loading meana mt system.
(MEANA-LEX)               - For running the arabic computational lexicon.
"


<cl> (meana-lex)

MENU FOR RUNNING THE ARABIC COMPUTATIONAL SYSTEM

*********************************************
1- NOUN INFLECTIONS (SINGULAR, DUAL AND PLURAL FORM).

2- NOUN'S PRONOUNS INFLECTIONS.

3- VERB'S INFLECTIONS.

4- VERB AND PRONOUN  INFLECTIONS.

5- THE OTHER INFLECTIONS such as "ann and its Sister", Demonstrator
   articles, Sperate Pronouns, "Kana" Perfect Tense Grammar.

YOUR CHOOSE?
```

Appendix K

THE ARABIC SCRIPT TO THE CHILDREN'S STORY

Below is the output which has been generated by MEANA MT to the children's story in Arabic script. The English and the Arabic F-Structure is too big to be attached. شُكرُ فَأرَةٍ *šukru faʾratin*

kāna hunāka ʾasadun nāʾimāʾn ʿalā ḥašāʾiši ʾifriqiyyati ʿindamā ǧāʾa-t fāratun

كَانَ هُنَاكَ أَسَدٌ نَائِمَاً عَلَى حَشَائِشِ إِفرِقِيَّةٍ عِندَمَا جَاءَت فَارَةٌ ـ

ṣudfatan.

صُدفَةً.

biālaḥaṭāʾi kānat labuwut asadin ʿalā 'l-ḥašāʾiš fa-taʿattarat al-faʾrat ḥwlahā.

بِالخَطَاءِ كَانَت لَبُوَة اسَدٍ عَلَى الحَشَائِش فَتَعَثَّرَت الفَأرَة حولَهَا.

man ʾazʿağ nawmiy.

مَن أَزعَج نَومِي.

ṯumma ḥaṣara 'l-fārata biāḥdā maḫālibayhi 'l-ḍaḫmati.

ثُمَّ حَصَرَ الفَارَةَ بِاحدَا مَخَالِبِيهِ الضَخمَةِ.

māḏā ʾinnahā laysat siwā fāratin.

مَاذَا إِنَّهَا لَيسَت سِوَى فَارَةٍ.

ʾanti biālẓabṭ mā ʾarġ-bhu liġaḏāʾī.

أَنتِ بِالظَبط مَا أَرغبهُ لِغَذَائِي.

bakat al-fāratu wa ṭabbaqat ya-daḥā lita-tawwasala 'l-raḥmat.

بَكَت الفَارَةُ وَ طَبَّقَت يَدَهَا لِتَتَوَسَلَ الرَحمَة.

sayyid al-asad raǧāʾun waffer ḥyāty.

سَيِّد الاسَد رَجَاءٌ وَفِّر حيَاتِي.

ʾṭfālī 'l-sabʿat yantaẓirawun ruǧuwʿiy.

أَطفَالِي السَبعَة يَنتَظِرَوُن رُجُوعِي.

ʾanti ʾum lisabʿatin.

أَنتِ أُم لِسَبعَةٍ.

naʿam ʾanā.

نَعَم أَنَا.

fāratun ṣaġyiraṯun miṯliy laā ʾakwwun kāfiyatan ḥattā turḍay ǧuwʿaka.

فَارَةٌ صَغِيرَةٌ مِثلِي لَا أَكوُن كَافِيَةً حَتَّى تُرضِي جُوعَكَ.

ʾiḏā ʾanta ta-takarramu wa tuwaffera ḥayātiyi.

إِذَا أَنتَ تَتَكَرَّمُ وَ تُوَفِّرَ حَيَاتِي.

anā 'qasamu 'n ʾusaddada laka biduwni taʾaḫiyri.

أَنَا اقسَمُ ان أُسَدَّدَ لَكَ بِدُونِ تَأَخِيرِ.

Bibliography

[AH72] Ahmed K. E. Abdel-Hamid. *A transfer Grammar of English and Arabic.* A Ph.D thesis, University Of Texas at Austin, 1972.

[Als83] Abdulaziz I. Alsweel. *Word order in standard Arabic A lexical approach.* A Ph.D thesis, University Of Washington, 1983.

[AM95] Ahmed H. Alneami and Jim McGregor. Morphological auto-Analyser and Generator for Arabic Nouns. *Processing Arabic*, Nijmegen University, Holland. Report 8:page:39, 1995.

[AM96a] Ahmed H. Alneami and Jim McGregor. AN ARABIC COMPUTATIONAL LEXICON. *5th International Conference and Exhibition on Multi-lingual Computing (Arabic and Roman Script). ICEMCO 96*, Cambridge, 11-13 April 1996, 1996.

[AM96b] Ahmed H. Alneami and Jim McGregor. A Computational Analysis and Generation of Arabic Verb "to be" Verb "Not to be" and Pronouns. *(This paper was accepted in ACLIC/PacFoCol 1994 Conf, but was not presented due to lack of funding).*, Department of Computer Science, Sheffield University, Internal Research Report Code: CS-96-03 1996., 1996.

[AU72] A. V. Aho and J. D. Ullman. *The theory of parsing translation and compiling.* Prentice Hall, Englewood cliffs, NJ, 1972.

[AU77] A. V. Aho and J. D. Ullman. *Principle of Compiler Design.* Addison Wesley, 1977.

[Bee90] K. Beesley. Finite-State Descriptions of Arabic Morphology. *In Proceedings of the second Conference on Bilingual Computing in Arabic and English*, Literary and Linguistic Computing Center, Cambridge University, 1990.

[BGQA87] Ch. Boitet, P. Guillaume, and Quezel-Ambrunaz. A case study in software evolution: from ARIANE-78.4 to ARIANE-85. *In Proceedings of the Conference on Theoretical and Methodological Issues in Machine Translation of Natural Language,*, Colgate University, Hamilton, NY, Augest 1985:27–58, 1987.

[BTS89] K. Beesley, Buckwalter T., and Newton S. Two-Level Finite-State Analysis of Arabic. *In Proceedings of the first Conference on Bilingual Computing in Arabic and English*, Literary and Linguistic Computing Center, Cambridge University, 1989.

[DeR69] F. L. DeRemer. *Practical Translators for LR(k) Languages.* Ph.D dissertation, M. I. T., Cambridge, Mass., 1969.

[DeR71] F. L. DeRemer. "Simple LR(k) grammars". *Comm. ACM*, 14:7:453–460, 1971.

[Dit92] Everhard Ditters. *A formal Approach to Arabic Syntax, The Noun phrase and the verb phrase.* LUXOR, Netherland, Nijmegen, 1992.

[EC90] M. M. Ebrahim and J. D. Clark. English to Arabic Machine Translation of Figurative Expression. *In proceedings of the First Conference on Bilingual Computing in Arabic and English, Cambridge*, session 2A:12p (no running page numbers), 1990.

[ECF89] M. M. Ebrahim, J. D. Clark, and A.A Fahmy. Arabic in Matchine Translation. *In proceedings of the First Conference on Bilingual Computing in Arabic and English, Cambridge*, session 2A:12p (no running page numbers), 1989.

[ESH89] T. El-Sadany and M. A. Hashish. An Arabic Morphological System. *IBM System Journal*, pages 20–30, 1989.

[Ewe85] O. K. Ewell. Computer Aided Translation Design and implementation. *In Studies in Machine Translation proceeding*

of the International workshop on Computer Aided Translation, Organized by the King Abdulaziz city for Science and Technology in Riayah, Ed. by A-razzak A-wahab:143–149, 1985.

[Far89] M. Farhat. Issues in the Development of a comprehensive Arabic Lexical Database. *In proceedings of the First Conference on Bilingual Computing in Arabic and English*, Literary and Linguistic Computing Center, Cambridge University, session 2A:13p (no running page numbers), 1989.

[FF90] A. Feddag and E. Foxley. A syntactic And Morphological Analyser for Arabic Words. *In Proceedings of the second Conference on Bilingual Computing in Arabic and English*, Literary and Linguistic Computing Center, Cambridge University, 1990.

[FF93] A. Feddag and E. Foxley. A Lexical Analyser for Arabic. *International journal of man-machine studies*, 38:2:313–330, 1993.

[GEGA85] G. Gazdar, Klein E., Pullum G.K, and Sag A. *Generalized Phrase Structure Grammar.* Blackwell, 1985.

[HN62] J.A Haywood and H.M. Nahmad. *A new Arabic Grammar of the Written Language.* Percy Lund. Humphries and Co. Ltd, London, Bradford, 1962.

[Hov88] Edward Hovy. *Generating Natural Language Under Pragmatic Constraints.* Hillslate, NJ, 1988.

[Hut89] W. J. Hutchins. *Machine Translation Past, Present, Future.* Ellis Horwood Series in Computers and their Applications, 1989.

[Jac81] R Jackendoff. On Kataz's autonomous semntics. *Language*, 57 (2):425–435, 1981.

[Kar83] Lauri Karttunen. A general Morphological Processor. *Texas Linguistic Forum no. 22 ed. by Dalryple et al*, Austin:Linguistics Department, University of Texas:165–186, 1983.

[Kat80] J. J. Kataz. Chomsky on meaning. *Language*, 56 (1):1–41, 1980.

[Kay85] M. Kay. Parsing in Functional Unification Grammar. *Natural Language Proceeding:Psychological, Computational and Theortical Prespectives*, Cambridge University press, 1985.

[KB82] R. Kaplan and J. Bresnan. *The Mental Representation of Grammatical Relations.* MIT Press, Cambridge, MA, 1982.

[KL82] R. Kittredge and J Lehrberger. *Sublanguage :studies of language in Restricted Semantic Domains.* DeGruyter, 1982.

[Kos71] Kees Koster. Affix Grammars. *In Algol 68 implementation (ed) Peck, J. Amsterdam:north-Holland*, pages 95–105, 1971.

[Kos83] Kimmo. Koskenniemi. Two-level Model for Morphological Analysis. *In Proceedings of IJCAI*, pages 683–685, 1983.

[Kos84] Kimmo. Koskenniemi. A general Computational Model for Word-form Recognition and Production. *In Proceedings of the International Conference on Computational Linguistics (COLING84)*, pages 178–181, 1984.

[Lag93] Klaus Lagally. ArabTex Report. *Insititut Fuer Informatik Universitaet Stuttgart Breitwiesenstrasse 20-22 W-7000 Stuttgart 80 GERMANY*, 1993.

[LBS85] W. P. Lehmann, W.S. Bennett, and J. et al Slocum. The METAL System. *Final Technical Report RADC-TR-80-374, Rome Air Development Center, Griffiss AFB, New York, January 1981.*, Available as Report AO-97896, National Technical Information Service, U.S. Department of Commerce, Springfield, Va., 1985.

[LT93] A. Lavie and Masaru. Tomita. GLR* - An Efficient Noise-skipping Parsing Algorithm for Context-free Grammars. *In proceedings of Third International Workshop on Parsing Technologies (IWPT-93)*, Tiburg, The Netherlands, August 1993:123–134, 1993.

[Maa87] H. D. Maas. The MT system SUSY. *In King, M. (ed.) Machine Translation Today: the state of Art: Proceedings of the Third Lugano Tutorial, Lugano, Switzerland,*, 2-7 April 1984. Edinburgh University Press:209–246, 1987.

[Meh86] Saad Abdul-sattar Mehdi. Arabic Language Parser. *In the International Journal man-Machine studies*, 25:593–611, 1986.

[Meh87] Saad Abdul-sattar Mehdi. *Computer Interpretation of Arabic.* A Ph.D thesis, University Of Exeter, 1987.

[Mei90] Hans. Meijer. The project on Extended Affix Grammars at Nijmegen. *In Attribute Grammars and their Applications SLNC*, 461:130–142, 1990.

[Mou88] A. H. Moussa. Computer Application to Arabic Roots and Arabic Words. *Applied Arabic Linguistics and Signal and Information Processing*, pages 213–215, 1988.

[NCTG92] Sergi Nirenburg, Jaime Carbonell, Masaru Tomita, and Kenneth Goodman. *Machine Translation A knowledge-Based Approach.* Morgan Kaufmann Publishers, San Mateo, California, Chapter 5, P 117, 1992.

[NH92] A. Narayanan and L. Hashem. A three-Level Finite-state Model for Arabic Morphology. *In proceeddings of the 3rd International Conference and Exhibition on Multi-Lingual Computing. (ARABIC and Roman Script)*, The Centre for Middle Eastern and Islamic studies, University of Durham, 1992.

[Roo89] P. Roochnik. A morphosyntactic parser for Arabic. *In proceedings of the First Conference on Bilingual Computing in Arabic and English*, Literary and Linguistic Computing Center, Cambridge University, session 2A:13p (no running page numbers), 1989.

[SA85] Jonathan Slocum and Authony Aristar. The Treatment of Grammatical Categories and Word Order in Matchine Translation. *In Studies in Machine Translation proceeding of the International workshop on Computer Aided Translation*, Organized by the King Abdulaziz city for Science and Technology in Riayah, Ed. by A-razzak A-wahab:155–168, 1985.

[Sat62] Arnold C Satterthwait. *Parallel Sentence construction Grammar of Arabic and English.* A ph thesis, Harvard University, 1962.

[Sha89] Patrick Shann. The selection of a parsing strategy for On-Line Machine Translation System in a sub-language domain. A new practical comparison. *In Proceedings of International Workshop on Parsing Technologies, Carnegie-Mellon Univ*, 1989.

[Sno65] James. A. Snow. *A Grammar of Modern Written Arabic Clauses.* A ph thesis, University of Michigan, 1965.

[SY90] Hussein I. Shahein and Sameh A. Yousef. A model for Arabic Morphological As Production System. *In Proceedings of the second Conference on Bilingual Computing in Arabic and English*, Literary and Linguistic Computing Center, Cambridge University, 1990.

[TK88] M. Tomita and Knight K. Pseudo Unification and Full Unification. *Technical Report, Unpublished*, CMC-CMT-88-MEMO, 1988.

[TM88] Wendy Thomson and Chris Martin. *An Introduction to Text Processing with LaTex.* Computing Services, University of Sheffield, MW3/10/88, 1988.

[TMMK88] Masaru. Tomita, T. Mitamura, H. Musha, and M. Kee. The Generalized LR Parser/Compiler. *Version 8.1: User's Guide*, Technical Report CMC-CMT-88-MEMO:20 April, 1988.

[Tom86] Masaru. Tomita. *Efficient Parsing for Natural Languge.* Kluwer Academic Publisher, Boston, 1986.

[Tra85] S. Traboulsi. SYSTRAN System. *In Studies in Machine Translation proceeding of the International workshop on Computer Aided Translation*, Organized by the King Abdulaziz city for Science and Technology in Riayah, Ed. by A-razzak A-wahab:133–140, 1985.

[Vaq85] B. Vaquuois. The Approach of Geta to Automatic Translation Comparison with some other methods. *In Studies in Machine Translation proceeding of the International workshop on Computer Aided Translation*, Organized by the King Abdulaziz city for Science and Technology in Riayah, Ed. by A-razzak A-wahab:29–96, 1985.

[Wea49] **W. Weaver. Translation. *Unpublished Memorandum*, New york, 1949.**