A Portable Natural Language Interface from Arabic to SQL

May, 1999

Badr Al-Johar

Submitted in accordance with the requirements
for the degree of Doctor of Philosophy to:

Department of Computer Science
University of Sheffield

Abstract

In recent years, natural language interface systems have been built based on the Front End and the Back End architecture which gives a guarantee of modularity and portability to the system as a whole. An Arabic Front End has been built that takes an input sentence, producing syntactic and semantic representations, which it maps into First Order Logic. Expressing the meaning of the user's question in terms of high level world concepts makes the natural language interface independent of the database structure. It is then easier to port the interface Front End to a database for a different domain.

The syntactic treatments are based on Generalised Phrase Structure Grammar (GPSG) whereas the semantics are expressed in formal semantics theory. The focus is mainly to provide syntactic and semantic analyses for Arabic queries based on correct Arabic linguistic principles. The proposed treatments are proved and tested by building a prototype system. The prototype is implemented using one of the existing systems called Squirrel.

An Arabic morphological analyser is also proposed and implemented to distinguish between two types of morphemes: internal morphemes which are a part of the word's pattern, and external morphemes which are independent words attached to the word but which are not part of the word's pattern. So, the system focuses on the extraction of morphemes from the various inflexions or forms of any Arabic word.

Acknowledgements

I would like to express my sincere gratitude to my supervisor Jim McGregor for supervising this work and providing helpful suggestions throughout the period of this study. I am also grateful to the other members of my panel, Prof. Mike Holcombe and Dr. Robert Gaizauskas for their valuable guidance and suggestions.

I am grateful to Anne De Roeck, the Head of Computer Science Department at University of Essex, for allowing us to use Squirrel. Thank you to John Carson, University of Essex, for his technical support during Squirrel installation.

Many thanks to all members of the NLP group at the University of Sheffield especially Dr. Ahmed Alneami, Dr. Saliha Azzam, Dr. Kevin Humphreys, and Mohammad Harmain for their helpful discussion and comments.

Also, thanks to Dr. Saad Aljabri of the University of Edinburgh, Dr. Mohammad Khayat of the King Abdulaziz University and Husni Almuhtaseb of the King Fahd University of Petroleum and Minerals for their helpful suggestions at the early stage of this work.

Dedications

This work is dedicated to my family here and there, especially my mother, my father, my wife and my children Abdulaziz, Rawan, and the little one Abdullatif. My warmest thanks must go to them as they have been so patient with me all this time when I was neglecting my family duties. They have given me nothing but love and encouragement which I hope I can make up to them.

Abbreviations

The following are codes and abbreviations used in the thesis.

ACC	accusative
AGR	agreement
AP	Adjective Phrase
CAP	Control Agreement Principle
COM	comment
COP	copular
DRC	Domain Relational Calculus
`fem`	feminin
FOL	First Order Logic
FSD	Feature Specification Default
Gen	Genitive
GPSG	Generalized Phrase Structure Grammar
HFC	Head Feature Convention
ID	Immediate Dominance
IntArt	Interrogative Article
`intrans`	intransitive
LP	Linear Precedence
`masc`	masculine
N	Noun
neg	negative
NLI	Natural Language Interface

NLP	Natural Language Processing
NOM	nominative
NP	Noun Phrase
obj	object
P	Preposition
PP	Preposition Phrase
PT	Property Theory
S	Sentence
Ś	pronounced 'Sentence-bar'
subj	subject
TOP	topic
`trans`	transitive
TRC	Tuple Relational Calculus
URC	Untyped Relational Calculus
V	Verb
VP	Verb Phrase

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Several computer-user interface styles have been identified and menu selection is the most widely accepted interface style. Unfortunately, it cannot allow the user to access any information that lies outside the scope of the predetermined options. That creates a need to focus on making the computer understand natural language and interact with users on that basis.

In recent years development of Natural Language Interfaces (NLIs) to database systems has been one of the most important areas in Natural Language Processing (NLP). It has long been recognised as a useful application of NLP techniques. For casual users, communication with databases in natural language is considered a convenient and easy method of data access.

A NLI to computer database systems provides users with the ability to obtain data stored in the databases by allowing them to query the systems in their natural language. In communicating with computer systems via a natural language, users can frame queries or statements in the way they visualise the data being discussed. They do not have to know how the data is stored or processed by a computer system [Grosz et al. 1987].

The main purpose of building an NLI in front of a database system is to trans-

late the input from some natural language into an internal representation in a database query language. This task is not as easy as it appears, as after a strong effort, few systems which purport to allow querying of a database in English (or other natural languages) are available commercially. Also, even with these systems the problem of completely unrestricted use of natural language queries has not been solved yet [Androutsopoulos et al. 1995].

1.1 Advantages v. Disadvantages of NLI

Allowing users to communicate with a computer through a natural language is an attractive idea. It has some advantages over the other interface styles but at the same time it does suffer from some disadvantages. The advantages and disadvantages are discussed, in the following subsections, by comparing the NLI style to other interfaces such as formal query languages, menu-based interfaces, and graphical interfaces.

Although interfaces using input of natural language via a keyboard are not likely to be hugely practical in the long term, they are still a useful testbed for developing the necessary techniques, such as semantic interpretation, which are important for other style interfaces like spoken systems [Androutsopoulos et al. 1995].

1.1.1 Advantages

There are some advantages of using NLIs to database systems over other kinds of interfaces. First of all, users of NLIs are not required to learn an artificial communication language. Formal query languages are difficult for non-computer-specialists to learn. Graphical interfaces and form-based interfaces are easier for non-computer-specialist users but, still, users need to be trained about how to invoke forms, link frames, select restrictions from menus, and so

on. An ideal NLI to a database system would allow queries to be formulated in the user's native language [Androutsopoulos et al. 1995]. In other words, NLIs allow the same information to be requested in a variety of ways. That means there would be no need for the users to spend time learning the system's communication language.

The advantage of building natural language interfaces in front of databases is not only to avoid the use of artificial languages (such as SQL) by the end users. It can also hide the structure of the database from the users. NLI users should not have to be aware of the database schema although they may deliberately query about the database structure, for instance:

> Is any information kept about employee's children?

In addition, there are kinds of questions that can be easily expressed in natural language which seem tedious to express using graphical or database query languages like SQL. Queries like:

> Which department has no engineers?
>
> Which company supplies every department?

can be easily expressed in natural language, but they would be difficult, at least for casual users, to express in most other kind of interfaces (e.g. graphical or form-based interfaces) [Cohen 1991].

Furthermore, some NLI systems provide the ability to use anaphoric reference, i.e. reference to a previous word or phrase, and elliptical[1] constructions across sentence sequences. The users of this kind of system are allowed to use very brief, under-specified questions, where the meaning of each question is complemented by the discourse context. The following are examples of queries obtained by EUFID [Templeton and Burger 1983].

[1]Ellipsis is leaving something unsaid which will be understood by the listener [Shapiro 1992].

What programmers know Prolog and C++?

Which of them live in California?

In Nevada?

How many know Pascal?

The notion of discourse context[2] is usually not supported in graphical interfaces, form-based interfaces, and formal query languages [Androutsopoulos et al. 1995].

Finally, in an experimental study of using NLIs to database systems for managers by [Morick 1984], the managers frequently perceived better substantiated decision making and planning as a benefit. Managers also recognised that there are benefits in allowing greater access to computers. NLIs to databases would allow them to obtain data that correspond to their way of looking at things. In other words NLIs would put control back into their own hands instead of depending on computer specialists.

The way information is presented significantly affects the performance of the users. Therefore, the use of NLIs will help system programmers to provide adaptive interfaces geared to the level of the current user so he/she is more productive [Morick 1984].

1.1.2 Disadvantages

Implementation of successful NLIs requires a detailed knowledge (i.e. lexical, syntactic, semantic, anaphoric references, common sense, scope of quantifiers, world model and so on) of the natural language being used which is very difficult to achieve.

[2]It means that the syntactic and semantic representations of sentences in discourse contexts relate to the representation of other sentences in the discourse [Shapiro 1992].

Also, the use of English, for example, as a natural language in a specific domain (e.g. geology, chemistry, etc.) may be more specialised (in terms of structure and meaning) than in the everyday use of English [Burton 1991]. These reasons make the idea of having a NLI system for a global language a dream and bring the idea of treating a limited subset of natural language or a so-called sublanguage. One may argue that a subset of natural language is no longer a natural language. However, current NLIs can only cope with a sublanguage. That means the linguistic capabilities of the systems are not obvious to the user. Maybe it is difficult for the user to understand or remember what kinds of questions an NLI can or cannot cope with.

> Q1) What is the ocean that borders African countries and that borders Asian countries?
>
> Q2) What are the capitals of the countries boarding the Baltic and France?

In the above examples, taken from the MASQUE system [Androutsopoulos 1992], the user will assume that the system can handle all kinds of conjunction queries, because the system is able to understand a certain conjunction query like Q1, while in fact it cannot answer most of them, question Q2 for instance. Failure to answer a particular query, by contrast, can lead the user to assume that similar queries cannot be answered, while in fact they can be answered [Androutsopoulos et al. 1995].

Kaplan [1983] pointed out, as a conclusion of his experiment on co-operative responses in a NLI system, that it is not the case that NLIs are desirable for all users wishing to query a database. An NLI is an appropriate vehicle for a user who is unclear about what data he/she wants or does not know what data is present in the database. It has the ability to express vague queries. Also, indirect responses to natural language queries can help the user to locate relevant information. On the other hand, users with specific or detailed needs may find NLIs too imprecise for expressing their questions. A formal query language may be more effective for them.

Another disadvantage is the understanding of the cause of failures. It is often not clear to the user, when the NLI systems cannot understand a query, whether the rejected question is outside the system's database coverage, or it is outside the system's linguistic coverage. That is why users sometimes try to rephrase a query referring to concepts the system does not know, because they think the problem is caused by the system's limited linguistic coverage. Similarly, because users do not think an alternative phrasing of the same query could be answered, they do not try to rephrase queries the system could conceptually handle [Androutsopoulos et al. 1995].

1.2 Research Motivations and Aims

Understanding Arabic requires treatments of all the following levels: morphology, syntax, and semantics. Most of the research on natural language processing for the Arabic language has concentrated on morphological analysis and little research treats the other computational linguistic levels [Khayat 1996].

Over the last thirty years, research in natural language processing for English and other Indo-European languages has resulted in several theories for parsing natural language. An inflectional language like Arabic, which has a different word order, needs a powerful theory to analyse its sentences both syntactically and semantically.

Although several syntactic studies have tried to apply modern linguistic theories, such as Lexical Functional Grammar (LFG) and Definite Clause Grammar (DCG), to Arabic, there is a lack of awareness about whether the syntactic analyses of these studies were based on the principles developed by Arabic linguists.

Before adopting a grammar formalism for processing any natural language, several criteria should be considered. The most important criterion is linguistic felicity. This means that the formalism should analyse statements of

the language based on sound linguistic theories [Shieber 1986].

This study uses Generalised Phrase Structure Grammar (GPSG) [Gazdar et al. 1985], which is a powerful grammar theory, to analyse Arabic statements. The focus is to provide the syntactic analyses that are acceptable to linguists.

In addition, the logical semantics of Arabic sentences, based on reviews of previous work see Chapter 4, has received very little work with no indication of what the major problems are and how they can be tackled [Al-Johar and McGregor 1997].

Therefore, this work also aims to propose well-formed expressions for Arabic statements based on model-theoretic semantics. The Arab grammarians' theories (إِسنَاد 'isnAd) and (العَامل Al'Aml)[3] will be used as a guide to assure validity of the treatment for Arabic.

The most characteristic feature of the Arabic language is that it is a highly inflected language. A token may contain verb, subject, and object as one word which makes the morphological structure of the Arabic language very complicated. There are a number of Arabic morphological systems that have been proposed but they tend to go up to the basic root level of the word which can cause a problem with the word semantics.

Thus, one of the aims of this work is to build a domain independent morpholgical analyser which can be ported to any other Arabic natural language processing system. The high degree of portability is achieved by blocking the use of knowledge bases, and the ability to treat unknown words. The system has the ability to treat the inflected Arabic proper nouns without having a database of proper nouns.

Finally, the architecture of recent NLI systems, such as CLARE [Pulman et al. 1993] and LOLITA [Smith et al. 1994], is such that a natural language question is transformed into an intermediate logical query. Then the logical query is

[3]These theories are discussed explicitly in Abn-Hesham [1985] and Hassn [1975].

translated to an expression in the database's query language, and evaluated against the database. Systems based on this approach make the linguistic Front End, which generates the logic queries, independent of the underlying DataBase Management System (DBMS). Thus, the natural language front end can be ported to a different DBMS, by customising the translator module which is the Back End. Also, this approach gives the flexibility of applying the system to another language by replacing the Front End with one for the new language.

Therefore, this study will focus on building an Arabic natural language interface to database systems using this approach.

1.3 Thesis Structure

This thesis comprises a further eleven chapters followed by appendices.

Chapter two discusses the natural language interface (NLI) to database systems. This chapter focus on natural language interface system architectures. There are two types of systems: domain specific systems and domain transportable systems. It reviews some of the well-known systems of each type. The problems of NLIs to databases are also discussed. A cross domain portability is given special attention here as it is one of the main features of the current work.

Chapter three provides a general outline of the system domain language which is Arabic. This chapter aims to describe the syntax of the Arabic Language and its characteristics and features. The description is based on the Arab grammarians' viewpoints. Also, it discusses the possible word order for Arabic and how Arabic is written nowadays.

Chapter four reviews existing Arabic natural language interface systems. It will also present an evaluation of some of them based on the information available

in their published documents.

Chapter five presents the proposed syntactic analysis of Arabic sentences using Generalised Phrase Structure Grammar (GPSG) Theory.

Chapter six discusses the idea of building a logical meaning representation for Arabic statements. The approach is to propose well-formed expressions for Arabic statements based on model-theoretic semantics. The Arab grammarians' theories (إِسنَاد 'isnAd) and (العَامل Al'Aml), see [Abn-Hesham 1985] and [Hassn 1975], were used as a guide to assure validity of the treatment for Arabic. This chapter classifies Arabic sentences into: verbless sentences, verbal sentences, and sentences of individuals. Each group receives a specific semantic treatment.

Chapter seven gives a presentation of the original Squirrel system, a NLI to database systems for English, which has been chosen as a basis for this work. It also describes the character set used for Arabic and discusses the example domain which is used. Finally, it discusses the graphical user interface which is constructed to improve the comunication between users and the system during the testing stage.

Chapter eight presents the morphological analyser that is implemented and is attached to the Arabic version of the Squirrel system. The analyser is based on the review and evaluation of previous algorithms that were discussed. Each module of the analyser is described separately. Examples of the input and the output of this analyser are given.

Chapter nine discusses the implementation issues of the syntactic and semantic grammar proposed previously in Chapters 5 and 6. It also discusses how grammar and lexicon compilers are used to generate the Arabic grammar rules and lexicon in Prolog readable form. The natural language coverage of this system is presented towards the end of the chapter.

Chapter ten presents the customisation process for Squirrel's back end to en-

able it to handle Arabic queries. In addition, it discusses the extensions that have been made to enable it to handle queries for times, places, and counts which were not handled in the original Squirrel.

Chapter eleven gives an overview of the evaluation in the natural language interface area. It also proposes an evaluation methodology for our system and a discussion of the evaluation results is given. The chapter highlights existing problems in Squirrel.

Chapter twelve concludes the research as a whole and makes suggestions for further developments.

§§§§

Chapter 2

Natural Language Interfaces

The main purpose of building Natural Language Interfaces (NLIs) in front of database systems is to translate the input from some natural language into an internal representation in a database query language. This task is not as easy as it appears.

This chapter concentrates on the issues of NLIs, or equivalently Natural Language Front Ends (NLFE), to database systems. The discussion will cover NLI architectures, NLI systems, NLI problems, and Portability.

2.1 NLI Architectures

Natural language interfaces have used different architectures for their development. They can be classified based on the natural language analysis technique which has been used to translate the natural language query into a unique internal representation to find an answer.

Pattern matching is one of the natural language processing techniques used in some of the NLI systems. Savvy, the pioneer of the interface systems for micro-

computers, was developed based on a pattern-recognition technique [Johnson 1985]. LUNAR [Woods 1968], the best-known NLI system in the late sixties, was built on a Syntax-based approach where the query is syntactically analysed then the parse tree is mapped directly to an expression in some database query language.

Semantic grammar is used in LADDER [Hendrix et al. 1978]. Queries in semantic grammar are still handled by parsing the query and mapping the output parse tree to an expression in a database query language, but the difference is that the grammar's categories do not correspond necessarily to syntactic concepts.

In most recent NLI work, a natural language question is transformed into an intermediate logical query. The reason for this is to express the meaning of the user's question in terms of high level world concepts, which are independent of the database structure. Then the logical query is translated to an expression in the database's query language, and evaluated against the database. TEAM is one good example of this approach [Martin et al. 1983, Grosz et al. 1987].

These approaches are discussed in the next sections in more detail.

2.1.1 Pattern Matching Systems

The essence of the pattern matching approach is to interpret input queries as a whole rather than building up their interpretation by combining the structure and meaning of words. So, the interpretation is obtained by matching patterns of words against the input question. There is a list of correspondences between equivalence classes of input utterances and interpretations, which are the ones associated with each pattern.

In the pattern-matching approach, the system may use a rule like:

```
Pattern: "manager"   'department'
 Action: Report manager of row where
                    department = 'department'
```

The rule says that if a user's request contains the word "manager" followed by a department name, then the system should locate the row which contains the department name, and print the corresponding manager. This rule would allow the system to generate the same response for all of the following queries [Androutsopoulos et al. 1995]:

> Who is the manager of Marketing?
>
> Print the manager of Marketing.
>
> Could you please tell me who is the manager of Marketing?
>
> :
>
> etc.

Savvy is one of the early NLI systems that relied on pattern-recognition rather than parsing techniques in answering the user's queries. The system does not need to look for exact matches between the input string and the words in the lexicon, instead it recognises the input pattern by measuring its closeness to patterns it holds already. This makes it to jump to the wrong conclusion with equal ease, for instance in a query like:

TITLES FOR EMPLOYEES IN FLORIDA.

it responds by listing the states for all employees because it picks up "IN" as the code for Indiana, assuming the query is about states [Johnson 1985].

The main advantage of this approach is its simplicity. No elaborate parsing and interpretation modules are needed and the systems are easy to implement. Pattern-matching systems often manage to come up with some reasonable answer, even if the input is out of the range of sentences that the patterns were designed to handle, as can be seen in the following rule:

```
Pattern:  "Capital"   "country"
 Action:  Report Capital and Country of each row.
```

This would allow the system to answer the question:

Is it true that the capital of each country is Athens?

by listing the capital of each country, which can be considered as an indirect negative answer [Androutsopoulos et al. 1995]. However, the shallowness of the pattern-matching approach would often lead to bad failures, as can be seen in the above query *"TITLES FOR EMPLOYEES IN FLORIDA"*.

It has been reported that Savvy is, significantly, language independent. It can match inputs to a pre-defined string in any other language (e.g French, Spanish and so on) as easily as to English [Johnson 1985].

2.1.2 Syntax-based System

Syntax is concerned with the ways words can fit together to form higher level units such as phrases, clauses, and sentences. In this approach the interpretations of large groups of words are built up of the interpretations of their syntactic constituent words or phrases which is just the opposite of the pattern matching approach, which interprets the input as a whole. The emphasis here is to construct a complete syntactic analysis of the input query first and then to construct the internal representation of the query.

In a system using this approach, the user's query is analysed syntactically, and the resulting parse tree is directly mapped to an expression in some database query language. The grammar in syntax-based systems describes the possible syntactic structures of the user's questions.

```
S
    NP   Det   which
         N     sample
    VP   V     contain
         NP    Silicon
```

LUNAR [Woods 1968] is a typical example of this approach. The grammar above (an example from LUNAR system), says that a sentence(S) consists of a noun phrase (NP) followed by a verb phrase (VP). A noun phrase consists of a determiner (Det) followed by a noun (N) and so on. Using this grammar, a NLI could figure out the syntactic structure of the question *"Which samples contain Silicon?"*. The system could then use *pattern* $\longrightarrow$ *action* rules for transforming the syntactic representation of the input sentence to produce the meaning interpretation, as in the following database query (Z is a variable):

```
(for_every Z(is_sample Z)
(contains Z Silicon);
(print out Z))
```

which would then be evaluated by the underlying database system. Syntax-based NLIs usually interface to application-specific database systems, that provide database query languages carefully designed to facilitate the mapping from the parse tree to the database query.

2.1.3 Semantic Grammar Systems

In this approach, the techniques of answering users queries is still obtained by parsing the input query and then mapping the parse tree to an expression in a database query language. But the difference between the syntax-based approach and this approach is that in semantic grammars the categories used

are defined semantically as well as syntactically. Thus a semantic grammar might have the category "description of a ship" instead of the category "noun phrase" in a syntactic grammar.

Semantic grammars were introduced as an engineering methodology. They allow semantic knowledge to be easily included in the system. The goal is to eliminate the production of meaningless parses by setting up the grammar so that only meaningful parses can be generated. The construction of the full interpretation of the input query is achieved by combining the basic constituents via rules of a higher level.

The following is an example, taken from LIFER [Hendrix et al. 1978], of how semantic grammars can be used:

⟨present⟩ the ⟨attribute⟩ of ⟨ship⟩

⟨present⟩ ⟶ what is | [can you] tell me

⟨attribute⟩ ⟶ length | beam | class

⟨ship⟩ ⟶ the ⟨shipname⟩ | ⟨classname⟩ class ship

⟨shipname⟩ ⟶ kennedy | enterprise

⟨classname⟩ ⟶ kitty hawk | lafayette

As can be seen, the technique is strongly domain dependent where all objects and their relations in the domain of application should be categorised in advance. Systems using this approach can achieve a high performance for the specific domain for which they were developed [Barros 1995].

The disadvantages of this approach become apparent when the domain starts to grow past a certain point. It becomes difficult to specify all the acceptable relations in advance [Barros 1995]. Also, a new semantic grammar has to be written whenever the NLI is configured for a new domain. The reason for this is that the grammars built for this approach are very closely tied to the

database structure and to the requirements of straightforward translation to a database query.

A number of well known systems were built under this paradigm. LADDER [Hendrix et al. 1978] was developed using LIFER [Hendrix et al. 1978], a framework which is a tool for helping the development of natural language interfaces based on semantic grammar. Also, this approach was used in the EUFID system [Templeton and Burger 1983].

2.1.4 Intermediate Representation

This type of architecture, corresponding to the current state of the art, separates the query understanding process into two major steps [Binot 1991]:

> In a first step, the natural language input will be processed by a domain independent analyser to produce the meaning of the input query which is expressed in an intermediate meaning representation formalism.
>
> In the second step, the intermediate meaning representation of the input query will be translated by a domain dependent interpreter to produce the equivalent database query to access the database and retrieve the corresponding answer.

Figure 2.1 illustrates the architecture of Squirrel [DeRoeck et al. 1991] as an example of this approach. The natural language input is first processed syntactically by the parser. The parser generates a parse tree after it consults a set of syntax rules. The parse tree is then transformed to an intermediate logical query, in terms of Property Theory (PT) representation, by the semantic interpreter. The Truth Predicate module is used to traslate the PT representation to a First Order Logic (FOL) representation. The syntax/semantics rules in modern systems are becoming increasingly influenced by principled linguistic

theories and they are often expressed in variations of well-known formalisms. The back end is responsible for producing the database query.

Some systems based on this approach use a world model which describes the structure of the surrounding world. It contains a hierarchy of classes of world objects and constraints on the types of arguments each logical predicate may have. Figure 2.2 shows a world model hierarchy for a business application. Similar hierarchies are used in TEAM [Martin et al. 1983] and CLARE [Pulman et al. 1993].

The mapping to database information specifies how logic predicates relate to database objects. In the case of an interface to a relational database, the simplest approach would be to link each logic predicate to an SQL statement.

The advantage of NLIs based on this approach is that the linguistic front-end, which generates the logic queries, is independent of the underlying database management system (DBMS). Thus, the NLI to a database system can be ported to a different DBMS, by rewriting the translator module. Also, this approach allows the use of generic linguistic front-ends such as CLE [Alshawi et al. 1988] and LOLITA [Smith et al. 1994], as a part of NLI systems.

2.2 NLI Systems

Wu and Dilts [Wu and Dilts 1992] characterised the previous work in NLIs in terms of how knowledge about linguistics, domain, and discourse is structured, NLIs can be classified as two types of systems:

Domain specific systems are characterised by either merging discourse with database knowledge, or merging linguistics with domain knowledge in their semantic grammars. Semantic-grammar-based systems tend to be easily built for small domains and can be very efficient.

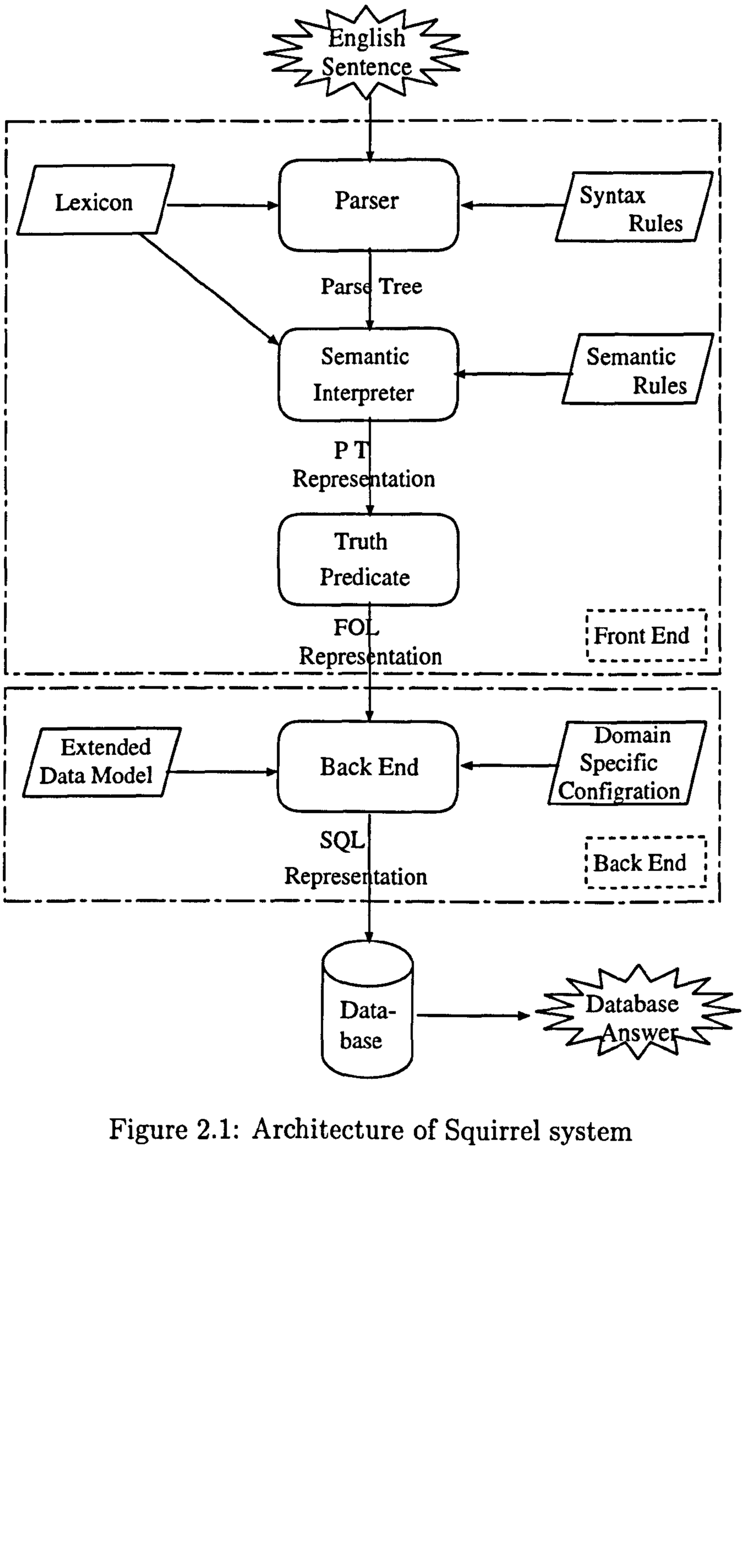

Figure 2.1: Architecture of Squirrel system

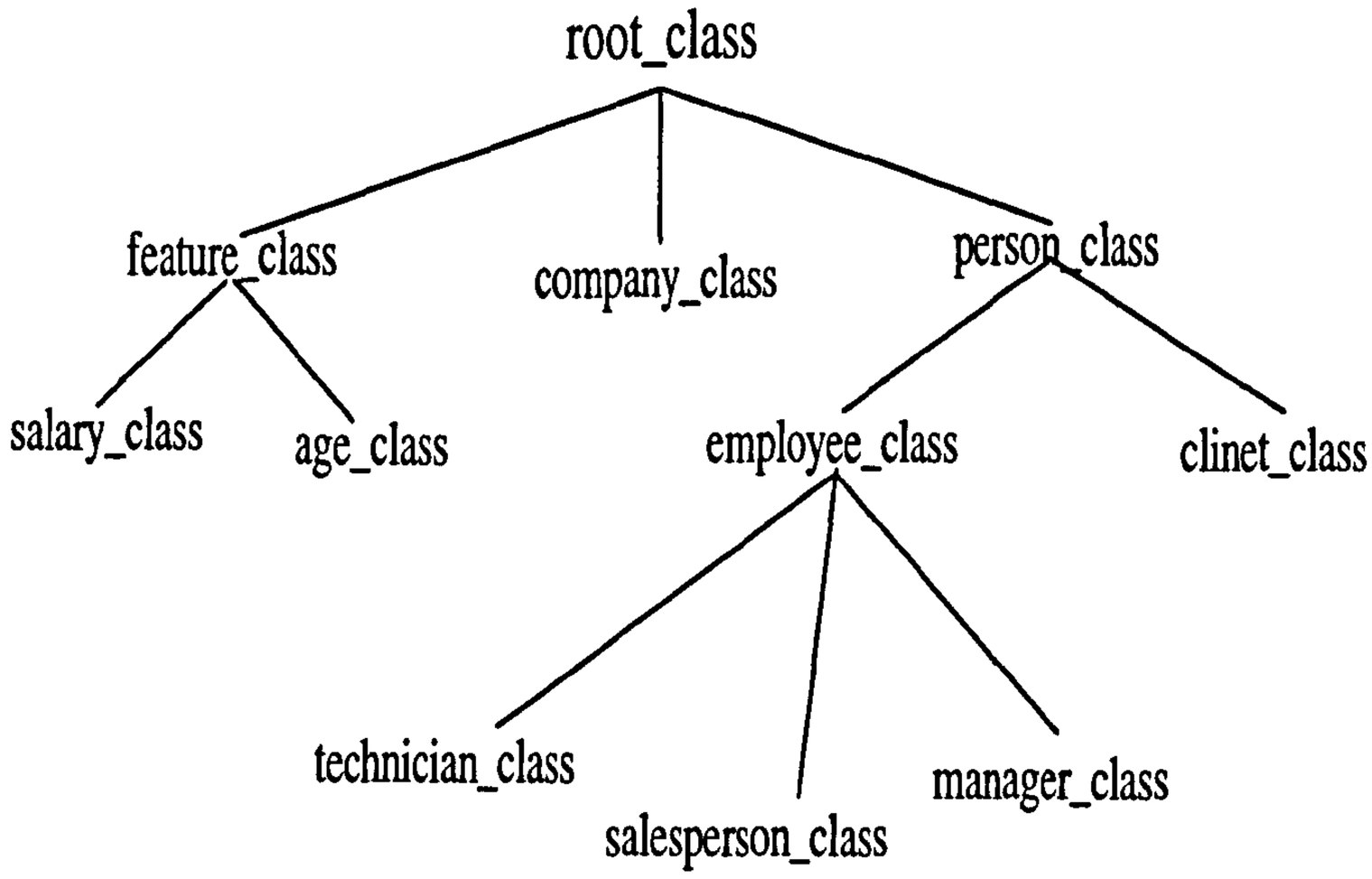

Figure 2.2: Structure of a world model

Domain transportable systems separate various levels of structure, using a generic syntactic analyser with added domain semantics and translating the NL query into intermediate representation language before generating target DBMS-specific queries. This type of system is easily transferred from one domain to another but at a cost of low efficiency.

Although current research systems try to integrate natural language with other media such as graphics, voices, visual, and hyper-media, that classification is still valid for the systems which accept queries via the keyboard.

The following are some milestone systems categorised on the basis of the above classification.

2.2.1 Domain Specific Systems

The history of natural language interface research goes back to the late sixties. The early systems were mainly concerned with interpreting single sentences in natural language into a database query for a specific domain.

Because of the restricted size of the domain which limits the number of possible ambiguities as well as the amount and variety of knowledge required by the understanding process, this kind of system is more efficient than the domain transportable systems in resolving ambiguities. The following are examples of systems based on this approach.

LUNAR

The LUNAR [Woods 1968] system was the best known NLI to database system of the late sixties and early seventies. It was designed by William Woods to help geologists access, compare and evaluate chemical analysis data on moon rock and soil composition obtained from the Apollo-11 mission.

The system was designed as an interface to a two file database containing information about chemical analysis of moon rocks. It consisted of a parser with a large dictionary, a general purpose semantic interpretation component, and a retrieval component.

LUNAR operates by translating a question entered in English into an expression in a formal query language. The translation is done with an Augmented Transition Network (ATN) parser coupled with a rule-driven semantic interpretation procedure, which guides the analysis of the question. There are three steps in LUNAR to process a question:

- Produce the derivation syntax analysis tree for the request by using an augmented transition network parser and heuristic information.

- Generate a representation of the meaning of the request in a formal query language by using a semantic interpreter.
- Execute the query language expression on the database to produce the answer to the request.

When a number of geologists asked questions of it, LUNAR was able to answer about 78% of them [Woods 1968]. The questions were limited to those that were relevant to the database, but there were no other restrictions on the inputs.

Although LUNAR pioneered many of the techniques that still underlie most NLIs, it was built having a particular database in mind. That means it could not be easily modified for different databases even with the internal representation methods used in LUNAR. These were argued to facilitate independence between the database and other modules. The way that these were used was somewhat specific to that project's needs.

LADDER

The LADDER (Language Access to Distributed Data with Error Recovery) system was a milestone in the development of practical NL database systems. It was designed at SRI as a sophisticated, human-engineered system, rather than a demonstration of a set of syntactic, semantic, and pragmatic theories. The stated goal of LADDER was to provide users with a transparent way of gaining access to information distributed over various databases, where any given user query might require information from several databases, each built on a different DBMS [Hendrix et al. 1978].

The tasks of getting information from the databases are distributed over three different modules:

INLAND (Informal Natural Language Access to Navy Data) is a

linguistic component, built using SRI's LIFER (Language Interface Facility with Ellipsis and Recursion), whose task is to accept a user query in a restricted subset of natural language and produce a query or sequence of queries (like a command list of constraints).

IDA (Intelligent Data Access) whose task is to translate the command list into a sequence of queries against individual files in the languages of the remote DBMSs.

FAM (File Access Manager) which is to find the location of the individual files and manage access to them and remote failures.

LADDER used semantic grammars that interleave syntactic and semantic processing. Although the author showed a preoccupation for allowing portability across domains by using semantic grammars which helped to implement systems to different application domains, a different grammar had to be developed whenever LADDER was configured for a new application [Barros 1995].

2.2.2 Domain Transportable Systems

The early natural language interface evolution led to the development of front ends presenting higher degrees of portability. In this section, we will describe three of the most prominent and well-grounded portable front end systems, based on an evaluation done by Barros and DeRoeck [1993]: TEAM, INTELLECT, and CLE which is a tool for supporting the development of NLIs such as CLARE.

TEAM

TEAM (Transportable English database Access Medium) [Martin et al. 1983], is a portable natural language interface to relational databases[1]. The insistence on transportability distinguishes TEAM from previous systems. Most previously built NLI systems used techniques that make them inherently difficult to transfer to new domains and databases. The internal representations in these systems typically intermix information about language with information about the domain and the database. In addition, in interpreting a query the systems mix what a user is requesting with how to obtain the information requested. To achieve the transportability TEAM separates information about language, about the domain, and about the database.

The system's major hypothesis is that the information needed to adapt an NLI to a new database domain can be acquired from users if it is constructed in a sufficiently well-principled manner. The users, who carry out customisation, are database experts but do not possess any special knowledge about natural language processing or the particular NLI.

The translation of an English query takes place in two steps: The *dialogic*, which constructs a logical form representation of the literal meaning of the user's input question, and the *schema translator* which uses the logical form of representation to produce a query in the appropriate database management system language.

TEAM had a domain independent parser (bottom-up), an augmented phrase structure grammar, core lexicon, semantic interpretation routines, basic sort/type taxonomy, basic pragmatic function, quantifier scope algorithm, and schema translator. None of these had to be touched in a move to a new domain or database.

---

[1]the relational model of data was introduced by Codd [1970]. A relational database consists of a set of relations. Each relation can be considered to be a tabular structure in which the rows represent tuples (entities) and the columns represent attribute domains (descriptions) of the tuples.

All queries are analysed by the parser to produce a set of syntactically acceptable parse trees using context-free grammar rules which are annotated with constructors providing for context-sensitive constraints. These annotations help to choose the 'best' syntactic parse tree, based on a priori syntactic criteria. The best parse tree is then annotated with semantic information. On the other hand, this kind of mechanism affects the system's transparency, since the choice of the best parse tree is made on a syntactic basis. Thus, the user's intended query meaning might be ruled out before reaching the database [Barros 1995].

The pragmatic analyser takes the semantic representation output and assigns specific meanings, of relevance for the current domain, which resolve remaining ambiguities. The final translation stage of the query, in *dialogic*, is the quantifier-scope determination process. The best relative scope for the quantifiers in the query is assigned, after considering all possible alternatives.

The logical form produced by DIALOGIC is translated into a query in the SODA database query language by the schema translator. This translator uses information in the *conceptual schema*, which contains information about the objects, properties, and relations in the domain of the database, and *database schema*, which provides information about the actual structures used in the particular database being used. The information used here is domain dependent.

Domain dependent information such as the lexicon, conceptual schema, and database schema were acquired from the user during TEAM's acquisition phase, which was designed to elicit information from someone who was not necessarily a specialist in NLP. The acquisition process is fully automatic which means the system decides when enough information has been acquired. This acquisition module is crucial to its success as a transportable system.

The system is designed to interact with two kinds of users: a database expert and an end user. The database expert engages in an acquisition dialogue with the system to supply the information needed to adapt the system to a new

database or to expand its capabilities in answering questions.

INTELLECT

INTELLECT, from ROBOT [Harris 1980], is the longest-established and most widely used main frame interface product [Johnson 1985]. It accepts fragment of queries and requests spelling correction as necessary. Also, it requests and assists the user to resolve ambiguities. Query paraphrases are provided.

INTELLECT was designed to be a domain-independent system which means it separates a linguistic module from a domain access module. The linguistic module consists of a parser and a semantic analyser which produce a domain independent meaning representation. This is then translated, by the domain access module, into a database query language for retrieving the database answer. The system's lexicon is not complete as it allows some words to appear in the database only.

Every possible syntactic interpretation of the query is generated by the parser to make the system able to deal with lexical and structural ambiguity. These interpretations are validated against the database. A dialogue is established with the user to clarify the intended interpretation in the case where more than one interpretation survives.

There is no knowledge base or world model in the system. All necessary domain dependent information is provided by the database itself. In some installations of INTELLECT, the employment of the database as a knowledge base is achieved by triggering two types of database: an operational database and an informational database which is derived from the operational database on a frequency basis (daily, weekly, etc.). The first one is directed to the updating process while the other database is heavily indexed to allow for quick validation of the interpretations.

The system customisation to a new domain involves only updating lexicon and database information. This process can be carried out by a database administrator, not necessarily by a system expert as in TEAM. INTELLECT is able to handle queries like:

> Show subtotals of direct commission by month for first quarter 90 in region X where net amount is at least 100,000.
>
> For summaries with loss ratio greater than 200, report region, branch and loss reserve by month.

One problem is the absence of English verbs from databases and data dictionaries. A few verbs, such as "report (on)", "give (me)" are built into INTELLECT, but the system does not derive the meaning of a domain-dependent verb like "fly" or "sail" from the data model. Another problem is that nouns and verbs are not grammatically recognised, as there is no full linguistic analysis of syntactic categories. The system does not attempt to analyse the English grammar of a query at all [Wallace 1984].

INTELLECT has difficulties interpreting logical AND expressions when reference is made to different values for the same field. It will interpret AND as OR. Also, the system does not distinguish between noun phrase coordination and clause coordination. A more detailed evaluation of INTELLECT and its limitations is found in Wallace [1984], Md Sap and McGregor [1992] and Johnson [1985].

Generic Front-Ends Systems

In recent years NLIs continue to evolve, adopting advances in the general-purpose systems that map natural language input to expressions of a logical language. These generic front-ends can be turned into NLI to database systems, by attaching additional modules that evaluate the logic expressions

against a database. The Core Language Engine (CLE) [Alshawi et al. 1988 1991] and LOLITA (Large-scale Object-based Linguistic Interactor Translator and Analyser) [Smith et al. 1994] are examples of this approach.

LOLITA is a NL system based on Natural Language Engineering (NLE)[2] principles. It is more than a generic system as it is not restricted to a single task type, but it is a general purpose language processing machine which can be used for any language analysis task in any domain. Because of the limited space here we will only overview CLE as a representative of this kind of system.

CLE is a domain independent system for translating natural language (English) sentences into formal representations of their literal meanings which are capable of supporting reasoning. It can be used as the basis for building natural language applications.

Each lexical entry includes four parts: the word form, optionally followed by its stem form, a list of syntactic categories for the word, and one sense entry, the semantic counterpart to the lexical entry in the syntactic component of the system. The word's meaning representation is conveyed by the word's sense entry in Quasi Logical Form (QLF).

The grammar rules are written based on the Generalised Phrase Structure Grammar (GPSG) formalism [Gazdar et al. 1985]. Each syntactic rule has one corresponding semantic rules. The semantic rule specifies how the semantic features and the interpretation of the mother node depend on features and interpretations of the daughter nodes.

The main goal of CLE is to translate a natural language input into a Logical Form (LF). This is achieved in two stages, first the system translates the NL sentences into QLF and then transfers the QLF to LF. The first step involves three processing steps: morphological analysis, syntactic parsing, and semantic

[2]The principal defining characteristic of NLE work is to engineer products which deal with natural language and which satisfy the constraints in which they have to operate [Boguraev et al. 1995].

analysis. The parser is based on a bottom-up left-corner analysis. Semantic analysis is conveyed only for constituents that are part of a complete syntactic analysis of the input sentence.

The QLF is a language-dependent, conservative representation of meaning of an input sentence based on the results of compositional linguistic analysis independently of contextually sensitive aspects of understanding. It has anaphoric terms, quantified terms, and anaphoric formulae. These constructs include syntactic and morphological information in the *Category* and logical information in the *Restriction* [Alshawi et al. 1991].

Moving on from the QLF representation level to the LF level involves four steps: sortal filtering, scope determining, reference resolution, and plausibility checking. Type information is introduced into the logical form for the interpretation of the constituent under analysis, to rule out any interpretation with incompatible types. In order to generate the LF, the QLFs with proposed scoping and reference resolutions are checked, by applying both linguistic and nonlinguistic constraints, for plausibility in the current domain.

The system is highly modular, most of its components can be adapted easily to a new domain or a new task. It has a lexicon acquisition tool VEX (Vocabulary EXpander) which allows users, with knowledge both of English and the application domain but not of linguistic theory or knowing how the lexical entries are presented in the system, to update the system lexicon.

CLE was used in CLARE [Pulman et al. 1993] as the linguistic processor for the analysis and generation of sentences. In CLARE mechanisms were developed for carrying out several types of reasoning. These include noun phrase reference, ellipsis or phrase fragments, and resolution of ambiguity. A further phase of reasoning is required to connect the linguistic concepts extracted to those employed in the information system itself. Similar reasoning is involved in generating English sentences representing the answers to queries. The natural language and reasoning techniques employed are as general purpose as possible. So, it is relatively easy to adapt components of CLARE.

Finally, CLE can be adapted to a new language. The system has been used to develop the Swedish Core Language Engine (S-CLE) [Gamback and Rayner 1992].

2.3 NLI Problems

Although the task of natural language interfaces is restricted enough to be feasible and would be very useful, building effective front ends has proved much more of a challenge than was expected [Copestake and Jones 1990].

2.3.1 Ambiguity

One major issue in NLI is ambiguity. There are two main classes of linguistic ambiguity: lexical ambiguity and structural ambiguity [Hassan 1988]. A question like:

Who takes the course on Java?

could be asking about students or about the lecturer. Here there is a lexical ambiguity in "Who" and "take". "Who" refers to people and does not differentiate according to their status as student or lecturer. If only one translation of "take" has been specified the system would not detect the ambiguity and considerable difficulties could be caused if the system's interpretation is not the one that the user intended.

If the ambiguity is recognised, and there is no prior context such as a series of questions about students, the only reasonable cause of action is to get the user to choose between the alternative readings.

The second class of ambiguity is structural ambiguity. Consider the following example taken from [Copestake and Jones 1990]:

Which courses does every lecturer teach?

This query could be interpreted as:

List the courses such that all lecturers teach that course;

or

For all lecturers list the courses that they teach.

One way of dealing with ambiguities is to display the choices after the user expresses the query. Another way to resolve the ambiguity is through a dialogue between the system and the user. Thus the system detects the ambiguity and asks the user for further clarification, as we do in human communication. But a long dialogue may lead the user to feel that the system is trivial and boring. It is important that ambiguity is detected and the user is consulted before going ahead with a database query.

2.3.2 Useful Answers

It is quite easy to build an interface which will translate a limited range of queries into formal queries on a particular database but it is obvious that such a system is not very useful to the users [Copestake and Jones 1990]. The answer meaning of the query is very important to the user. Consider a query like:

How many students failed in COM335 in summer 95?

If the query returns a null result (i.e. the answer is just '0'), what does the answer mean. Does it mean:

A- nobody failed.

B- COM335 was not offered in summer 95.

C- there is no such course called COM335.

The user should be informed which part of the query returned the null result.

In order for any natural language interface to be useful, it must build a model of the user for effective co-operative interaction. This is important for generating useful natural language responses [Mc Kevitt et al. 1992].

2.3.3 A Failure to Find an Answer

SQL tends to be able to answer more questions than most NLIs can understand. One of the main problems with NLIs is the difficulty of determining the cause of a failure to find an answer [Md Sap and McGregor 1992]. It is an important aspect of NLIs to have the ability to deal constructively with failures. Failures can be ascribed to a lack of syntactic, semantic, or pragmatic coverage of the system, or to a user failure. A user failure arises when his beliefs about the domain of discourse differ from those of the system.

Another problem is that an NLI to a database may misinterpret a question submitted by the user, without the user becoming aware that his/her query has been misinterpreted. Natural language questions often have more than one reading; the NLI may select a reading of a question that is different from the reading the user had in mind when he/she typed the question.

2.3.4 Conjunctions

One of the difficult problems for any parsing algorithm is the scope of conjunctions. Consider the following example, borrowed from Loqui [Binot 1991]:

> Print the title and the duration of the project that follows Loqui.

The scope of the conjunction 'and' must be determined in order to obtain a correct semantic interpretation by making a choice between the two following possibilities:

(the title and the duration) of the project.

or

(the title) and (the duration of the project).

The logical structures corresponding to these two interpretations are:

$$(The!1x|project(x) \wedge follows(x, Loqui)) \wedge$$
$$(The!1y|title(y, x)) \wedge$$
$$(The!1z|duration(z, x)) \wedge print(y) \wedge print(z),$$

$$[(The!1y|title(y, v)) \wedge print(y)] \wedge$$
$$[(The!1x|project(x) \wedge follows(x, Loqui)) \wedge$$
$$[(The!1z|duration(z, x)) \wedge print(z)].$$

In the second interpretation, the title cannot refer to the variable x which denotes a project that follows Loqui, because it is outside the scope of that variable. Instead it introduces the $title(y, v)$ with a free variable v which is not acceptable. For that reason the first interpretation will be prefered. Binot [Binot 1991], proposes a possible solution which is to generate the possible interpretations and order them by proposing the most restrictive ones first, then let the domain dependent interpreter make a reasonable choice.

Another problem with conjunctions is their usage in the natural language. The word 'and' is not necessarily used to denote the logical meaning. It is often used to denote disjunction rather that conjunction [Androutsopoulos et al. 1995]. For that reason some NLI systems, EUFID [Templeton and Burger 1983] for example, change the English 'and' to logical "or" when the two phrases within the scope of the conjunction are values for the same field, as can be seen in the following examples from EUFID:

List the applicants who live in California and Arizona.

Which minority and female applicants know Fortran and Cobol.

The first example means any applicant who lives in California or Arizona should be reported as an applicant who has only one state of residence. The second example present multiple conjunctions in a single query which is ambiguous, as it can be interpreted with logical 'and' or with logical 'or'.

2.3.5 Yes/no Queries

In natural language interfaces users may wish to use yes/no questions in their interaction dialog. For this type of questions some NLIs, such as Squirrel, return a boolean answer with a constant value yes for the true case and an empty value for the false case. While other systems, like EUFID, map the yes/no question into a query which will retrieve some data if the answer is "yes" and no data if the answer is "no".

This is because no database management systems has yet the ability to answer "yes" or "no" explicitly [Templeton and Burger 1983]. However, the response may be empty for several reasons as explained in section (2.3.2) above.

The problem becomes even more difficult if there is a conjunction in this kind of question. For example consider a query like:

Are Mark and John in course number 3325?

If Mark is in the course and John is not, neither "yes" nor "no" will be a correct anwer. This case needs an explicit answer.

2.4 Portability

Portability is one of the important issues in NLI systems. There are two directions in NLIs transformation. The first one is to customise the NLI system to a new domain while the second one is to transform the system to another natural language.

Based on the literature, a great amount of effort of the past three decades has been expended on domain portability which results in a number of ideas and systems (e.g. TEAM, Loqui, EUFID, Squirrel, LOLITA, INTELLECT and so on). On the other hand, there has been little work done on language portability.

2.4.1 Domain Portability

Binot [1991] defines it as:

> *the capability to adapt natural language processing to a new application domain with a minimum of effort. [P.59]*

Early systems (see section 2.2.1) were designed for a specific database application which means they give a direct translation of the natural language input to the specific domain. This direct translation increases the difficulty of separating the application dependent semantic processes from the more general mechanisms implemented in the parser [Binot 1991]. Even in LADDER, which showed preoccupation with allowance for portability across domains, a different grammar had to be developed whenever the system was configured for a new application.

The second generation of NLI systems, corresponding to the state of the art, separate the interpretation process into two stages. The first stage is domain

independent which translates the input natural language into a formal meaning representation. The second stage processes the formal meaning representation to produce the equivalent of the database query language which is domain dependent. This separation minimises the effort of the customisation process for the new domain.

An acquisition session is assumed by the portable systems in order to facilitate the customisation process. It is responsible for obtaining domain dependent knowledge which is necessary to compound the lexicon (if it is not complete), the grammar (if it is not fully general), the world model or knowledge bases (if any), and the data model. Each system has its way of dealing with the customisation process, which can be manual as in INTELLECT, semi-automatic as in CLE, or fully automatic as in TEAM. Also, they have different kinds of users in charge of this process: database expert, system designer, application expert, and end user [Barros 1995].

An evaluation on the degree of portability of some of NLI systems can be found in Barros and DeRoeck [1993].

2.4.2 Cross-Language Portability

This idea comes from the fact that the largest part of the NLI research area has been based on using English for the natural language input which results in a number of NLI systems with a sophisticated natural language capability. Thus, when some one wants to build a NLI system for another language (such as Arabic, Spanish, Swedish, Portuguese and so on) it is better to adapt and convert an existing one instead of starting from zero. This may minimise the time and effort required for the development of the new system.

A number of English front end systems have been adapted to other languages. CHAT-80 [Warren and Pereira 1982] was adapted to Portuguese [Lopes 1984]. The Loqui system [Binot 1991] was originally based on English then it was

converted to two languages, French and German. Also, the CLE system was adapted to several languages, Swedish [Gamback and Rayner 1992], French and Spanish [Rayner et al. 1996].

One objective of the previous work was to examine the effort and time needed to port a natural language processing system to other languages. The focus was to identify the problems of extending an English-based NLP system to handle more Indo-European languages. Another objective was to raise the issue of modularity and portability for NLP systems in order to make them adaptable to a new natural language.

2.5 Conclusion

The study of natural language interfaces is a practical topic and is increasingly becoming an appropriate test area for many natural language processing techniques, resulting in several systems available commercially. Most of these systems purport to allow querying of databases in English.

Having said that, it must be remembered that this claim is flexible in its possible interpretations. These systems may be extremely effective tools for database access, and may use some form of English (or other natural languages), but it does not mean that the problem of completely unrestricted use of natural language queries has been solved [Androutsopoulos et al. 1995].

The portability issue during last decade was concerned with porting the front ends to a new application domain with a minimum of effort. However, the large number of systems available based on the English language has caused this issue to become gradually directed at cross language portability. Several English systems have been adapted to other languages.

§§§§

Chapter 3

Arabic Syntax

The Arabic language is one of the Semitic language family. Thus its structure is different from English or other Indo-European languages. The Semitic languages are recognised in terms of the similarities of features characterising them. Of course there are differences between Arabic and other Semitic languages. Arabic has its own unique alphabet system and it is written from right to left.

This chapter aims to describe the syntax of the Arabic language which will be the natural language input to the Squirrel system, see Chapters 7, 8, 9, and 10. It provides a general outline of Arabic and its characteristics and features.

3.1 Background

There are three classifications of Arabic: Classical Arabic (اَلفصحَى alf.s.h_A), Colloquial arabic dialects, and Modern Standard Arabic. Classical Arabic is the language of the Quran (the holy book of Islam). It could be also viewed as the language of the pre-Islamic poets and the language of medieval Islam. Colloquial Arabic dialects, on the other hand, consist of the languages of the

different Arab countries. They are used as oral communication for daily life by people of the dialect area. There are no written transcripts for such dialects.

Modern Standard Arabic (MSA) is the language of today's Arabic newspapers, magazines, periodicals, letters and modern writers. It is also used as the medium of oral communication in formal speeches, and in television and radio broadcasts. MSA could be viewed as classical since there have been no major changes modifying the structure of the classical language. MSA, however, differs from Classical Arabic in two aspects: adopting minor stylistic changes and expanding the lexicon to include new technical terms [Al-jabri 1997]. The language that this work is concerned with is MSA.

The most characteristic feature of the Arabic language is that the great majority of its words are built up from or analysed down into roots, each of which consists of three radicals (فَعَل fa'al) [Ziadeh and Winder 1957]. The actual words are produced by using these radicals as a base and adding prefixes, infixes and suffixes, according to certain patterns.

Arabic is a synthetic 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 numbers, persons, genders, derived forms, moods, and tenses [Mehdi 1986].

3.2 Word Classes

Arabic Grammarians recognise three parts of speech or word classes: noun (إسم 'ism), verb (فعل f'l), and article (حرف .hrf). Adjectives, adverbs, and pronouns are classified as nouns [Abn-Hesham 1985]. The concept of verb is the same in Arabic as in English. Articles in Arabic include conjunctions, prepositions, and interjections.

3.2.1 Nouns

There are two kinds of nouns in Arabic: solid nouns which are not derived from a verb form, and derived nouns which are derived from a verb form. Thus, nouns like verbs are distinguished by the wealth of derivatives from the root. The process of deriving nouns in Arabic is relatively systematic. The following example nouns are derived from the root (كَتَبَ kataba):

Table 3.1: Examples of derived nouns

ktAb	كتَاب	book
kAtb	كَاتب	writer
kutub	كُتُب	books
ktAbT	كتَابة	writing
mktwb	مكتوب	something written
mktb	مكتب	office
mktbT	مكتبة	library

Any adjective is a kind of noun and is not considered a separate part of speech. The common way of creating an adverb is to use the corresponding noun in the accusative case.

Pronouns

Arabic has twelve forms for independent personal pronouns. They can occur either as free forms or as suffixes attached to nouns, verbs, or prepositions. Their form is affected by the number and gender of the things they relate to. Table 3.2 presents the independent personal pronouns of Arabic.

Arabic has different forms of relative pronouns based on the gender and number, as shown in Table 3.3.

Table 3.2: The independent pronouns

singular	dual	plural
ana, I (mas,fem) أَنَا	huma, They (mas,fem) هُمَا	antum, You (mas) أَنتم
anti, You (fem) أَنتِ	antuma, You (mas,fem) أَنتُمَا	nahnu, We (mas,fem) نحن
anta, You (mas) أَنتَ		hum, They (mas) هم
hiya, She (fem) هي		antunna,You (fem) أَنتن
huwa, He (mas) هو		hunna, They (fem) هن

Table 3.3: The relative pronouns

Pronoun		Gender	Number
الّذي	All_dy	masculine	singular
الّتي	Allty	feminine	singular
الّذَان	All_dAn	masculine	dual
الّتَان	AlltAn	feminine	dual
الّذين	All_dyn	masculine	plural
اللاتي	AllAty	feminine	plural

There is another type of pronoun which is dependent, i.e. it is attached to another word like the pronoun (هَا hA) in (يدرسهَا ydrshA) and cannot be alone. These pronouns mostly refer to nouns or proper nouns preceding them.

3.2.2 Verbs

There are two kinds of verb: complete verb and incomplete verb. A complete verb is one which refers to an action and a tense while an incomplete verb is one which refers to tense but not to an action. Arabic has a limited number of incomplete verbs: (كَان kAn = to be) and her 'sisters' and (ليس lys = not to be) and her 'sisters'.

In Arabic verbs can be derived from verb roots by the systematic addition of one or more affixes to a root. Each of these derived forms bears a specific semantic relationship to the simple verb.

There are only two tenses of verbs in Arabic, perfect which denotes completed actions, and imperfect which denotes incomplete actions (irrespective of time). Arabic perfect is usually the same as the English past tense, and Arabic imperfect is the same as the English present or future tense [Mehdi 1986].

3.2.3 Articles

Articles in Arabic have no meaning when used alone but when used in a sentence or phrase they may have more than one meaning. The main articles in Arabic are conjunctions, negations, conditions, prepositions, and interjections. Some articles are attached to the noun or to the verb as a prefix and/or a suffix to add extra information such as gender, number, and time.

The definite article for all cases, i.e. number and gender, is (ال Al), which is written prefixed to the word it defines. There is no indefinite article in Arabic.

3.3 Gender

There are two classes of gender in Arabic:

(a) only feminine, which may be either real, as (إمرأة 'imra'T) a woman, or natural, as (ألشمس alŝms) the sun;

(b) only masculine, which may be either real, as (رجل rĝl) a man, or natural, as (ألقمر alqmr) the moon.

Arabic does not have what is called the *neuter* gender. Gender is grammatical not necessary natural which means nouns may be either masculine or feminine, but not necessarily male or female. Thus, gender for any mammal noun means male or female but for anything else means either masculine noun or feminine noun. Any mammal common noun is affected by its gender in its referent. In Arabic, the plural masculine and masculine form of irregular plural (broken plural in Arabic see Section 3.4) of a mammal common noun are used to refer to male gender, and can also be used to refer to a group of mixed gender male and female.

3.4 Number

There are three general forms for number in Arabic: singular which is anything up to one, dual for two, and plural for three or more. The dual can be constructed by adding a suffix to the end of the singular of a word after removing its case ending as in the following example:

rsAlTuN	رسَالةٌ	a letter
rsAltAn	رسَالتَان	two letters (nominative)
rsAltyn	رسَالتين	two letters (accusative and genitive)

If the word is feminine, the dual is constructed by changing the last suffix and then adding the new suffices. There are three plural types:

a) The sound masculine plural: This can be constructed by adding (ون wn), as in (مدرّسون mdrrswn), for nominative case and (ين yn), as in (مدرّسين mdrrsyn), for accusative and genitive cases.

b) The sound feminine plural: This can be constructed by changing the ending (ة T) to (اتُ Atu) for the nominative case and (اتِ Ati) for the accusative and genitive cases. (شجرة ŝĝrT) for example becomes as (شجرات ŝĝrAti) or (شجرات ŝĝrAti).

c) The broken plural: This can be constructed according to some common patterns. But it is not like (a) and (b) above, sometimes the singular needs to be completely changed to produce its plural. A word like (نسَاء nsA') is the plural form for (إمرأة 'imra'T).

3.5 Vowels and Nunation

There are three short vowel marks which correspond to the three cases that occur in Arabic nouns: nominative, accusative, and genitive. These marks are:

a) (ضمّة .dmmT), a small character above a consonant أُ, as in (رجلُ rĝlu);

b) (فتحة ft.hT), a small diagonal stroke above a consonant أَ, as in (رجلَ rĝla);

c) (كسرة ksrT), a small diagonal stroke under a consonant إِ, as in (رجلِ rĝli).

Nunation is concerned with the process called (تنوين tnwyn), doubling of these short vowels to become (ضمّتين .dmmtyn), (فتحتين ft.htyn), and (كسرتين ks-rtyn). The word (رجلُ rĝlu), for example, becomes like (رجلٌ rĝluN). These are placed on the last character of an indefinite noun and have a phonetic effect of placing (ن n) at the end of the word.

3.6 Arabic Sentence

Arab grammarians have proposed two theories to describe the functional role of Arabic sentences: (إسنَاد 'isnAd) and (عمَاد 'mAd) [Abn-Hesham 1985]. The first one states that the Arabic sentence is constructed mainly from (مسند msnd), i.e comment or predicate, and (مسند إليه msnd 'ilyh), i.e topic or initial constituent. Therefore, for every (مسند msnd) there must be a (مسند إليه msnd 'ilyh) in the sentence. The second theory is concerned with the essential and the complement parts of the Arabic sentence. It states that the only essential parts in the sentence are the (مسند msnd) and (مسند إليه msnd 'ilyh) and are called (عمدة 'mdT). The other part is a complement and is called (فضلة f.dlT).

In terms of syntactic structure Arab linguists have proposed several classifications of Arabic sentences from different viewpoints. The most basic common classification divides Arabic sentences into two types: verbal and nominal [Hassn 1975]. A verbal sentence is one that starts with a verb. On the other hand, a nominal sentence is one which starts with a noun or a number of tools, such as question's articles, followed by a noun. It may or may not contain a verb.

Abn-Hesham [1985] divides Arabic sentences into: verbal, nominal, and circumstantial which starts with a preposition. He also defines another classification which is based on: complex sentence (جملة كبرَى ĝmlT kbr_A) and simple sentence (جملة صغرَى ĝmlT .s.gr_A). The complex sentence is the one which

contains another sentence embedded in it while the simple sentence consists of only two phrases each of which cannot form a sentence on its own.

Saad Al-jabri [Al-jabri 1989] viewed the different classifications of Arabic sentences and came to the conclusion that they did not satisfy the requirements of computational modeling. Therefore, he proposed the following classification for Arabic sentences:

- *A pure sentence* is a sentence with a sequence of nouns or verbs or both but it does not contain any tool.
- *A tool sentence* is a sentence with a sequence of nouns or verbs or both, containing at least one tool.
- *A Tool* is an article such as preposition.
- *A simple sentence* is a sentence that represents an independent structure.
- *A complex sentence* is a sentence that contains more than one independent structure.
- *A nominal sentence* is a sentence that starts with a noun or a number of tools followed by a noun. It may or may not contain a verb.
- *A verbal sentence* is a sentence that starts with a verb. It may be preceded by one or more tools.

Aljabri has provided a new computational classification model for the syntax of Arabic sentences taking into account most of the linguists' views.

3.7 Interrogative Syntax

Because this work mainly treats interrogative sentences, we will explain here interrogative sentences in some detail. Arabic questions start with interrog-

ative tools which are either articles or pronouns. The articles are used for yes/no questions while the interrogative pronouns are used for wh questions as discussed in the following sub-sections.

3.7.1 Interrogative Tools

Based on the Arabic word classification, there are two types of interrogative tools: articles and nouns. A list of all of the Arabic interrogative tools is shown in Table 3.4.

Table 3.4: The interrogative tools

Pronouns	من	mn	Who
	مَا	mA	What
	مَاذَا	mA_dA	What
	لِمَاذَا	lmA_dA	Why
	أَين	'ayn	Where
	متَى	mt_A	When
	كم	km	How many / How much
	أي	'ay	Which
Articles	أَ	'a	Is/are/did/do
	هل	hl	Is/are/did/do
			These are approximate meaning as there is no exact equivalent in English.

These tools can be classified, according to the purpose of the question, into three types. The first type contains elements for asking about a statement and its intrinsic validity, i.e. yes/no questions. The second one contains elements for enquries about an essential element (subject, object) of the interrogative statement, while the third one is, elements for asking about circumstances, like (متَى mt_A), or adverbial aspects of the interrogative sentence, like (أَي 'ay).

3.7.2 Yes/no Questions

This type of question should start with question article either (أَ 'a) or (هل hl) as a question tool followed by a sentence. They are similar but they are not alike. They should be found at the beginning not in any other places in the query sentence. Both queries Q2 and Q4 are incorrect syntactically[1]. Both articles can be used to introduce a nominal sentence and a verbal sentence without any effect on the word order of the sentence. The question article is part of the query sentence but not part of the following sentence. In other words, an Arabic yes/no question is built up from a question article plus a nominal sentence as in Q3 or a verbal sentence as in Q1.

Q1) أَ نجح علي ؟
'a nĝ.h 'ly ?
(did passed Ali)
Did Ali pass?

Q2)* نجح أ علي ؟
nĝ.h 'a 'ly ?
(pass did Ali)

Q3) هل علي في الفصل ؟
hl 'ly fy Alf.sl ?
Is Ali in the class?

Q4)* علي هل في الفصل ؟
'ly hl fy Alf.sl ?
(Ali is in the class)

The (أ 'a) is always used at the beginning of the sentence even if the sentence has a coordination article or preposition. Frequently, it can be used to introduce a negative statement as in Q5 but it emphasizes the statement, rather

[1]The star indicates the sentence is incorrect grammatically.

than questions its validity. (هل hl) is almost like (أَ 'a), except that (هل hl) comes after conjunctions not before them. Also it cannot be used to introduce a negative statement as in Q6, a sentence contains (أَن 'an), or a sentence which contains more than one choice.

Q5) أ ليس أحمد في المدرسة ؟
'a lys 'a.hmd fy AlmdrsT ?
is not Ahmed in the school?

Q6)* هل ليس أَحمد في المدرسة ؟
hl lys 'a.hmd fy AlmdrsT ?
is not Ahmed in the school?

There is not any kind of agreement between these interrogative articles and the rest of their sentences.

3.7.3 WH Questions

These questions have an interrogative pronoun which introduces and indicates the element of the sentence about which the question is asked. Unlike the yes/no question, the question tool here is part of the sentence. These question pronouns usually have the priority to be at the beginning of the sentence as in Q7 which is the usual word order for an Arabic question.

The pronoun (من mn) can be used in asking about a person and (مَا mA) and (مَاذَا mA_dA) about a thing. (أَي 'ay) has an explicative or a partitive meaning. The interrogative pronoun (كم km) is for quantitative and qualitative value.

The personal pronoun that follows these interrogative pronouns should be realised as a resumptive repetition of the interrogative pronoun. They may be preceded by a preposition as in Q8.

The (أَي 'ay) is usually followed by a genitive in the singular or plural form, which can be in a definite or indefinite form, and can be in any of the three grammatical cases; accusative, nominative, and genitive. It may be preceded by a preposition but not followed by a preposition.

(كم km) is always found at the beginning of the sentence because of its indefinite character. It may be more closely determined by an accusative singular function as an accusative of specification like the example in Q9.

Q7) من درّس مَادة بيسك ؟
mn drrs mAdT bysk ?
who taught course Basic.

Q8) في أَي قسم يدرس الطَالب زيد ؟
fy 'ay qsm ydrs Al.tAlb zyd ?
In which department is the student Zyd studying?

Q9) كم طَالبَاً رسب في مَادة بيسك ؟
km .tAlbAaN rsb fy mAdT bysk ?
how many student failed in course Basic.

Questions about any circumstances or adverbal aspects should be introduced by (متي mty), (أَين 'ayn), and (كيف kyf). These interrogative pronouns may be preceded by a preposition or conjunction.

3.8 Word Order

Word order is one of the important linguistic aspects of Arabic. The most common word order is Verb-Subject-Object (V-S-O). In standard linguistic analysis all the following three sentences have the same meaning in Arabic.

S1) كَتَبَ خَالِدٌ الرِسَالَةَ

kataba _haAliduN alrisaAlaTa V-S-O

wrote Khalid the letter

Khalid wrote the letter

S2) كَتَبَ الرِسَالَةَ خَالِدٌ

kataba alrisaAlaTa _haAliduN V-O-S

wrote the letter khalid

Khalid wrote the letter

S3) رِسَالَةٌ كَتَبَ خَالِدٌ

risaAlaTaN kataba _haAliduN O-V-S

letter wrote it Khalid

A letter is written by Khalid

One might ask why the Subject-Verb-Object and the Object-Subject-Verb are not considered as possible Arabic word orders. In fact, there is a debate about the possibility of the subject preceding the verb[2]. Consider the following examples:

S4)	عليًا	ضرب	الرّجل
	Alrrĝl	.drb	‘lyA
	the man	hit	Ali
S5)	عليًا	ضربَا	الرّجلَان
	AlrrĝlAn	.drbA	‘lyA
	the two men	hit (they)	Ali

S6)	عليًا	ضربُوا	الرّجَال
	AlrrĝAl	.drbUA	‘lyA
	the men	hit (they)	Ali

The above sentences (S4-S6) are complex sentences (جملة كبرَى ĝmlT kbr_A)

[2]More details about this issue can be found in [Abn-Hesham 1985],[Hassn 1975] and [Abdullatif 1982].

as each one contains another sentence embedded in it. For example, the first noun in sentence S4 (الرّجل Alrrĝl) is the topic (مبتدأ mbtda') and the verbal sentence (ضرب عليًا .drb 'lyA) is the comment (خبر ـhbr). The subject of the verbal sentence, i.e. (ضربَا عليًا .drbA 'lyA) in sentence S5, is the anaphora (أ a), which is the dual pronoun (الف الاثنين Alf AlA_tnyn), while in sentence S6 it is the anaphora (وَا wA), which is the plural pronoun (وَاو الجمَاعة wAw AlĝmA'T). Both anaphora refer to the preceding nouns (الرّجلَان AlrrĝlAn) and (الرّجَال AlrrĝAl) respectively.

For sentence S4, the verbal sentence, i.e. (ضرب عليًا .drb 'lyA), contains no overt sign for the subject. In this case the subject of that sentence is a covert pronoun (هو hw) referring to the preceding noun. This approach is more suitable with the use of Arabic [Abn-Hesham 1985].

3.9 Written Arabic

The vowel and nunation marks, which are placed at the end of the word, correspond to the three cases: accusative, nominative, and genitive. These endings are very important when you use all possible Arabic word orders as in the sentences (S1-S3) above.

Unfortunately, Arabic users rarely use these marks. Most Arabic written materials such as books, journals, documents, articles, papers and so on do not have these vowels for different reasons. One reason is, it is too time-consuming to write each character and its vowel. In addition, Arabic readers use other marks to distinguish subject from object as can be seen in the following sentences.

S7) ضرب عَليًا أَحمد V-O-S
.drb ‘alyA ’a.hmd
(hit Ali Ahmed)
Ali hit Ahmed.

S8) ضرب الرّجلَان الولد V-S-O
.drb Alrrĝlan Alwld
(hit the two men the boy)
The two men hit the boy.

S9) كلّم موسَى عيسَى V-S-O
kllm mws_A ‘ys_A
(spoke Mosy Essa)
Mosy spoke Essa.

S10) الرّجَال ضربهم محمّد V-O-S
AlrrĝAl .drbhm m.hmmd
(the men hit them Mohammad)
The men hit by Mohammad.

The first mark is (اَ a) which is usually added as a suffix to the object as in the word (عليًا ‘lyA) in S7. The second mark is the number marker in the case of dual and plural. There are two types of number markers: nominative marker like (ان An) in (الرّجلَان AlrrĝlAn) which is presented in the S8 example and accusative like (ين yn). The third mark is the type of anaphora. Arabic has two types of anaphora: indicative anaphora which is used for subject like the (وَا wA) in S6 and subjunctive anaphora which is used for the object as in S10. But if there are no vowels and none of the other two marks exist, as in the sentence of S9, then the sentence must be read based on the basic Arabic word order: Verb-Subject-Object.

§§§§

Chapter 4

A Review of Arabic NLI Systems

There are a number of Arabic interface systems which need to be examined before constructing any new work. Unfortunately, these systems are not available to use. Therefore, they can be evaluated only through the publications which describe them but they cannot be evaluated through an experimental evaluation session.

This chapter reviews Arabic natural language interface systems. It will also present an evaluation of some of them based on the information available in their publications.

4.1 Overview

Arabic sentences have been processed by a number of systems for different purposes. Some of them, for example, are concerned with machine understanding, i.e. make the machine parse the sentence and generate the (إعراب 'i'rAb), while others are concerned with machine translation. The focus here

is on systems which treat Arabic interrogative sentences.

The Arabic Language Interpreter (ALI) system [Mehdi 1987 1986] was built in the late eighties to interpret and represent the meaning of declarative and interrogative sentence(s). ALI used the Definite Clause Grammar (DCG) formalism, developed first by Pereira and Warren [1980], implemented in Prolog to parse Arabic sentences.

An expert system for understanding Arabic sentences using the Augmented Transition Network (ATN)[1] approach is proposed by El-Dessouki and his colleagues [El-Dessouki et al. 1988 1989]. They divided the rules into two sets: language independent rules and language dependent rules. The semantic analyser of the system checks the meaning of the sentences according to the system domain which was limited.

The Natural Arabic Understanding System (NAUS) is a large scale project developed by Khayat and his colleagues [Khayat 1989]. The project was to address the different areas of natural Arabic understanding and propose approaches to tackle some problems of these areas. The system accepts questions from the user and generates the answer. It can also treat some ellipsis.

During the nineties there has been a number of systems concerned with treating Arabic interrogatives. El Kareh and his colleagues [El Kareh et al. 1990] developed computer tools to analyse Arabic questions using GENIAL, a tool kit for the rapid construction of robust French natural language interfaces. The main constituents in a sentence handled by the system are a verbal phrase and a nominal phrase. The semantics are constructed by certain predicates associated with the words' lexical entries.

A Knowledge Based Arabic Question Answering System (AQAS) [Mohammad et al. 1993] is a system built to deal with a knowledge base in a radiation domain. The system recognises any query as consisting of two parts: a known part (the thing asked about) and a required part (the information requested),

[1] More details about the ATN formalism can be found in Allen [1995].

see Section 4.4. The AQAS main function is to extract the information from the given question that gives it the ability to retrieve the answer.

Al-Khazoon is a system dealing with Arabic interrogatives [Al-Khonaizi et al. 1995]. In this application, one abstract data type can be created to capture all possible Classified Bases (CB) which can be represented in the form of a relational database table. This table contains several columns which correspond to the CB used (i.e. human, non-human, verb, etc.). Each simple sentence in the Arabic natural language text is stored in a separate record in this table.

Yamani and Al-Zobaidie [1996] proposed a Question-Answering System (QAS) for Arabic. The system is based on Lexical Functional Grammar (LFG) theory, introduced first by Kaplan and Bresnan [1982], to analyse Arabic interrogative sentences to produce functional and semantic structures according to the LFG representation.

In the following sections some of these systems are discussed in much more details and are then followed by an evaluation section. The ALI and NAUS are chosen from the eighties and AQAS and QAS as systems from nineties. The selection is based on the theories that are used by the system and the completeness of the system, i.e. any system which concern with syntax only or semantic only will be excluded.

4.2 ALI

ALI [Mehdi 1986 1987] accepts sentences from a user, parses them and generates inferences from them. These inferences constitute an events database which can be questioned by the user. ALI consists of four main components: Input, Parser, Word Analyser, and Dictionary. The system treats simple declarative and interrogative sentence(s) written from left to right using the Latin alphabet.

The Word Analyser is the most important part of the system. It takes a word from the input sentence and strips its suffixes, then checks to see whether this word is in the dictionary module or not. Its main function is to analyse the different words of a sentence when entered in the system.

The aim of the parser is to parse an Arabic sentence and to produce a syntactic tree-like structure for the parsed sentence. The grammar is based on a combination of Phrase Structure Grammar (PSG) and Case Grammar (CG) [Fillmore 1968] written in terms of the Definite Clause Grammar (DCG) [Pereira and Warren 1980] formalism. The parser uses a top-down and left-to-right strategy. The grammar rules are extended to achieve agreement, in number and gender, between subject and predicate and also to achieve end changing which is required in the subject or predicate, according to whether they are feminine or masculine. The syntactic output tree for a sentence like (*"akala al waladu al muz." the boy ate the banana*) is:

```
P= S ( predicate ( verb ( akala), per, sing, m, rd),
       subject (det (al), noun (walad), sing, m, sn),
       object (det (al), noun (muz), sing, m, a)).
```

Every word or particle of the input sentence is assigned to its grammatical category. The system will not carry out any error analysis. Thus, in the case of an unsuccessful match, the user will be asked to try another sentence.

The system combines Case Grammar (CG) [Fillmore 1968], and Semantic Marker and Selectional Restrictions (SMSR) to build the meaning representation of the sentence. ALI focusses on the adaptation of the sentences to predefined senses which are guided by semantic case structures and are formulated around the verbs. However, ALI does not produce an intermediate level of meaning representation during the course of analysis. Instead, it maps well-defined constituents directly into a deep semantic representation [Mehdi 1987].

ALI is initially restricted to three types of basic Arabic sentences: simple nominal sentences which contain subject and predicate, simple verbal sentences which start with a verb followed by a subject, and simple interrogative sentences which are in the form of Q-V-E where Q is the question word, V is the verb, and E is entity in the question.

4.3 NAUS

The Natural Arabic Understanding System (NAUS) was developed by Khayat and his colleagues [Khayat 1989]. It consists of five components: morphological analyser [Al-Uthman 1989], end-case [Al-Sawadi and Khayat 1995], syntax [Al-jabri 1989], semantics and knowledge [Khayat and Al-Muhtaseb 1988], and sentence generator [Al-Safran and Khayat 1992]. NAUS has a user interface [Al-Muhtaseb and Khayat 1988] which changes the input string into an internal representation character set and vice versa.

The system's morphological analyser converts each word of the input to its root. Verbs and nouns are determined by the root of the word, the derivation pattern, tense, number, gender, prefix, infix, and suffix.

NAUS syntactic analyser is designed and implemented based on a new classification of Arabic sentences [Al-jabri 1989]. Most of the syntactic constructions are implemented in a Context Free (CF) grammar which was introduced by Chomsky [1965]. However, some constructions, such as incomplete questions and ellipsis, are treated by transformational grammar. The system uses rules like the following constructions rules:

```
interrogative_sentence ---> interrogative_article, noun.
interrogative_sentence ---> interrogative_article, noun, verb.
interrogative_sentence --->
                        interrogative_article, noun, verb, noun.
```

```
nominal_sentence ---> noun, noun.
nominal_sentence ---> noun, verb.
verbal_sentence ---> verb, noun.
verbal_sentence ---> verb, noun, noun.
```

The purpose of the end-case analyser is to determine the prefix(es) and suffix(es) end-case marks of the given word according to its syntactic position. The semantics and knowledge module translates the syntactic output into a predicate or a group of predicates to find the answer. The knowledge was divided into two classes: subjects and verbs. Each class is represented in a way similar to semantic networks. The system's knowledge base domain was about human beings, animals, plants, and the interaction between them.

The system treats simple verbal sentences, simple nominal sentences, and simple interrogative sentences. It uses pattern matching to determine the types of the words in the user input [Al-Muhtaseb and Khayat 1988]. NAUS could handle questions like:

> هل سعد انسَان؟
hl s'd AnsAn?
is Saad human.
> مَاذ يئَكل حسين؟
mA_d y'akl .hsyn?
what eating Hussain.
What does Hussain eat?
> هل يتنفس؟
hl ytnfs?
is breathing.
Is he breathing?
> لِمَاذَا؟
lmA_dA?
why.

4.4 AQAS

The Knowledge Based Arabic Question Answering System (AQAS) [Mohammad et al. 1993] divides any query into two parts: a known part (the thing asked about) and a required part (the information requested), see an example bellow. The main function of the system is to extract the information from the given question that gives it the ability to retrieve the answer.

The parser first checks the input sentence looking for a noise word, like (من فضلك mn f.dlk = please) for example, which is to be ignored by the system. Then it converts the input query into an internal meaning representation (IMR) as a parse tree structure. Some morphological processes are performed to locate input tokens in the dictionary.

> مَا هي اعرَاض مرض الَاريثيمَا؟
>
> mA hy A'rA.d mr.d AlAry_tymA?
>
> what it symptom disease the Rethyma.

The AQAS internal meaning representation for the above question will look like:

Question-article	=["مَا mA","لِلعَاقل lil'Aql"]	
Requirement	=["عَرَاض'arA.d","عرض 'r.d"]	(informatin requested)
Known	=["الَاريثيمَا AlAry_tymA","مرض mr.d"]	(things asked about)

The IMR of AQAS is simple, looking for certain words in the query to specify the required information. The above meaning representation is processed by the interpreter to find the information that guides it to a generic frame to start with. The system's generator translates the output of the interpreter and displays it to the user.

4.5 QAS

QAS is a Question-Answering system for Arabic proposed by Yamani and Al-Zobaidie [1994 1996]. The system is based on using Lexical Functional Grammar (LFG) theory, introduced first by Kaplan and Bresnan [1982], to analyse Arabic interrogative sentences.

It consists of four modules: user interface, syntactic module, semantic module, and common-sense module. The system does not have a morphological analyser. QAS is domain dependent using stories which are stored in the domain object data base. The user interface allows the user to choose the story he or she needs to query. The syntactic module uses rules to produce the functional structure (F-Structure) for the input sentence.

The meaning representation for the input question is constructed in two stages. The first stage is to build the semantic structure (S-Structure) using the semantic module. The second one is to build the knowledge structure (K-Structure) using the common-sense module. The K-Structure is an extension of the S-Structure. This meaning representation is domain dependent.

The different structures of the interrogatives will follow the same behaviour pattern in the parsing process. In order to prove the ability of the system to retrieve answers, the interrogative structures are utilised as specimens [Yamani and Al-Zobaidie 1996]. The system is built to query a knowledge base which is implemented in a frame-based system.

4.6 Evaluation

ALI is a good prototype system for understanding Arabic input sentences and capturing inferences. Yet, it is restricted to questions which are stated in a simple manner and start with one of a limited number of interrogative

particles and pronouns. It is mainly concerned with a simple verbal sentence of the V-S-O form only where there are in fact three word orders for Arabic verbal sentences (V-O-S, O-V-S, and V-S-O). Therefore, the system handles the interrogative sentences which start with a limited number of interrogative tools followed by a simple verbal sentence. There is no treatment for questions which do not have any verb.

The complexity of ALI's semantic approach makes it domain dependent. Thus, the customisation process of ALI to a new domain would require totally new dictionaries and the modification of the grammar rules. Also, this approach makes the system take a longer time to establish the semantic cases for nominal sentences as no semantic cases are used to guide the process of interpreting noun phrases.

Khayat and his colleagues tried to address the problems of processing Arabic in general and propose solutions through the NAUS system. A comprehensive morphological analyser [Al-Uthman 1989] has been proposed and implemented in NAUS. In addition, the syntactic analyser is based on the adapted classification of Arabic sentences implemented in general rules in the Context Free grammar. However, these general rules cannot cope with all different aspects of Arabic sentences. For example, there are so many rules which govern the relation between the subject and the predicate in the nominal sentences.

The AQAS system gives a good focus on the query-phrase as it is the main source of information found in the question. Still, the system builds its meaning representation based on selecting the query key words without showing the relationship between them. In addition, the meaning representation is organised in a way which matches the knowledge base of the system which makes it domain dependent.

Yamani and Al-Zobaidie [1996] tried to propose what they called an interrogative-language framework through QAS using LFG theory. The advantage of their proposal is the separation between the semantic structure and the commonsense structure. Nevertheless, the problem of their proposal is in the treatment

of the co-ordinated interrogatives.

If there are two questions, one with a verb and the other one without, in a conjunction construct, the verb of the first sentence will be passed to the second sentence. Therefore, the following example:

> أَينَ ذَهَبَ المهندس أَحمد و أَينَ مُحَمَّد؟
> 'ayna _dahaba Almhnds 'a.hmd w 'ayna mu.hammad?
> where did the engineer Ahmed go and where Mohammed?

will be interpreted by passing the verb of the first question which is (ذَهَبَ _dahaba) to the second sentence as follows:

> أَينَ ذَهَبَ المهندس أَحمد و أَينَ - مُحَمَّد؟ (verb gap)
> 'ayna _dahaba Almhnds 'a.hmd w 'ayna - mu.hammad? (verb gap)
> where did the engineer Ahmed go and where - Mohammad?

There are two complete questions, in the above question, joined by the particle waw: "ayna thahaba ahmed" and "ayna mohammad". Here we cannot say the second question refers to the previous verb in the first question. Two Arabic nouns (ayna is a noun in Arabic) may represent a complete sentence without a need for a verb. The question "ayna thahaba ahmed" is about Ahmed who might have gone somewhere while "wa ayna mohammad" is about Mohammad where the hearer assumes that we are expecting him to be somewhere nearby.

4.7 Conclusion

Early systems dealing with Arabic interrogatives appeared in the eighties which is late compared with the work done in other languages such as English. Although, there are a number of interface systems for Arabic, research in natural language processing for Arabic is in its early stages.

A number of ideas have been proposed to treat some of the Arabic interrogative structures but, unfortunately, some of these treatments, for example the one which is proposed by Yamani and Al-Zobaidie [1994], are inappropriate for Arabic language phenomena.

All of these systems are domain dependent and do not consider portability as an objective. Thus, applying one of them to another domain will require almost rebuilding the system. Finally, most of these systems are designed to deal with knowledge bases rather than databases.

§§§§

Chapter 5

A GPSG Treatment for ARABIC

As shown in the previous chapters an inflectional language like Arabic, which has a different word order, needs a powerful theory to analyse its sentences both syntactically and semantically.

But before adopting a grammar formalism to process any natural language, several criteria should be considered. The most important criterion is linguistic felicity. This means that the formalism should analyse statements of the language based on the principles developed by linguists [Shieber 1986].

Based on this criterion and other criteria this chapter presents a description and presentation of Arabic interrogatives syntax using Generalised Phrase Structure Grammar (GPSG) [Gazdar et al. 1985] theory.

5.1 Background

There are a number of research effort that have attempted to describe Arabic syntax according to one of the modern linguistic theories. Snow [1965] used Transformational Grammar (TG) theory [Allen 1995] to give a grammatical description of the clause structure of modern written Arabic. The TG was also employed by Bakir [1980] to provide a descriptive account of the variation in word order in Arabic sentences.

Case Grammar [Fillmore 1968] theory has been used by Saad [1982] to study the semantic-syntactic properties of verbs in classical Arabic. Mehdi [1987] applied Definite Clause Grammar (DCG) [Pereira and Warren 1980] along with Case Grammar (CG) to parse Arabic simple sentences. Al-Shishiny [1995] also proposed a description for Arabic verbal sentences based on the DCG formalism.

A formal syntactic description of the noun phrase and verb phrase in modern standard Arabic has been proposed by Ditters [1992] using the Extended Affix Grammar formalism. Lexical Function Grammar (LFG) [Kaplan and Bresnan 1982] was employed by Yamani and Al-Zobaidie [1994] and Abu-Arafah [1995] for parsing Arabic. In addition, Alneami [1997] used an LFG to generate Arabic sentences.

Finally, Yusuf [1983] used GPSG [Gazdar et al. 1985] to study word order variation in Arabic. He did not study the O-V-S word order, as he considered any sentence of that type as ungrammatical. Also, his focus was only on the syntactic level of the Arabic sentence.

5.2 GPSG

Generalised Phrase Structure Grammar (GPSG) is a grammatical theory that was developed by Gazdar and co-workers within theoretical linguistics in the early 1980s. It falls into the general category of unification grammars which refer to the merging of information from different sources to produce an integrated description of some larger entity [Shieber 1986].

GPSG posits only one level of syntactic representation, i.e. surface structure [Sells 1986]. Based on Gazdar et al. [1985], GPSG grammar rules are decomposed into the following types of rules:

- *Immediate dominance rules* (ID-rules), which are similar to Context-Free Phrase Structure Grammar (CF-PSG) rules, give information about what the daughters of a node (the right hand side) can be.
- *Linear precedence statements* (LP-statements) which give information about the relative order of sister constituents. The LP-statements require this partial ordering to be respected by the linear order of sister categories.
- *Feature Co-occurrence Restrictions* (FCR) constrain the type of features that can occur together in a category.
- *Feature Specification Default* (FSD) to state a value as default for a feature whenever no specific value for the feature is required by a rule.
- *Metarules* which define new ID-rules based on the ones in the grammar. These rules capture redundancies among ID-rules.
- *Other constraints* such as Head Feature Convention (HFC), and Control Agreement Principle (CAP).

(1) $S \longrightarrow NP, VP$

The difference between the CF-PSG rules and the ID-rules is that the ID-rules specify no ordering among the various sub-constituents [Bennett 1995]. For example, the above ID-rule, rule (1), states that S can dominate NP and VP, but it does not tell any thing about the relative order of the daughters. The information about the relative order of these daughters is given by the following LP-statement:

(2) $NP \prec VP$

which states that NP must precede VP when they are sisters in a tree. GPSG introduces syntactic categories as sets of syntactic feature specifications, which are ordered pairs consisting of features and their values. The feature value is either atomic or a syntactic category which is a partial function from features to their values. Therefore, GPSG has two kinds of rules to govern the distribution of features.

(3) a. FCR: [VFORM] $\supset$ [+V,-N]
b. FSD: [+N,-V,BAR 2] $\equiv$ [ACC]

For instance, the above FCR rule (3a) says that if a category has associated with it the VFORM feature then it must be [+V,-N]. Whereas the FSD rule (3b) states that an N2 must be [CASE ACC] unless some specific statement requires another case and anything with the feature [CASE ACC] must be an N2 unless the contrary is specified.

(4)
$$\begin{array}{rcl} \text{VP} & \longrightarrow & W, \quad \text{N2} \\ & \Downarrow & \\ \text{VP[PAS]} & \longrightarrow & W, \quad (\text{P2}[BY]) \end{array}$$

GPSG employs the metarules concept as a way to capture redundancies among the ID-rules. The general syntax for metarules, as shown in rule (4), says: given a rule meeting pattern $A \longrightarrow \alpha$ there is another rule meeting pattern $B \longrightarrow \beta$. The metarule (4), for example, states that for every ID-rule which licenses VP to dominate N2 and some other material, indicated by W, there is another ID-rule that allows VP[PAS] to dominate the same material with

an optional P2[BY] but without N2.

Finally, the HFC and CAP are principles deployed in GPSG to govern the distribution of features. These general principles play an important role within the theory which make Sells [1986] state "GPSG is in fact a theory of how syntactic information 'flows' within a structure."

Of course, due to the limited space, we will not be able to go through all details of the theory here. However, an explanation of each formalised principle and its role in the general scheme are discussed in [Gazdar et al. 1985]. Also, a good introduction to the theory is in the [Bennett 1995], [Steurs 1991], and [Sells 1986].

5.2.1 Why GPSG Theory

There are several criteria that should be considered before adopting a grammar formalism to process any natural language. Shieber [1986] states three criteria for choosing the formalism to analyse a natural language. The first is linguistic felicity which means that the formalism should analyse statements of the language according to linguistic theories for the language. The second is expressiveness, which concerns what kind of analysis can be expressed in the grammar formalism. The third is computational effectiveness, focusing on limitations imposed by the machine on the analysis to be adopted for machine processing.

GPSG, with a few rare exceptions, can express all the structures of most natural languages. In addition, it has been used as a theoretical basis for analysing a number of phenomena in various languages [Humphrey 1992]. In fact, Shieber characterises it as the theory which has been especially successful in dealing elegantly with the subtleties of co-ordination phenomena and long-distance dependencies [Shieber 1986]

Furthermore, there are a number of systems, such as the Core Language Engine (CLE) [Alshawi et al. 1988 1991], Squirrel [DeRoeck et al. 1991, Fox 1995], EDS Natural Language Processing Toolkit system [Humphrey 1992], which have achieved good results in the field of NLP, and which were implemented using this theory.

Finally, in spite of the fact that Yusuf [1983] used a GPSG, where every syntactic rule in the grammar is to be associated with a semantic rule to determine the meaning of the constituent, his work focused only on the syntactic ID-rules of Arabic and did not discuss the semantic rules of Arabic sentences. Also, his study is limited to some of the Arabic word orders. Because he used only the ID-rules of the theory and did not use the rest of GPSG rules and notations, his study cannot be considered as presenting a complete GPSG analysis for Arabic.

Our study will present analyses for Arabic sentences using the full range of GPSG rules and notations as discussed in the following sections.

5.3 Declarative Sentence

As we have seen in Section 3.7, any interrogative sentence may contain a nominal sentence or a verbal sentence embedded in it. Therefore, it is better to describe these types of declarative sentences before discussing the interrogative sentences. The following sub-sections will discuss the nominal and verbal Arabic sentences in terms of the GPSG formalism.

5.3.1 Nominal Sentence

A nominal sentence is composed of two main parts: a topic (مبتدأ mbtda') , i.e. (مسند إليه msnd 'ilyh), and a comment (خبر hbr), i.e. (مسند msnd). The ID-

rules for sentences (5a-c) are in (6). Usually, the topic precedes the comment but sometimes the comment can precede the topic as in (5c). The LP-rules in (7) state this phenomenon.

(5) a. الولد طويل
Alwld .twyl
(the boy tall)
The boy is tall.

b. أَحمد في البيت
'a.hmd fy Albyt
(Ahmed in the house)
Ahmed is in the house.

c. في البيت أَحمد
(fy Albyt 'a.hmd)
in the house ahmed
In the house is Ahmed.

(6) a. $S \longrightarrow NP[+Top], NP[+Com]$
b. $S \longrightarrow NP[+Top], PP[+Com]$
c. $S \longrightarrow PP[+Com], NP[+Com]$

(7) a. $NP[+Top] \prec PP[+Com]$
b. $PP[+Com] \prec NP[+Top]$

The topic can only be a noun or noun phrase but not any other type of constituent such as a verb, verb phrase, prepositional phrase, and so on. This condition can be stated as a default specification feature for a non-noun or non-noun phrase as in (8). But there is no default feature for the N or NP since it can be a topic or a comment.

(8) a. FSD: PP $\supset$ $\sim$[+Top]
b. FSD: VP $\supset$ $\sim$[+Top]
c. FSD: AP $\supset$ $\sim$[+Top]

(9) a. FCR: NP ⊃ [+Top]
b. FCR: NP ⊃ [+Com]

The comment fulfils the predicate position in the nominal sentence and can be a NP, AP, PP, as in (6), or Ś[1], i.e. nominal or verbal sentence as in (10). In the case of a comment being a sentence it must have a referring item, such as a pronoun (ه h) in the sentence (أَبوه طويل 'abwh .twyl) of example (10a), to reflect the relationship between the topic and the comment. This means the comment sentence Ś[+Com] is not an independent sentence.

(10) a. حسين أَبوه طويل
.hsyn 'abwh .twyl
Hussain father his tall
Hussain's father is tall.
b. أَحمد درّس مَادة الحَاسب
'a.hmd drrs mAdT Al.hAsb
Ahmed taught course Computer

(11) a. $S \longrightarrow NP[+Top]$, Ś[+Com]
b. Ś[+Com] $\longrightarrow VP$

There is controversy about the analysis of a sentence like (10b). It has been analysed here, using the ID-rules (11), as a sentence starting with a topic, i.e. NP[+Top], followed by a sentential comment. On the other hand, it was analysed by Snow [1965], and Yusuf [1983] as a NP, recognised as the subject, followed by a VP, using the ID-rules (12). In fact, Snow used the S-V-O as the normal word order in his work.

(12) a. $S \longrightarrow NP[+subj]$, VP
b. VP $\longrightarrow V, NP[+obj]$

The main issue here is whether the subject can precede the verb in the Arabic

[1]Pronounced 'Sentence-bar' to indecate the maximal projection of the embedded sentence.

language[2]. Consider, for example, the first two sentences in (13a-b) which are in the V-S-O form. We can see that the verb form in (13a-b) remains unchanged even when the subject form is changed, i.e. being in a dual or plural form.

On the other hand, a dependent pronoun (ضمير متصل .dmyr mt.sl) has been added to the verb (ضرب .drb) of sentences (13c-d). The function of these dependent pronouns is to refer to a noun mentioned earlier. If the S-V-O word order is an acceptable Arabic word order, why are the verbs in (13c-d) changed, i.e. have a pronoun. In other words, if Arabic allows the subject to precede the verb, there is no need for the pronoun to be added to the verb. But if we look to the sentences in (14), there is no such pronoun attached to its verb. In this case, i.e. the singular case, a covert pronoun exists after the verb to refer to the subject.

(13) a. ضرب أَلرَّجل عَليًا
.drb 'al-rragl 'alyA
(hit the man Ali)
The man hit Ali.

b. ضرب أَلرِّجَال عَليًا
.drb 'al-rrigAl 'alyA
(hit the men Ali)
The men hit Ali.

c. أَلرَّجُلَان ضربَا عَليًا
'al-rragulAn .darabA 'alyA
The two men hit they Ali.

d. أَلرِّجَال ضربُوا عَليًا
'al-rrigAl .darabUA 'alyA
The nem hit they Ali.

---

[2]Linguistically, there is a debate on this issue which is beyond our focus here. More details can be found in Hassn [1975], Abn-Hesham [1985] and Abdullatif [1982].

(14) أَلرَّجل ضرب عَليًا
'al-rragl .drb 'alyA
The man hit Ali.

Therefore, the approach of preventing the subject from preceding the verb, is more suitable with the use of Arabic [Abdullatif 1982]. In terms of computational linguistics this approach gives a systematic way to parse Arabic sentences. Because, if the subject does not exist itself after the verb, it must be either an overt pronoun, in the case of dual or plural, or a covert, in the case of singular [Al-Johar and McGregor 1998].

(15) $NP[+Top] \prec \acute{S}[+Com]$

We have seen from the LP-rule (7) that the comment may precede the topic but this is not true for all types of comments. If the comment is a form of a sentence, it cannot precede the topic as the former should refer to the latter. That is the purpose of the LP-rule (15) above.

In Arabic the HFC of a nominal sentence is the comment. Therefore, based on the CAP, the topic must agree in number with the non-prepositional phrase comment, i.e. if the comment is a noun, noun phrase, and sentence. They also should agree in gender if the comment is a noun or noun phrase. But there is no gender agreement if the comment is a sentence or a prepositional phrase.

(16) a. أَحمد درّس روَان
'a.hmd drrs rwAn.
Ahmed taught Rwan

b. أَحمد تدرّسه روَان
'a.hmd tdrrsh rwAn.
Ahmed teaching him Rwan

Fassi Fehri suggested that the NP agrees with the VP in number and gender as in (16a) [Fassi Fehri 1988]. This is not always the case. Consider, for example,

the verb in (16b) where there is no gender agreement between the verb and the preceding noun. His suggestion may be based on the recognition of the first nominative noun as the subject of the following verb. Again this is another problem of the consideration of S-V-O as an alternative word order for Arabic.

5.3.2 Verbal Sentence

The verbal sentence is one that starts with a verb. It is composed of at least two essential parts (عمدة 'mdah): a verb (فعل f'l), i.e. (مسند msnd) which expresses the temporal action or condition, and an agent (فَاعل fA'l), i.e. (مسند إليه msnd 'ilyh) to which the verbal action is attributed. The verb may have no object, or one or more objects based on its type, i.e. intransitive, transitive, or bitransitive. The ID-rules in (18) are used in analysing the sentences in (17).

(17) a. رسب عَلي
rsb 'aly
failed Ali

b. درس أَحمد الكتَاب
drs 'a.hmd AlktAb
studied Ahmed the book

c. درّس أَحمد عليًا الكتَاب
drrs 'a.hmd 'lyA AlktAb
taught Ahmed Ali the book

(18) a. $S \longrightarrow VP$

b. $VP \longrightarrow V, NP[+subj]$

c. $VP \longrightarrow V, NP[+subj], NP[+obj]$

d. $VP \longrightarrow V, NP[+subj], NP[+obj], NP[+obj]$

Based on the discussion in the above section, the first LP-rule here, see rule (19a), is to clear the position of subject to be after the verb. One default

specification feature for any completed verb of Arabic is that it must have a subject if it is in the active form, as in the FSD-rule (19b), while if the verb is in the passive form, the object will act as subject, i.e. deputy subject. In addition, the intransitive verbs cannot take NP as an object as stated in the FSD-rule (19c). Of course it can sometimes have an object in preposition phrase form as in the FCR-rule (19d).

(19) a. $V \prec NP[+subj]$

b. FSD: [+V,-N,BAR 2,+Active] $\supset NP[+subj]$

c. FSD: [+V,-N,BAR 2,+Intrans] $\supset\sim NP[+obj]$

d. FCR: [+V,-N,BAR 2,+Intrans] $\supset PP[+obj]$

(20) الأَوْلَادُ أَكلوَا الكعكة

Al'awlAdu 'aklwA Alk'kT

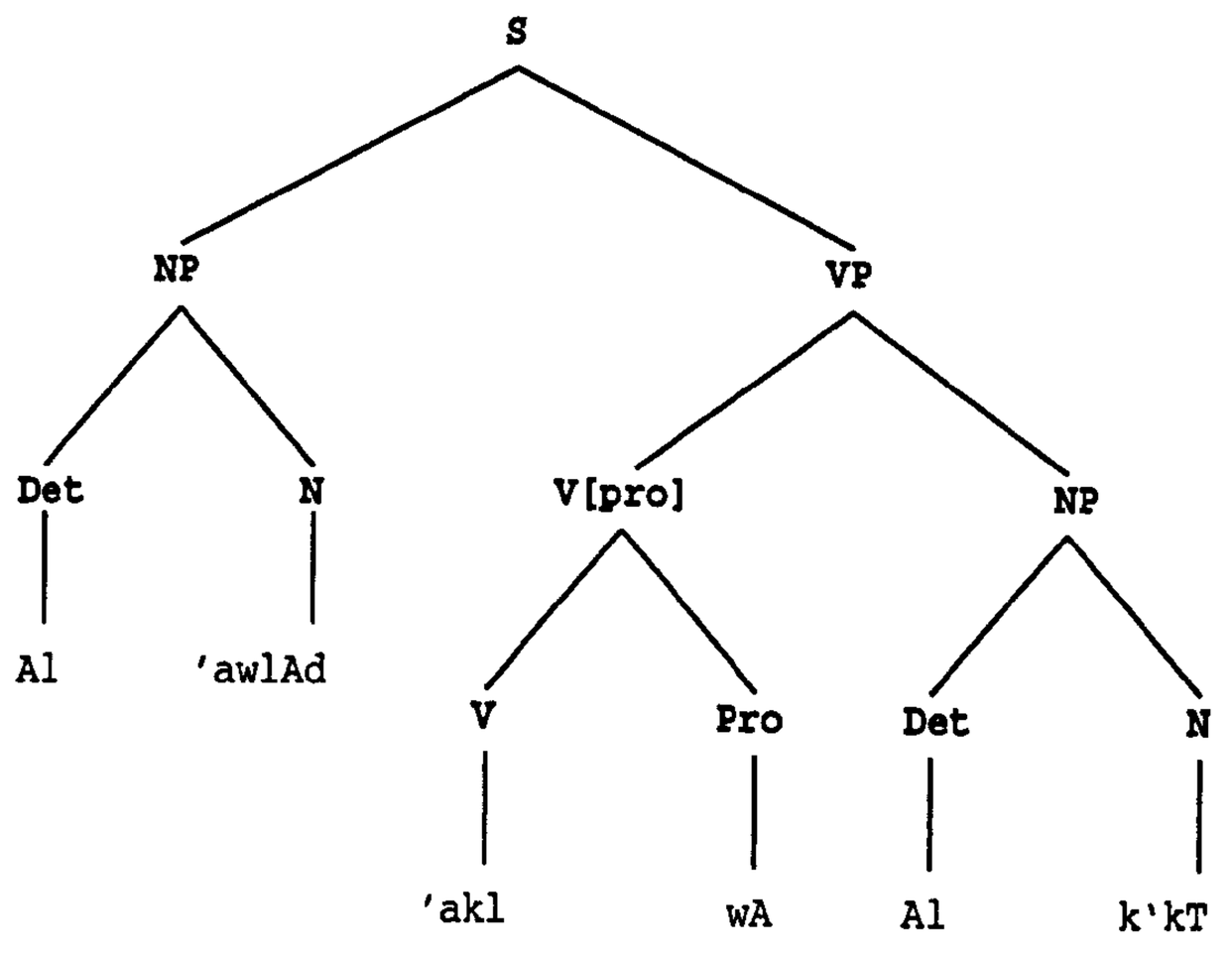

Figure 5.1: The Yusuf's parse tree for sentence (20)

The above rules treat the verbal sentence only when the subject is not an anaphor. If the subject is a pronoun, i.e. either overt as in (20) or covert as in (14), the first thing that needs to be stated is the constituent structure. The tree in Figure 5.1 is one possible structure proposed by Yusuf [1983]. He puts the pronoun as a part of the structure of V. Therefore, There are two types of

verb structure shown in his grammar: $V[+pro]$ which contains a pronoun and $V[-pro]$ where there is no pronoun.

Yusuf's proposal may be due to the use of the S-V-O as a word order. The recognition of the first noun as a subject makes him recognise the following pronoun of the verb and the verb as one item. If a dependent pronoun is attached to a verb, it is only to refer to a noun or noun phrase which precedes that verb. In fact, besides the phonological structure, any pronoun following the verb has nothing to do with the verb. These pronouns are representative of their subject or object because they have the same features, i.e. number, person, gender, case, and so on, of the thing they refer to. Therefore, they should be governed by the verb as a subject or object under the VP maximal projection of the verbal sentence. So, the constituent structure of sentences like (20) is shown in Figure 5.2.

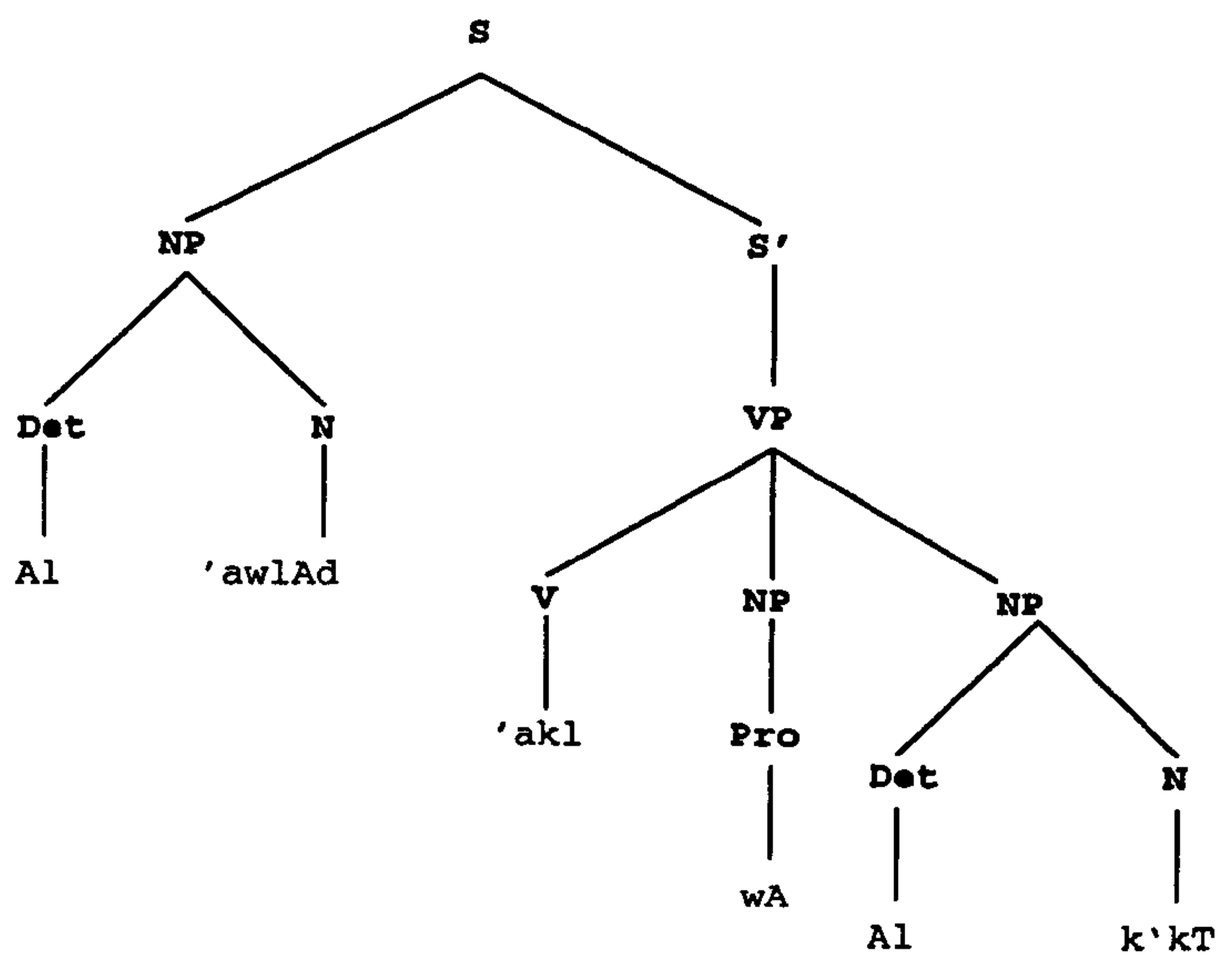

Figure 5.2: The parse tree for sentence (20)

When there is no noun or pronoun appearing as a subject of the verb, as in example (14), the subject considered, in this case, is a covert pronoun. The covert pronoun will be recognised as the independent pronoun "he" (هو hw), if the verb is in a masculine form, or "she" (هي hy), if the verb is in a feminine

form. The syntactic structure of (14) will be discussed further when we talk about unbounded dependencies, Section 5.5.

The verbal sentence may contain an object complement for the verb. The object complement is considered a nonessential part of the verbal sentence. It is just there to improve the meaning of the sentence and that is why the Arab grammarians call it as (فضلة fdhlah) [Hassn 1975]. It can be a noun, noun phrase, adjective phrase, or relative clause. While the subject must not be within a prepositional phrase constituent, based on the FSD-rule (21c), the object can be in a form of PP as in the FCR-rule (21d). It can also be an overt pronoun but unlike the subject, it cannot be a covert pronoun as can be seen in the rules (21a-b).

(21) a. FCR: Pronoun[+subj] $\supset$ [+covert]
b. FSD: Pronoun[+obj] $\equiv \sim$[+covert]
c. FSD: [+PP] $\supset \sim$[+subj]
d. FCR: [+PP] $\equiv$ [+obj]

The verbal sentence requires an agreement between the verb and the subject. The verb should agree with the subject in gender only if the subject is a real feminine or a pronoun that refers to a real feminine. There is no agreement in number between the verb and the subject.

5.4 Interrogative Sentence

There are two types of question in Arabic: Yes/no questions and WH questions. These have a different constituent structure and meaning, see 3.7. Therefore, each type will be discussed in a separate sub section, as follows.

5.4.1 Yes/no Questions

This type of question should start with a question article, either hamzah (أَ a) or (هل hl) as in (22a) and (22c). These articles cannot be in the middle of the sentence as in (22b) and (22c)[3]. Figure 5.3 shows the parse tree for sentence (22a).

(22) a. أَ درّس زيَد الطلَاب؟
'a drrs zyd Al.tlAb?
(did taught Zyd the student)
Did Zyd teach the students.

b. *؟نجح أَ علي
nj.h 'a 'ly?
passed is Ali

c. هل أَحمد في الفصل؟
hl 'a.hmd fy Alf.sl?
Is Ahmed in the class.

d. *أَحمد هل في الفصل؟
'a.hmd hl fy Alf.sl?
Ahmed is in the class

It can be noticed from Figure 5.3 that this type of sentence is not like the previous types of sentences, i.e. there is no such a (مسند-مسند إليه msnd 'ilyh-msnd) contrast between the article (أَ 'a) and the rest of the sentence. Therefore, the ID-rules for these types of questions are in (23) which state that a yes/no question consists of an interrogative article and a complete sentence, either verbal or nominal. These rules are governed by the LP-rule in (24) which forces the question article to be at the beginning of the sentence.

[3]The star indicate the sentence is incorrect grammatically.

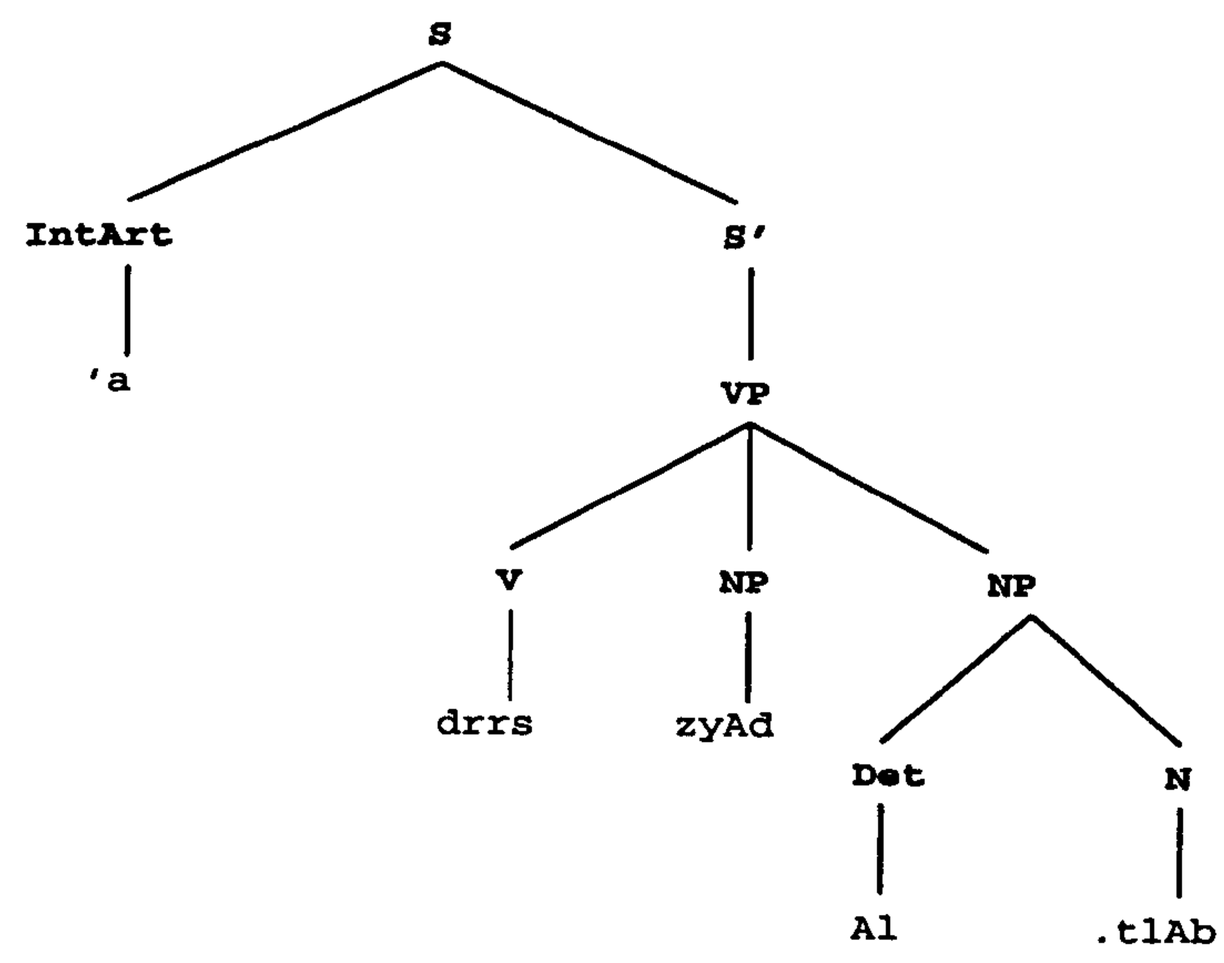

Figure 5.3: The parse tree for sentence (22a)

(23) a. $S[+yesno] \longrightarrow IntArt$, Ś[+nominal]
b. $S[+yesno] \longrightarrow IntArt$, Ś[+verbal]

(24) $IntArt \prec$ Ś

Although, there is some similarity between these articles, they are not alike. They differ in their use. The humza (أ a) can be used for (تصديق t.sdyg), which means question its validity, or (تصور t.swr), which means emphasise the statement. On the contrary, the (هل hl) can be used only for (تصديق t.sdyg), i.e. question its validity. This means that the (هل hl) cannot take a negative nominal or verbal sentence. That is why sentences (25a) and (25b) are correct while sentence (25c) and (25d) are incorrect.

(25) a. أ ليس زيد يدرس مَادة بَاسكَال؟
'a lys zyd ydrs mAdT bAskAl
is not Zyd studying course Pascal

(25) b. أَ لم ينجح سَامي؟
'a lm ynj.h sAmy?
is not pass Samy

c. *؟هل ليس أَحمد في الفصل
hl lys 'a.hmd fy Alf.sl?
is not Ahmed in the class

d. *؟هل لم يرسب أَسَامة
hl lm yrsb 'asAmT?
is not fail Osamh

(26) a. FSD: IntArt[+Tsd,-Tsw] $\supset \sim$ S[+neg]
b. FCR: IntArt[+Tsw,+Tsd] $\supset$ S[+neg]
c. FCR: IntArt[+Tsw,+Tsd] $\supset$ S[-neg]

The boolean features Tsw and Tsd are used to distinguish (هل hl) and (أَ 'a) articles; the (هل hl) has the feature [+Tsd,-Tsw] while the [+Tsd,+Tsw] feature is for the (أَ 'a). Based on (25c-d) there is a need for the default specification rule of (26a) to insure that the (هل hl), which comes only for validity (تصديق t.sdyg), cannot be followed by a negative sentence. On the other hand, the FCR-rules (26b) and (26c) state that the (أَ 'a) can be followed by a negative or positive sentence.

(27) FSD: IntArt[+Tsd,-Tsw] $\supset \sim$ VP[+Com]

Another difference between these articles is in the constituent structure of their following sentence. The interrogative article (أَ 'a) can be followed by a verbal sentence or a nominal sentence with no restriction on the type of its comment. On the contrary, the (هل hl) cannot be followed by a nominal sentence which has a comment as a verbal constituent. This phenomenon is described in the FSD-rule (27).

5.4.2 WH Questions

These types of questions have an interrogative pronoun, which introduces and indicates the element of the sentence about which the question is asked. Unlike the yes/no questions, these types of questions are presented in a form of topic (مبتدأ mbtda') and comment (خبر_hbr) constituents. In other words, there is a (مسند-مسند إليه msnd 'ilyh-msnd) contrast between the interrogative pronoun and the rest of the sentence as shown in Figure 5.4. (29) is the ID-rule for a query like (28a). Even if the comment is in the form of a sentence, like the verbal sentence (مَاذَا يدرس أَحمد في قسم الحَاسب؟ ydrs 'a.hmd fy qsm Al.hAsb)in (28b), still it will be recognised as a comment because this sentence is not an independent sentence, i.e. it has some reference to the element that is mentioned in the topic which is (مَاذَا mA_dA).

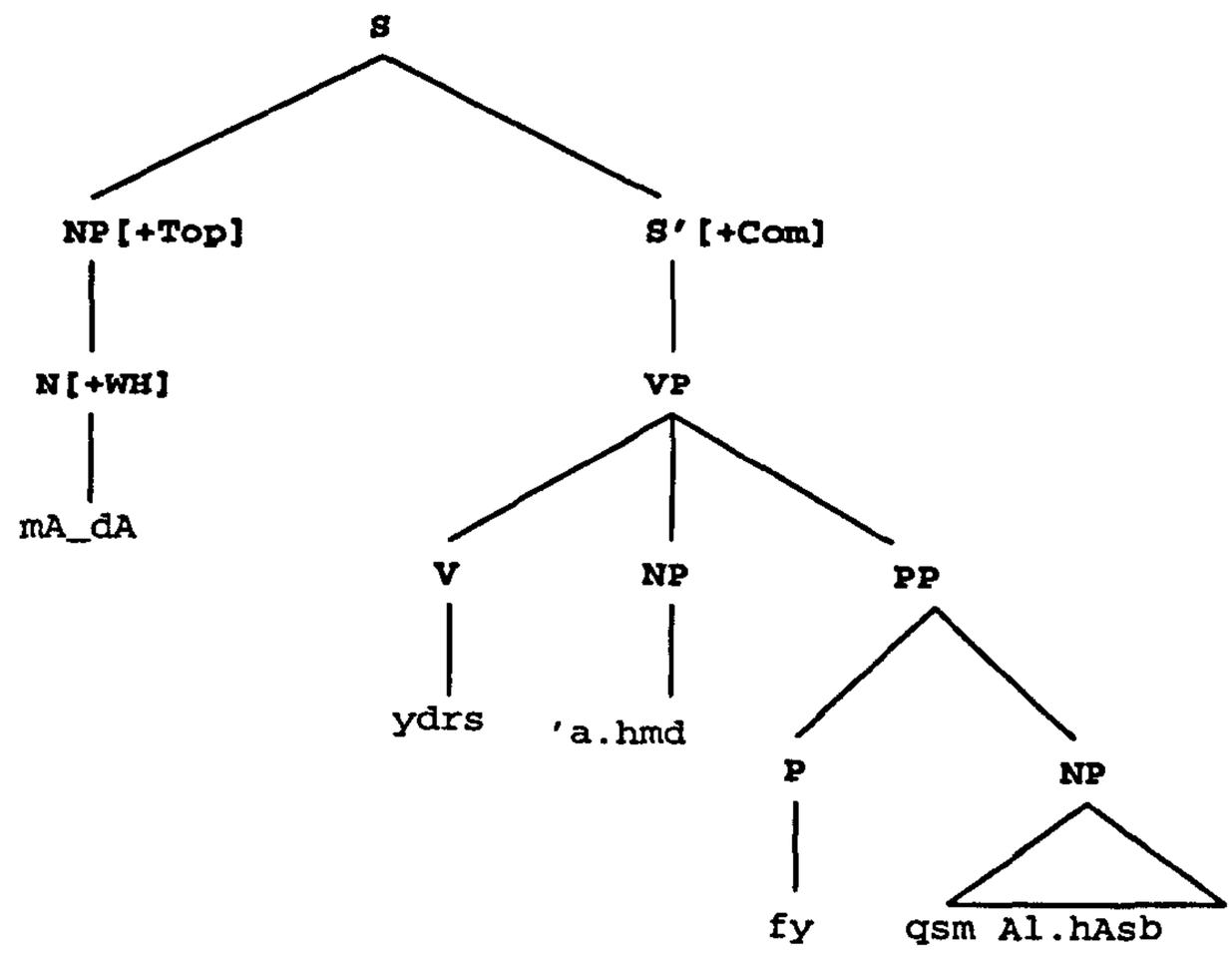

Figure 5.4: The parse tree for sentence (28b)

(28) a. مَا إِسم الطَالب رقم ٩٨٦٥٥٤؟
mA 'ism Al.tAlb rqm 986554?
what name the student number 986554

b. مَاذَا يدرس أَحمد في قسم الحَاسب؟
mA_dA ydrs 'a.hmd fy qsm Al.hAsb?
what studying Ahmed in department Computer

(29) $S \longrightarrow NP[+Top, +WH], NP[+Com, -WH]$

These interrogative pronouns usually have the priority to be at the beginning of the sentence. This raises a need for the following LP-rules:

(30) a. $NP[+WH] \prec NP[-WH]$
b. $NP[+WH] \prec VP$
c. $NP[+WH] \prec PP[-WH]$
d. $NP[+WH] \prec ADV[-WH]$

(31) FSD: VP $\supset \sim$ [WH]

The WH-pronouns have different co-occurrence restrictions. The (من mn), for example, is used for asking about human while (مَا mA) and (مَاذَا mA_dA) are used for non-human, i.e. things as in (33) FCR-rules. The (من mn), (مَا mA), and (مَاذَا mA_dA) may be followed by an independent personal pronoun, as in (32a) for instance, in this case the personal pronoun will be realised as a resumptive repetition of the interrogative pronoun.

(32) a. من هم طلَاب مَادة قوَاعد البيَانَات
mn hm .tlAb mAdT qwA'd AlbyAnAt
who they students course Database

b. في أَي قسم يدرس محمّد؟
fy 'ay qsm ydrs m.hmmd?
in which depatment studying Mohammed

(33) a. FCR: [+N,-V,+WH,+mn] ≡ [+Hum]
b. FCR: [+N,-V,+WH,+mA] ≡ [-Hum]
c. FCR: [+N,-V,+WH,+mA_dA] ≡ [-Hum]

In addition, the (من mn), (مَا mA), and (مَاذَا mA_dA) may be preceded or followed by a preposition but they cannot take a dependent genitive. On the other hand, the (أَي 'ay) is usually followed by a genitive, as shown in the (34a) FCR-rule, which can be in a definite or indefinite form, in a singular or plural form.

(34) a. $FCR: [+N, -V, +WH, +ay] \supset [+N, -V, +Gen]$
b. $FCR: [+N, -V, +WH, +km] \supset [+N, -V, +Sing, +Acc, +Indef]$

(كَم km) is always found at the beginning of the sentence because of its indefinite character. It may be more closely determined by an accusative singular function in indefinite form as an accusative of specification. This is stated in FCR-rule (34b). Questions about any circumstances or adverbial aspects modifying the statement should be introduced by (متَى mt_A), for time, (أين ayn), for place, and (كيف kyf), for adverb. Rules similar to the (33) co-occurrence restriction rules can be stated for the latter interrogative pronouns based on their attributes.

(35) a. $Article[+Coord] \prec NP[+WH]$
b. $NP[+WH] \prec\sim Article[+Coord]$
c, $PP[+Ques] \supset NP[+WH]$

Finally, all of the WH-pronouns may be preceded by a co-ordination article but they cannot be followed by a co-ordination article. This phenomenon is presented in (35a) and (35b) LP-rules. In addition, these pronouns may occur in the prepositional phrase contrast as in sentence (32b) above which needs the FCR-rule (35c).

5.5 Unbounded Dependencies

Unbounded dependencies is a term which was introduced by Gazdar [1981] to refer to a class of construction of wh-movement or relative clauses. For example, the question in (36) cannot be treated as straightforward instances of NP + VP. This is because the wh-phrase is not understood as the subject of the immediately following verb. It may be object of the next verb as in (36). In fact, the relation between the wh-form and its "understood" position may be of any distance.

(36) Which team did they beat?

Unbounded dependencies occur in different constructions in Arabic; see the examples in (37). These constructions have the property that there is a missing phrase within a main clause.

(37) a. محمَّد ضرب الوَلد
m.hammad .drb Alwald
Mohammed hit the boy

b. من الّذي رأَيت؟
mn All_dy r'ayt?
who that you saw

If something is missing from a constituent, it can be encoded in GPSG by means of a feature. GPSG employs SLASH categories to pass information around the tree, i.e. between the displaced position and the missing position, to maintain the agreement.

In (37a) there is no noun or pronoun that appears as a subject of the verb in the comment, i.e. the verbal sentence (ضرب الوَلد .drb Alwald). Therefore, the subject is recognised as an empty element. Similarly, the transitive verb (رأَيت r'ayt) in the relative clause of (37b) needs an object which does not

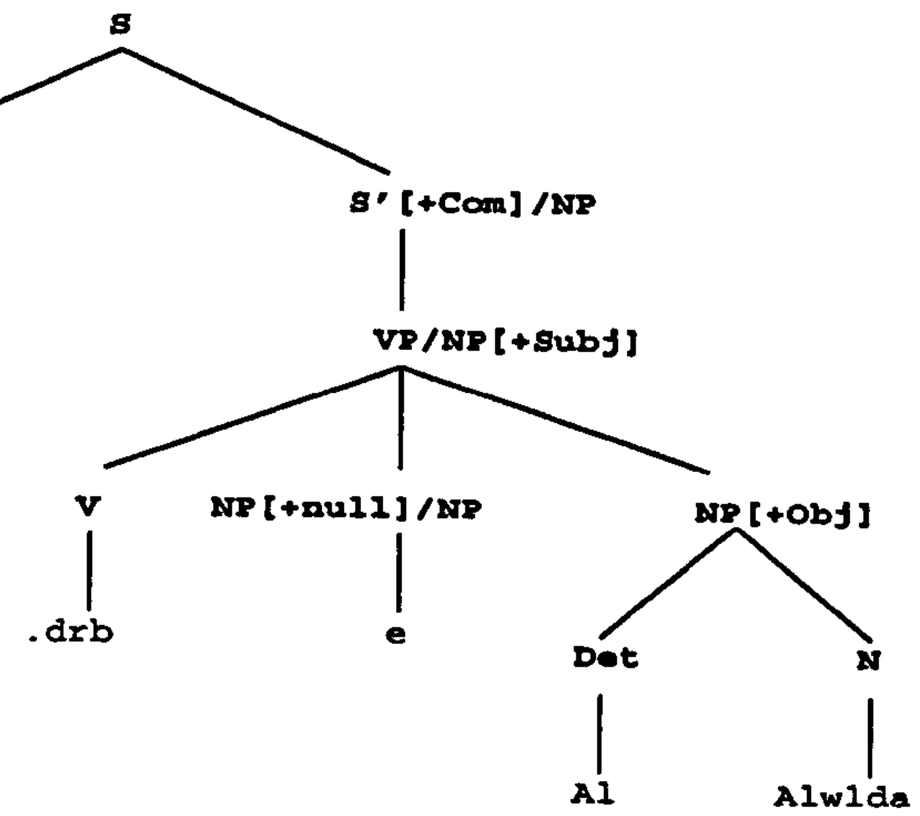

Figure 5.5: The parse tree for sentence (37a)

exist within the clause. The syntactic analysis of (37a) can be seen in Figure 5.5.

5.6 Conclusion

Although several syntactic studies have tried to apply modern linguistic theories to analyse Arabic statements, there is a lack of awareness about whether the syntactic analyses of these studies were based on sound linguistic principles.

This study uses GPSG to analyse Arabic statements. The focus is mainly to provide the syntactic analyses based on correct Arabic linguistic principles. This study has shown that GPSG is a powerful syntactic theory not only for Indo-European languages but also for Semitic languages such as Arabic. In GPSG, every syntactic rule in the grammar is to be associated with a semantic rule to determine the meaning of the constituent. Therefore, the following chapter will discuss the semantic issues.

§§§§

Chapter 6

Formal Semantics For Arabic

6.1 Introduction

GPSG [Gazdar et al. 1985] makes use of a kind of model-theoretic semantics called Montague semantics, sometimes called formal semantics. It should be noted that Montague semantics is built around the concept of functions, i.e. the verb for example, and arguments, i.e. the subject and the object. Another important note is that this theory has been mainly applied to English and some other Indo-European languages where there is at least a verb or copular verb in the sentence. But as we have seen in Chapter 3 Arabic is different from English in word order and sentence constituency, i.e. not all sentences have a verb or copula verb for instance.

Arab grammarians proposed a theory called (إسنَاد 'isnAd) to describe the functional role of the sentence's parts, see Section 3.6. It gives some semantic relationship between these parts, i.e. how each part is related to the others. (سيبويه Sybwyh) studied the relationship of each word to the others in Arabic sentences and developed a theory called (العَامل Al'Aml) meaning governance. His theory makes a distinction between a regent (عَامل 'Aml), a

dependent (معمول m'mwl), and the relation (تعلّق t'llq) between them. He classified Arabic words in terms of their governance which appears as a vowel or nunation, i.e. (العلَامة الاعرَابية Al'lAmT AlA'rAbyT), in the other word or words [Amaireh 1987].

These theories, however, do not describe a complete formal semantics for the constituents of Arabic statements. In addition, they have no mechanism for constructing a meaning representation. From the computational literature review, see Chapter 4, logical meaning representation for Arabic sentences has received very little work with no indication of what are the major problems and how they can be tackled [Al-Johar and McGregor 1997].

Henceforth, this chapter will discuss the idea of building a logical meaning representation for Arabic statements. The approach is to propose well-formed expressions for Arabic statements based on formal semantics. The Arab grammarians' theories will be used as a guide to assure validity of the treatment for Arabic.

In terms of the type of function tools that might exist, we classified Arabic sentences into three groups. The first group consists of sentences which do not have verbs and they contain other function tools such as quantifiers, adjectives, prepositions, verbal nouns, and so on. The second group of sentences contains verbs in their constituents and may also have the function tools of the first group. The third group are sentences which contain only individuals, i.e. an individual constant or an individual variable, and there is no kind of function. Each group needs a specific semantic treatment. This chapter will classify Arabic sentences into: verbless sentences, verbal sentences, and 'sentences of individuals'.

This classification does not mean that the semantics will be built away from the syntactic rules or that there is no relation between the syntactic rules and the semantic rules. Each semantic rule will interpret the structure provided by the syntactic rule associated with it.

6.2 Verbless Sentences

Unlike English, Arabic nominal sentences may not have any kind of verb. Sentences in (1), for example, contain no verb in their constituents and yet they are correct. This type of sentence has been studied by a number of researchers in order to find a suitable analysis based on modern linguistic theories.

(1) a. زيد في المدرسة
zyd fy AlmdrsT
(Zyd in the school)
Zyd is in the school.

b. في البيت محمّد
fy Albyt m.hmmd
(in the house Mohammed)
Mohammed is in the house.

c. العلم مفيد
Al'lm mfyd
(the science good)
The science is good.

Bakir [1980] analyses a verbless sentence as a verbal sentence by positing a copular verb (كَانَ kAn = to be) between the sentence-initial NP and the constituent that follows. He formulates the following transformational rules for this type of sentence:

(2) a. $S \longrightarrow NP \quad VP$
b. $VP \longrightarrow V[COP] \quad NP$
c. $VP \longrightarrow V[COP] \quad PP$

Fassi Fehri [1987] also analysed them based on the same hypothesis. He uses the V-S-O (verb-subject-object) as the basic word order not only for the verbal sentences but also for nominal sentences. He analyses any verbless sentence as if there is a copular verb, invisible on the surface, at the beginning of the

sentence, as shown in the following transformation rules:

(3) a. $S \longrightarrow V \quad NP \quad NP$
b. $S \longrightarrow V \quad NP \quad PP$

Although the copular hypothesis does make an easy analysis, especially in semantics, for this type of sentence, it is not based on the theories prefered by the linguists. For example, when the copular verb is positted in a nominal sentence, it makes the topic, called its noun (أَسم كَان 'asm kAn), in the nominative form and changes the predicate to be in the accusative form, called its predicate (خبر كَان ـhbr kAn). So, if there is a copular deletion, from the surface, in this type of sentence, the predicate should always be marked in the accusative form. But by viewing this type of sentence, see examples above, there is no indication for a copular deletion, i.e. the predicate is always in the nominative form.

These sentences consist of topic (مبتدَأ mbtda') and comment (خبر ـhbr). The topic (مبتدَأ mbtda') must be a type of noun phrase. There are various types of noun phrases: indefinite and definite noun phrases, quantified noun phrases, and noun phrases of verbal nouns. On the other hand, the comment (خبر ـhbr) could be a noun phrase, an adjective phrase, or a preposition phrase.

The principle of compositionality which is assumed by formal semantic accounts of NL states that the interpretations of the parts of a construction are combined by applying a function to its arguments to yield the interpretation of the whole construction. Then the result may be combined with other parts of the sentence [Gazdar et al. 1985]. Consequently, our work is based on examining each phrase construction, using (العَامل Al'Aml) theory, to propose a suitable treatment within the model-theory framework.

At the sentence level there are two options: either the first phrase is applied as a function to the second phrase or the second one is applied to the first one. The first option is adopted by Montague [Dowty et al. 1981] while the second

option is used by Gazdar et al. [1985]. Based on the (إِسنَاد 'isnAd) theory we will treat the comment phrase as denoting a function from the topic to the sentence.

6.2.1 Indefinite Noun Phrases

Common nouns extensionally denote sets of entities. For example, the extension of a noun like (طَالب .tAlb 'student') is the set of all students. Therefore, a common noun must be a type like $< e, t >$, a set-denoting expression. As mentioned before, Section 3.2.3, Arabic has no article for presenting indefiniteness. This means that an indefinite noun phrase will be presented in the form of a one-place predicate, like ".tAlb(X)".

But sometimes an indefinite common noun may be followed by an adjective (صفة .sfT) as a qualification (نعت n't), see (4a) for an example. In this case, we will get two open formulae each one containing free variables as in (4b). These formulae are not well-formed formulae based on formal semantics. The free variables in the open formulae must be bound by a quantifier appropriate to a connective, like $\wedge$, $\vee$, $\rightarrow$, and $\leftrightarrow$. Therefore, a special semantic rule for such a phrase is proposed in (4c), to bind the free variables and present the connective, to give the well-formed expression in (4d) for that sentence.

(4) a. أَحمد طَالب مجد
'a.hmd .tAlb mĝd.
(Ahmed student serious)
Ahmed is a serious student.

b. .tAlb'('a.hmd') mĝd'('a.hmd')

c. $\lambda p[CN(p) \wedge ADJ(p)]$

d. $\exists x[.tAlb'(x) \wedge m\hat{g}d'(x)]$

6.2.2 Definite Noun Phrases

Common nouns can be defined by the prefix definite article (ال Al 'the'). In formal semantics, the determiner is treated as a functor over the noun, as in (5a), and given the translation in (5b). A definite phrase like (5c) will be interpreted by the (5a,5b) rules to give the semantic formula in (5d).

(5) a. Det'(N')

b. Det': $\lambda P[\lambda Q \exists y[\forall x[P\{x\} \leftrightarrow x = y] \wedge Q\{y\}]]$

c. الطَالب

Al.tAlb

the student

d. $[\lambda Q \exists y[\forall x[.tAlb'\{x\} \leftrightarrow x = y] \wedge Q\{y\}]]$

The translation of the determiner (ال Al 'the'), in (5b), consists of two open formulae: one formula is derived by applying the translation of the complement of the determiner while the other is derived by applying the translation of the other phrase of the sentence to an instance of the same variable. These two formulae are combined by the prepositional connective, $\wedge$, to form a well-formed formula (wff) with no free variables. Based on that, a verbless sentence like (6a) is given the semantic representation in (6b).

(6) a. الطَالب مجد

Al.tAlb mĝd.

(the student serious)

The student is serious.

b. $\exists y[\forall x[.tAlb'(x) \leftrightarrow x = y] \wedge m\hat{g}d'(y)]$

The translation of the determiner (ال Al 'the'), in (5b), states that P and Q are of type $< e, t >$ which means they cannot be individuals. In Arabic, it is possible to have one of these formulae containing only an individual. For example, sentence (7a) consists of a noun phrase for an individual variable as a topic, which contains only the interrogative pronoun (من mn), and a definite

noun phrase as a comment. If we supply this sentence to the translation rule, it will produce the semantics in (7b). But the semantics of (7b) is inappropriate as it leaves the second formula open, i.e. containing a truth-predicate made of two arguments, i.e. a variable fva representing the individual (من mn), becomes a functor applied to another variable.

(7) a. من الطَالب
mn Al.tAlb.
who the student

b. $\exists y[\forall x[.tAlb'(x) \leftrightarrow x = y] \wedge fva(y)]$

c. $\lambda P[\lambda Q \exists y[\forall x[P\{x\} \leftrightarrow x = y] \wedge hw'(Q\{y\})]]$

d. $\exists y[\forall x[.tAlb'(x) \leftrightarrow x = y] \wedge hw'(fva, y)]$

In order to make a well-formed expression for (7a), we need to add a functor which presents the relationship between the arguments. We need either to add the functor within the semantic lexical entry for individuals or to add it within the semantics of the determiner. The first option is not practical as it will cause a double entry for each individual while the second one will require another semantic entry for the determiner. The second option has been chosen. Thus, by applying the (7c) translation rule to the previous sentence we get the representation of (7d).

(8) a. من الطَالب زيد
mn Al.tAlb zyd.
who the student Zyd

b. $\exists y[\forall x[.tAlb'(x) \wedge zyd'(x) \leftrightarrow x = y] \wedge hw'(fva, y)]$

c. $\exists y[\forall x[.tAlb'(x) \wedge hw'(x, zyd') \leftrightarrow x = y] \wedge hw'(y, fva)]$

(9) a. من الطَالب رقم ٩٨٢٣١
mn Al.tAlb rqm 98231.
who the student number 98231

b. $\exists y[\forall x[.tAlb'(x) \wedge rgm'(x, 98231') \leftrightarrow x = y] \wedge hw'(y, fva)]$

A definite common noun may also be complemented by explicative apposition

(عطف بيَان ‘.tf byAn), as in (8a) for example, or premeditative (بدل bdl) as in (9a). In this case we need to expand the semantics of the common noun, to include its complement and share the instance of the same variable. The expansion cannot be at the lexical type level but it can be at the phrase level. The semantic rule for the common noun in such a phrase will look like $\lambda x[CN(x) \wedge COM(x)]$ which gives the (8b) and (9b) output. It can be noted that this rule produces a well-formed expression for a sentence like (9a) but is inappropriate for (8a) as (8b) contains an individual argument as a functor for another individual variable. This problem can be solved by making another rule similar to $\lambda x[CN(x) \wedge hw'(COM(x))]$, which is applied only when the common noun of a definite phrase is complemented by individuals, to produce the (8c) semantics.

6.2.3 Quantified Noun Phrases

A quantified phrase is one which contains a quantifier, such as (كل kl = all), (جميع ĝmy‘), and so on. These quantifiers must be determined by another noun in a genitive form as a determining word (المضَاف إليه Alm.dAf ’ilyh). If the determining noun (المضَاف إليه Alm.dAf ’ilyh) is indefinite, as in (10a), the quantifier will have a primary meaning in the sentence. The quantifier (كل kl) in (10a), for example, refers to all items of the entity set of (طَالب .tAlb) in a given domain as given in the (10b) semantics.

(10) a. كل طَالب نَاجح في مَادة الحَاسب الآلي
kl .tAlb nAĝ.h fy mAdT Al.hAsb AlAly.
all student passed in course Computer Science

b. $\forall x[.tAlb'(x) \rightarrow \exists a[mAdT'(a) \wedge fy'(x, nA\hat{g}.h', a)]]$

Contrary, when the quantifier is followed by a definite noun, it will not give any primary meaning to a given sentence. In (11a), for example, there are two quantifiers following each other: (كل kl) and (ال Al). If the semantics of

(11a) is built based on the semantic formula for both of these quantifiers, this will give unnecessarily complicated semantics, as in (11b), for such a sentence. Arab grammarians stated that a quantifier like (كل kl) in such a case comes as corroboration (التوكيد Altwkyyd) [Abn-Hesham 1985] which means it is just to confirm an existing meaning. If we compare sentence (11a) with the one in (12a), they will give the same inferences. Therefore, we choose to build the semantics of sentences like (11a) based on the semantics of the determiner only, i.e. ignoring the quantifier semantics. This will give a well-formed formula (wff) as in (11c) for this kind of sentence.

(11) a. كل الطلّاب نَاجحون
kl Al.tlAb nAĝ.hwn
all the students passed

b. $\forall z[\exists y[\forall x[.tlAb'(x) \leftrightarrow x = y] \wedge nA\hat{g}.h'(y)].tAlb'(z) \rightarrow nA\hat{g}.h'(z)]$

c. $\exists y[\forall x[.tlAb'(x) \leftrightarrow x = y] \wedge nA\hat{g}.h'(y)]$

(12) a. الطلّاب نَاجحون
Al.tlAb nAĝ.hwn
the students passed

b. $\exists y[\forall x[.tlAb'(x) \leftrightarrow x = y] \wedge nA\hat{g}.h'(y)]$

6.2.4 Verbal Noun Phrases

These phrases are constructed with verbal nouns as a head. A verbal noun is a word that is derived from a verb, but has the characteristics of a noun or adjective. These verbal nouns are used as if they were verbs. there is no equivalent in English. The verbal nouns are: nomen agentis (إسم الفَاعل 'ism AlfA'l) like (نَاجح nAĝ.h = passed) , nomen patientis (إسم المفعول 'ism Almf'wl) like (مدرّس mdrrs = teaching), verbal adjective (الصفة المشبهة Al.sfT AlmŝbhT) like (منكسر mnksr = broken), and substantive (المصدر Alm.sdr = understanding) like (فَهمٌ fahmuN). Examples of these nouns are given in the

sentences of (13a) and (14a).

(13) a. محمّد نَاجح
m.hmmd nAĝ.h
Mohammed passed
b. nAĝ.h(m.hmmd')

(14) a. زيد مدرّس المَادة
zyd mdrrs AlmAdT
Zyd teaching the course
b. $\exists y[\forall x[mAdT'(x) \leftrightarrow x = y] \wedge mdrrs'(zyd', y)]$

Based on the (العَامل Al'Aml) theory, verbal nouns can be treated as verbs because they govern the other word or words in the sentence. They can govern from one dependent, a subject, up to three dependants (معمولَات m'mwlAt), one subject and two as objects, depending on the verbs they are derived from. Therefore, a verbal noun might be a type of $< e, t >$, $< e, < e, t >>$, or $< e, < e, < e, t >>>$ depending on the verb they are derived from. From the formal semantics each of them will receive one of the following expressions:

(15) a. intransitive verbal noun $= \lambda x[P(x)]$
b. transitive verbal noun $= \lambda x[\lambda y[P(x, y)]]$
c. bitransitive verbal noun $= \lambda x[\lambda y[\lambda z[P(x, y, z)]]]$

The above expressions mean that the intransitive verbal noun will be translated to a one-place predicate, while the transitive and bitransitive will be translated to a two-place predicate, and a three-place predicate as in (15a-c) respectively.

6.2.5 Adjective Phrases

These phrases are constructed with adjectives (الصفة al.sfT). An adjective can be in the form of a "solid noun" (إسم جَامد 'ism ĝAmd), which is not derived

from a verb such as (أَسدَا 'asdA) in (16a), or in the form of an "unsolid noun" (إسم مشتق 'ism mŝtq), which is derived from a verb such as (إسم الفَاعل 'ism AlfA'l), (إسم المفعول 'ism Almf'wl), (الصفة المشبهة Al.sfT AlmŝbhT), and so on. For example, (مقَاتل mqAtl) in (16b) is derived from the verb (قتل qtl).

(16) a. رأَيت رجلًا أَسدَا
r'ayt rĝlaN 'asdA
saw I a man lion
b. الجندي مقَاتل
Alĝndy mqAtl
the soldier fighter

Adjectives like (أَسدَا 'asdA) and (مقَاتل mqAtl) are used to ascribe the property to the entity they belong to. Semantically, both types of adjectives behave like intransitive verbs in the way they enter into control and agreement patterns. Therefore, adjectives are presented in the form of a one-place predicate.

(17) a. الطَالب المجد محمّد
AlṫAlb Almĝd m.hmmd
the student the serious Mohammed
b. $\exists y[\forall x[.tAlb'(x) \wedge \exists y[\forall x[m\hat{g}d'(x) \leftrightarrow x = y] \wedge hw'(y,x)] \leftrightarrow x = y) \wedge hw'(y, m.hmmduN')]$

Another point in these phrases is that they must agree with their entity form, i.e. in terms of gender, number, and definite/indefinite. So, these phrases may be headed by a determinable article as in the example (17a). This will raise the same problem of determiner in Arabic which has been discussed in Section 6.2.2 above to give a representation as in (17b).

6.2.6 Preposition Phrases

A prepositional phrase consists of a preposition as a head followed by a complement. In the (العَامل Al'Aml) theory, prepositions work only with noun phrases and give them 'genitive' case value. Their function is to realise the connection of the prepositional complement with the other constituent in the sentence.

(18) a. $\lambda x[\lambda y[\lambda z[fy'(x,y,z)]]]$
b. $\lambda x[\lambda y[fy'(x,y)]]$

Prepositions are lexically ambiguous, as each one can have different meanings depending in its usage, and there is a debate about how to represent them to solve the problem of ambiguity. DeRoeck proposes a solution which is to represent them as three place predicates [DeRoeck et al. 1991]. The representation for the preposition (في fy), for example, is given in (18a).

(19) a. أَ يدرس بدر في شفيلد
'a ydrs bdr fy ŝfyld.
is studying Badr in Sheffield
b. $fy'(bdr', ydrs', \hat{s}fyld')$

(20) a. زيَاد في مصر
zyAd fy m.sr.
Zyad in Egypt
b. $\lambda z[fy'(zyAd', m.sr', z)]$
c. $fy'(zyAd', m.sr')$

Based on (18a), the semantic representation for a sentence like (19a) will look like (19b). But that representation cannot handle a sentence like (20a) as it will produce an incomplete formula as in (20b). In order to solve such a problem, we propose another representation for Arabic prepositions as two place predicates as in (18b). Therefore, each preposition has two representations: one as a three place predicate and the other one as a two place predicate. (18a) will

give a semantic representation like (20c) for the (20a) sentence.

6.3 Verbal Sentences

This type of sentence must have a verb phrase which has a verb as a head. Verbs in the (العَامل Al'Aml) theory are the strongest regent (عَامل 'Aml). They can govern from one dependent, as a subject, up to three dependents (معمول m'mwl), one subject and two as objects, depending on their type [1]. Therefore, each verb, depending upon its type, will be of one of the following types:

(21) a. intransitive verbal $= < e, t >$
b. transitive verbal $= < e, < e, t >>$
c. bitransitive verbal $= < e, < e, < e, t >>>$

The above types mean that an Arabic intransitive verbal noun will be translated to a one-place predicate, as in (21a), while the transitive and bitransitive will be translated to a two-place predicate and a three-place predicate, as in (21b) and (21c) respectively. In this type of sentence we treat a verb phrase as denoting a function applied to noun phrases. Thus, by using the lambda operator in (22b), which is the well-formed expression of type (21b), we can build the semantics of (22c) for the sentence of (22a).

(22) a. درّس مَازن عليًا بَاسكَال
drrs mAzn 'lyA bAskAl
taught Mazen Ali Pascal
b. $\lambda x[\lambda y[\lambda z[P(x, y, z)]]]$
c. $drrs'(mAz', 'lyA', bAskAl')$

[1] Of course there are seven verbs in Arabic which can have four dependants, one subject and three objects [Dahl 1993]. These are ignored here because they are rarely used.

6.4 'Sentences of Individuals'

In Arabic, a sentence may consist only of two simple noun phrases, as in the examples (23a,b) below. This type of phrase contains only individuals such as proper nouns, i.e. names like (أَحمد 'a.hmd) , (زيد zyd), and so on, or pronouns like interrogative pronouns. In the formal semantics, these kinds of words receive the type $< e >$. In this case, the sentence contains only two arguments, an individual variable and an individual constant, and there is no function tool which can be the functor of these arguments. In other words, the difference between this type of sentence and the other ones which have been discussed above, is that in the previous ones there is at least one function formula in the sentence construction while there is no function formula in this type of sentence.

(23) a. من أَحمد؟
mn 'a.hmd?
who Ahmed

b. من هو أَحمد؟
mn hw 'a.hmd?
who he Ahmed

In our system the semantics of sentences of this type can be achieved by a special rule as in (24a). This rule takes the noun phrase arguments, i.e. the topic and the comment of the sentence, and combines them in the form of a predicate using the word "hw" as the functor. There are two reasons behind choosing the word "hw". Firstly, the word "hw", in this kind of sentence, is just an extra article, as mentioned by linguists see [Abn-Hesham 1985] for example. Therefore, it can be added or deleted, as in the examples above, without any problem, unlike copular verbs which should have a grammatical effectiveness in the sentence. Secondly, the relationship between the phrases of this sentence is equalisation and the article "hw" can represent this function. The semantic representation will look like (24b).

(24) a. $\lambda x \lambda y[hw'(x, y)]$
b. hw'(x,'a.hmed')

There is a close relation between syntax and semantics in GPSG: the structure provided by syntactic rules is the one interpreted by semantic rules. Thus, this type of sentences need to be analysed by a special syntactic rule combined with the semantic rule in (24a).

6.5 Conclusion

Formal semantics is very important in portable question answering systems because it expresses the meaning of the user's question in terms of high level world concepts, which are independent of the database structure. Thus it is not difficult to port systems based on this approach to other databases or knowledge domains.

Treatments for Arabic statements to build well-formed expressions have been proposed based on the formal semantics. The treatments are within the frame of the Arab grammarians theories (العَامل Al'Aml) and (إسنَاد 'isnAd).

A semantic classification for Arabic sentences has been proposed which is based on the type of function they have. From this classification we find that the semantics of Arabic sentences cannot be always built by applying the semantics of one part to the other one. Some sentences contain open formulae which need to be bounded before the application process. In addition, the application process needs to be done by a special rule at the sentence level as there is no form of formula in the sentence parts.

Finally, well-formed propositions are produced for Arabic fragments without the need for the idea of a copular deletion hypothesis as suggested by Bakir [1980] and Fassi Fehri [1987].

§§§§

Chapter 7

Implementation

The treatments proposed in the previous chapters should be proved and tested by building a prototype system. The decision has been made to implement the prototype using one of the existing systems called Squirrel[1].

This chapter will give a brief description of Squirrel and the reasons behind using it. In addition, it presents the character set which is used for Arabic in the prototype. This chapter also gives a description of an example domain which is used for the prototype.

7.1 The Squirrel System

Squirrel is a portable natural language front end prototype[2] for the interrogation of logical and relational database systems. Squirrel can handle a single-query which contains no anaphora. The system does not have an algorithm to

[1]developed at The University of Essex under SERC grant GR/E/69485.

[2]A commercial version is being developed under DTI and EPSRC funding, grant GR/54956.

resolve anaphora[3]. Further references to the system are [DeRoeck et al. 1991, Fox 1995, Barros 1995, Lowden et al. 1991-1992a-].

The system consists of two main components: the Front End and the Back End. Both the Front End and the Back End were designed to guarantee modularity and portability of the system as a whole.

7.1.1 The Front End

The Front End accepts the input sentence (in English), producing syntactic and semantic representations. Both the grammar and lexicon are written based on Generalised Phrase Structure Grammar (GPSG) [Gazdar et al. 1985], [Bennett 1995]. The lexicon is incomplete, treating unknown words as proper nouns to be fully interpreted when reaching the database. The system builds its meaning representation by using annotations on the grammar rules and lexical items. The parser is bi-directional which means it works bottom-up or top-down. The initial semantic representation is as a Property-Theoretic (PT) term [Turner 1992] which it maps into First Order Logic(FOL). All representations are independent of the domain of application.

As the system does not have a morphological analyser, a new lexical entry is required for each inflected word (e.g. "works", "working", "worked" and so on which are all derived from the word "work"). There are no entries for data belonging to open classes such as proper nouns, these are postponed to be fully interpreted when reaching the database.

Squirrel's parser is a bi-directional chart parser which means it works bottom-up or top-down. Depending on the grammar rule last used, the active arcs may be extended in either direction. For each successfully parsed input string, one or more traditional syntactic tree is returned. The nodes of the syntactic tree

[3]This is the status of the version we have, although a PhD [Barros 1995] work has been done to build a discourse model for anaphora resolution for the system.

are labelled with rule numbers. All valid syntactic interpretations are passed on for further processing. This is part of Squirrel's ambiguity resolution.

The parser treats unknown words which occur in the syntactic position of an NP in the input string as proper names and sets up a temporary lexicon entry for them. For example, if the input string contains `tata` as a word and it is an NP in the syntactic position, a new lexicon entry for it is set up. The syntactic category for the unknown word is `propn` with uninstantiated features and having `lambda(p, p: tata)` as a semantic value.

That treatment is required to make the parser able to use an incomplete lexicon (i.e. it does not contain names). Apart from that, any proper nouns must be encoded in the lexicon in order to ensure successful parsing of sentences.

7.1.2 The Back End

The Back End uses an Extended Data Model (EDM) and Domain Specific Configuration (DSC) to translate FOL formulae, first into Untyped Relational Calculus (URC) and Domain Relational Calculus (DRC) expressions, which are translated via Tuple Relational Calculus (TRC) into SQL. All representations at this level are domain dependent.

Although, ruling out inappropriate semantic analyses based on the domain information is a task of the Back End, it introduces a new level of ambiguity. All successful representations which are generated at any translation stage, are passed on for further processing at the next translation stage. Only appropriate interpretations with respect to the domain application are passed on after the database consultation which is the final filtering process.

If there is more than one valid interpretation that retrieves valid database answers, all the query-answer pairings are passed on to the user. More details about the system's Back End can be found in [Lowden et al. 1991-1992b].

7.2 Why Use Squirrel?

One of Squirrel's main goals is for the front end system to be as independent as possible of the database domain being queried. It aimed to show that natural language front ends could maintain a high degree of portability and can be built without recourse to world models or inference engines under principled design criteria. The system was developed to meet three criteria: a high degree of portability, reliability, and transparency.

A high degree of portability was achieved by three characteristics: modularity, domain-independence and a non-complete lexicon. For modularity, knowledge (i.e. lexicon, grammar, data model and so on) is distributed among several files. Also, several intermediate representations of the query are used, until the final output is released. Squirrel, blocks the use of knowledge bases with information about the domain (apart from the data model and lexicon) in order to be domain-independent. It treats unknown words as placeholders to be bound once the query reaches the database.

Within the same context, the same query must always generate the same final representation (the SQL form) for the query and the same database answer must be obtained. Further, the system must not try to resolve ambiguity at any cost. The reason for this is to avoid an incorrect interpretation from being picked out which was not intended by the user. Thus, for ambiguous queries, one representation is produced for each possible interpretation, followed by its corresponding database answer. These features determined the system's reliability.

For transparency, all of the system's output must reflect the internal processes. The user should be able to understand clearly how the system works out the meaning of the query, so the user is able to interpret the system's response. This is related to the ambiguity resolution strategy which the system employes.

Finally, Squirrel is available and licensed free for research purposes.

7.3 Arabic Alphabet

Each character is written from right to left. There are no capital letters in Arabic. Apart from some individual letters, there is no separation letters, while writing words. The shape of the letter depends on its place in the word. It might be: stand alone, joined to the preceding letter only, joined to the preceding and the following letters, or joined to the following letters only.

To represent Arabic characters in the UNIX environment we need arabisation software. But because this work is just for research purposes, Latin characters can be used to represent the Arabic characters. The alphabet of Arabic consists of 28 characters while Latin consists of 26 characters, so, two numbers are used to represent two Arabic characters.

For every Arabic character its equivalent from Latin is used and for those which do not have such an equivalent a random Latin character is used. In this way it is very easy to run the system using Arabic alphabet characters by using arabisation software and changing the character set in the system to the equivalent in Arabic. The Arabic alphabet with the Latin equivalent used in the system is listed in Appendix A.

In Arabic there is a small notation above a character called (شدّة ŝddT). This is used when there is a double character, one of them being replaced by this notation. The system will use the double character as a sign for this notation. Thus, users should not ignore this in order to get a correct interpretation for the word.

It should be noted that our system can process queries without vowels. It can also accept vowels if they appear at the end of proper nouns or names. Apart from that the system cannot process queries with vowels, at least for the time being.

7.4 The Example Domain

Although the goal of this research is building a domain independent natural language interface, we need to prove the success of such an idea through example domains. Therefore, we used two example databases. The first one is used for system development and for describing how the system works. The second one is used to show how the system can be ported to a new domain to prove the portability of the system. The first database example will be discussed here while the second one will be discussed later in Chapter 11.

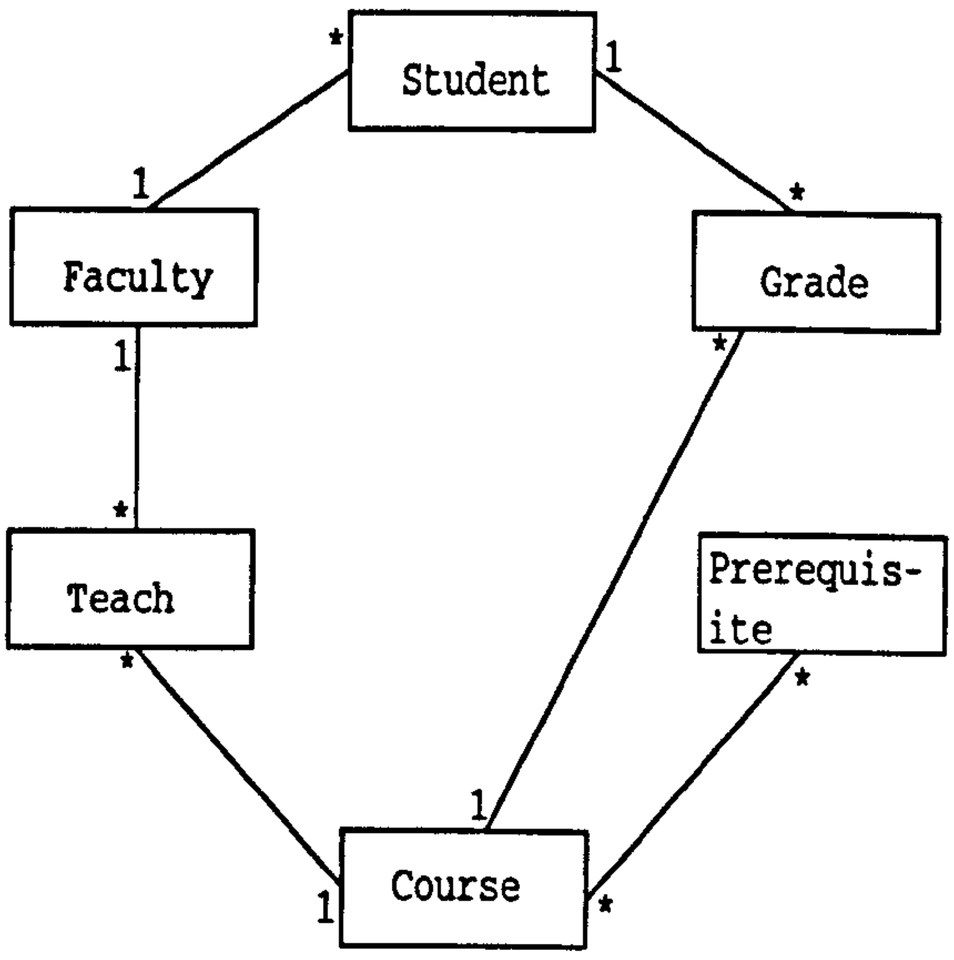

Figure 7.1: Data structure of the example domain

The system is built to access an example of a student database. The data model of the database is relational. The database contains six entities: student, faculty, course, prerequisite, section, and grade. All the relations between them are one to many except the relation between course and prerequisite as shown in Figure 7.1.

All of these entities are kept in different tables. The following represents the database tables structure.

Student

st-name	st-num	st-gender	st-address	st-degree	st-major

st-department	fac-num	st-year

Faculty

fac-name	fac-num	fac-gender	fac-address	fac-degree	fac-title

st-department	fac-major

Course

cors-name	cors-num	cors-category	cors-department	cors-credit

Prerequisite

cors-num	precors-num

Teaching

cors-num	tch-semster	tch-year	fac-num

Grade

st-num	cors-num	semster	grd-year	st-grade

7.5 Graphical Interface

Squirrel does not have a graphical interface. It reads a query from the Prolog prompt and returns results back. All error messages are also returned to the user through the Prolog prompt which makes the output of the system look nasty as it returns every possible translation in each stage see for example

Section 10.3.2.

We have constructed a Graphical Interface (GI), as shown in Figure 7.2, on top of Squirrel. One reason behind the graphical interface is that it will filter system responses and display only complete translations of the query. Also, if there is no complete translation for the query, the GI will tell the user what the problem is only once. Finally, the GI will improve comunication between users and the system during the testing stage of this work.

The user graphical interface module is built using the Tcl-TK toolkit. It allows Squirrel to work off line as all inputs and outputs will be stored in separate files. So, the execution of the GI module follows the following structure:

1. Disply input-output screen.
2. Accept the query from user.
3. Load Sicstus Prolog.
4. Process the query.
5. If there is one or more answers, display them to the user one by one upon the user's request.
6. If there is no answer, display error message.

Because this work uses Latin characters to represent the Arabic characters, a virtual Arabic keyboard is built in the user interface. So, by pressing the button of the Arabic character, which appears in Figure 7.3, the system will type the Latin character that represents it in the input query field. The user does not need to know which Latin character represents which Arabic character in the system.

It should be noted that this graphical interface causes a delay in the system responses. This should not be a problem as the system is just a prototype.

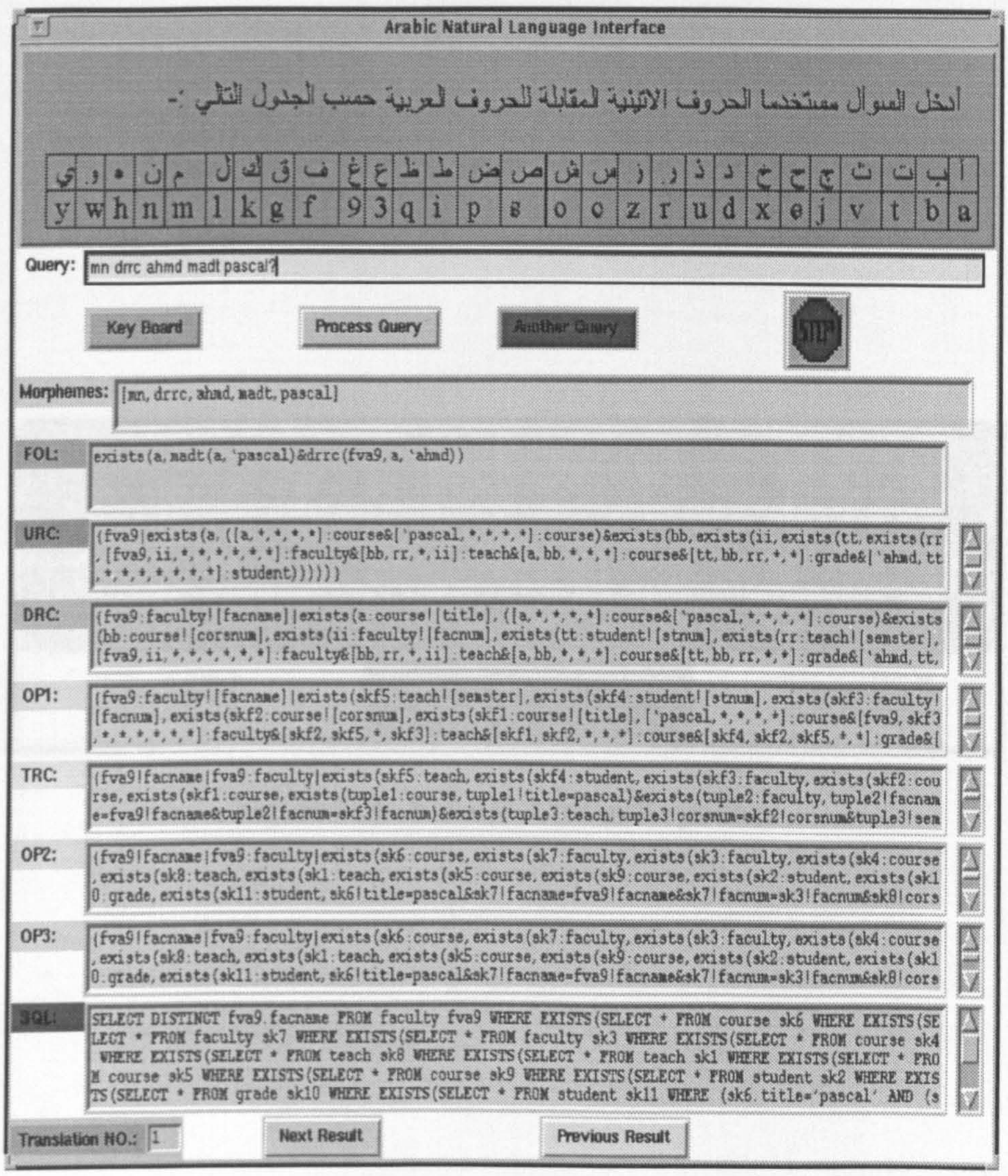

Figure 7.2: User Interface

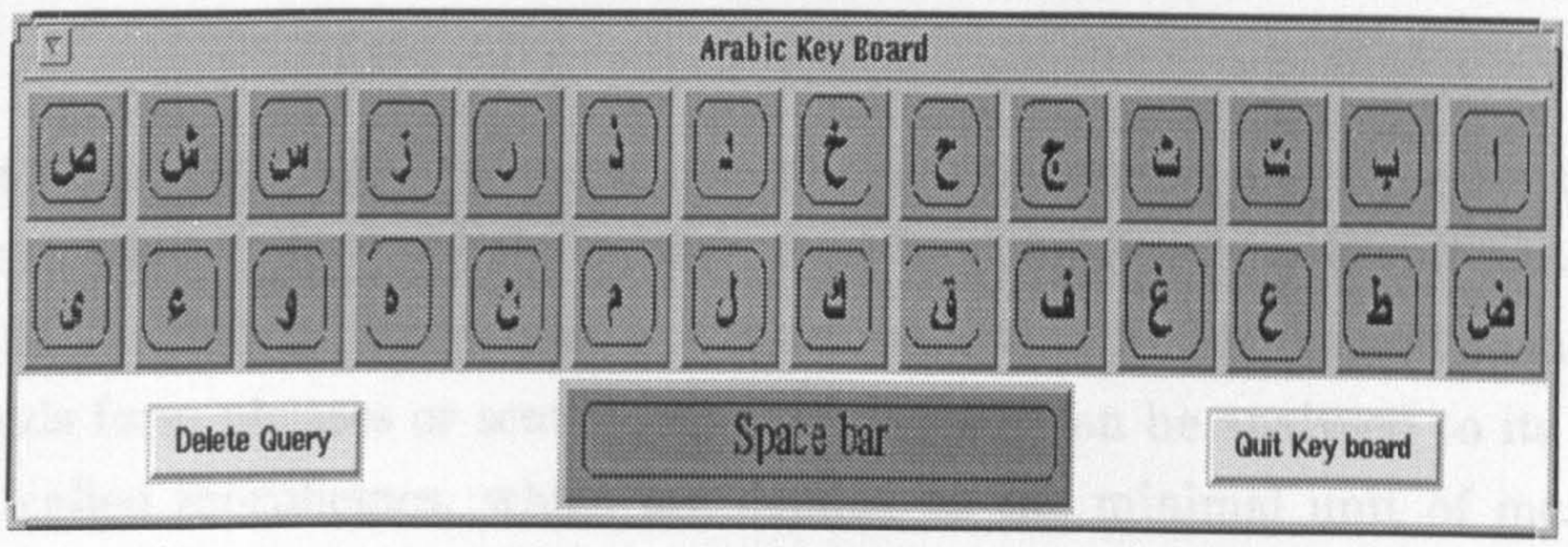

Figure 7.3: Virtual Arabic Keyboard

Chapter 8

Morphological Analyser

Any natural language processing system needs to process and interpret words in some way. A word, on the surface, may be defined as a string of letters or non-blank symbols. Successive words are separated by blanks and sequences of words form phrases or sentences. A word itself can be analysed to its basic units called morphemes, which are defined as the minimal unit of meaning [Winograd 1983].

A portable natural language front end system requires a lexicon with a large number of words with their syntactic and semantic information. Since this is a problem for Indo-European languages, such as English, it is, certainly, a non-trivial task for a semitic language, such as Arabic, where words are mostly made up of more than one morpheme.

In this chapter we present ALMHLIL which is the morphological analyser that has been implemented and is attached to the Arabic version of the Squirrel system. ALMHLIL is built based on the review and evaluation of other systems which are discussed in the Section 8.2 on previous algorithms. Each module of the analyser is described in a separate subsection. An example of ALMHLIL's output is given before the conclusion section.

8.1 Introduction

Morphology, in linguistics, is simply a term for the studies concerning the structure of words or the forms words take in their different usages and constructions along with their formation [El-Sadany and Hashish 1989]. The morphological structure of the Arabic language is very complicated. A token may contain verb, subject, and object as one word [El-Dessouki et al. 1988]. For example, a verb like (درّسوهم daraswhm = they taught them) is written as one word while it is made up of the verb (درّس darras = taught), the subject masculine plural pronoun (و w = they), and the object masculine plural pronoun (هم hm = them). Similarly, a noun such as (بيتنا bytna = our house) is made up of the noun (بيت byt = house) and the spoken plural pronoun (نا na = our).

An Arabic word can have up to seventy different derivations depending mostly on grammatical information and partly on semantic information [Mehdi 1987]. Table 8.1 shows examples of some derived words for the root of the word (درس drs = studied).

Table 8.1: An example of derivation words

الكلمة word	type	meaning	root
درس drs	(verb)	studied he	drs
درّس drrs	(verb)	taught he	drs
تدرّس tdrrs	(verb)	teaching she	drs
يدرّس ydrrs	(verb)	teaching he	drs
درس drs	(noun)	lesson	drs
مُدرِّس mdrrs	(noun)	teacher	drs
مُدرِّسون mdrrswn	(noun)	teachers	drs
مُدرِّسات mdrrsat	(noun)	teachers (fem)	drs
مَدرَسَةٌ mdrst	(noun)	school	drs
دَارس dArs	(noun)	the one who study	drs

Words can be classified, primarily, into two groups: words which exhibit morphological variations having either syntactic or semantic value, and those which have a stable morphological shape. Verbs and nouns belong to the first class while prepositions, negations, and coordination are examples of the second class [Beeston 1970].

Roots in Arabic are a sequence of consonant phonemes in determined order. They cannot be derived from any other word or combination of words, i.e. nouns, verbs, and articles. Three classes of words have been stated as the roots of Arabic: tri-radical, quadri-radical, and quinque-radical. Verbs can be derived from tri-radical roots, like (أستكتب 'astktb = asked to write) which derived from (كتب ktb = to write), and quadri-radical roots, like (تدحرج td.hrĝ = rolling) which is from (دحرج d.hrĝ = to roll). More than sixty-three percent of the Arabic roots use the tri-radical roots [Alneami 1997].

Nouns, in Arabic, can be built from their basic classes of roots: tri-radical as in (مكتبة mktbT = library) from (كتب ktb = write), quadri-radical as in (مهندس mhnds = engineer) from (هندس hnds = engineer), and quinque-radical as in (سفرجلتّان sfrĝltAn = two quince) from (سفرجل sfrĝl = a quince).

It should be noted that each type of Arabic root has a number of patterns. Patterns can be defined as a string of Arabic characters containing the basic root and some affixes, i.e. prefix, infix, suffix, or a combination of them, attached to it. The tri-radical root, for example, has fifteen patterns. These patterns are recognized as the stem of the inflected word.

8.2 Previous Algorithms

A number of algorithms have been proposed for morphological analysis of Arabic words. One approach, proposed by Hilal [1986], differentiates between function words, which will be treated first, and regular words, such as nouns

and verbs. El-Sadany and Hashish [1989] developed a two-way algorithm (analysis/generation) capable of dealing with nonvowelized, semivowelized, and vowelized Arabic words.

An algorithm for generating the root and the pattern of a given word is presented by Al-Fedaghi and Al-Anzi [1989]. ARABATN is a morpho-syntactic parser for Arabic proposed by Roochnik [1989] using the Augmented transition Network (ATN) framework. Bessley et al. [1989] describe a comprehensive two-level Finite-state model for Arabic developed at ALPNET in the USA.

A three-level model is proposed by Narayanan and Hashem [1992]. The model represents distribution patterns for a very limited range of components which constitute a third level on top of the standard two-levels. An Auto-Analyser and Generator developed by Alneami and McGregor [1995] to generate the Arabic words in their machine translation system is based on the LFG formalism.

The next sections present some of these algorithms in more detail followed by a general evaluation.

8.2.1 Hilal's Algorithm

Hilal [1986] proposed a morphological analyser based on dividing Arabic words into two classes: regular words which depend on the derivation rule mechanism (nouns and verbs), and function words which do not depend on the derivation rules (these are mainly articles)[1].

The system analyses the input word in two stages: first the function words, and then the other types. At the beginning the system checks whether the input is in the article dictionary or not. If the input word is not in the dictionary, then either the word is not an article, which means it is a regular word, or the word

[1] Articles in Arabic form a wider category than in English.

is formed of primary articles and then needs to be analysed into a sequence of articles.

All possible ways of analysis will be tried until the correct ones are reached. A word like (فهم fhm), for example, may be interpreted as (فَهِمَ fahima), (فَهَّمَّ fahhamma), (فَهُمُ fahmu), (فَهِّم fahhim), (فَّهَمَّ ffahamma), (فـ+هَمَم hamam+f), (فَهُم fahum), (فـ+هم hm+f), and (فـ+هم +هم hm+hm+f). Of course, the analyser will reject the last analysis.

If the input word is not an article, i.e. it is a noun or a verb, or a 'normal' word to use Hilal's terminology, it will be analysed at the second stage. The analysis is done by separating prefix and suffix letters. This separation is repeated until the word is found in the dictionary.

The function word analyser contains a lexicon of all basic function words, such as (و w), (في fy), (ب b), (ف f), and so on, and a set of rewrite rules for the decomposition of words in terms of the function words and the connecting elements. On the other hand, the regular word analyser has a set of morphological rules in addition to a number of lexicons, a lexicon for all prefixes, a lexicon for all suffixes, a lexicon for all strong and weak roots, and a lexicon for schemes.

8.2.2 EXTRACT

Al-Fedaghi and Al-Anzi [1989] proposed an algorithm called EXTRACT for generating the root of a given Arabic word and its pattern. The main concept of the algorithm is to check the trigraph forms of a given word to see whether they form a valid root in Arabic or not. This is done by locating the position of the root's letters in the pattern and then examining the letters in the same position in the given word.

In the word (خطير h.tyr), for example, all patterns with length four letters,

such as (فعيل f'yl), (فَاعل fA'l), etc., are examined. The positions of the root letters in each of these patterns are coded by EXTRACT. In the pattern (فعيل f'yl), for instance, they are in place one, two, and four. After extracting the root letters the algorithm can identify the root-pattern corresponding to the given word which is (خطر h.tr, فعيل f'yl) in our example.

In the case where the input word contains its full triliteral root letters, all possible patterns that have the same length as the input word will be examined. On the other hand, if the input word has lost one of its triliteral root letters, the algorithm will test all patterns of length equal to one less than the length of the given word. Similarly, for the words that have lost two letters of their triliteral root letters, all patterns of length equal to the length of the given word minus two are examined.

EXTRACT has four cases it can deal with: the full triliteral root, the missing of the third letter of the root due to (أَلتشديد 'altŝdyd), the missing of one letter of the root of the input word, and the missing of two of the root letters of the input word.

8.2.3 Two-level Finite-state

At ALPNET a morphological analyser and translation tool from Arabic to English was built [Bessley et al. 1989, Bessley 1990]. It is based on Two-level Finite-state theory which was presented first by Koskenniemi [1983]. The algorithm works at two levels: regularised lexical level and surface orthographical level. The lexical level is a collection of morphemes, each consisting of a string of characters, for any valid word. The surface level, on the other hand, is a string of characters representing an Arabic word as it actually appears in standard Arabic orthography.

The representations at the two levels are tightly related by rules. The root lexicon stores roots (containing three constants) in the form 'C_C_C' and the

pattern lexicon stores inflectional patterns in the form '_a_a_' where the underscores '_' are called Detours. The analysis process starts with the pattern lexicon and the analyser routines recursively switch between the two levels of lexicon whenever a Detour character is found.

The system takes on-line words written in Modern Standard Arabic and identifies all of their valid morphological analyses. The input words may be fully vowelised, partially vocalised, or unvocalised. The system displays all possible readings of each word attached with the full breakdown of their morphemes, including the roots and patterns where appropriate. For example the word (بنت bnt) has:

Lexical: $bint + u$

Surface: bnt

The above example represents the two levels of the word "bintu" based on the simple indivisible stem 'bint' with the definite nominative $suffix + u$ where the short vowels are not realized at this level.

In 1995, work began to rebuild the ALPNET's system using the finite-state format developed by Xerox Research Center Europe to produce a large morphological analyser for Modern Standard Arabic [Bessley 1996]. The rules were rewritten to support generation more reliably. The system has a Java user interface which was added to allow users to interact with the system via the Internet in standard Arabic orthography [Bessley 1998].

8.2.4 Evaluation of the Algorithms

There are a number of approaches which have been devised to perform morphological analysis of Arabic words. In general, most of the Arabic morphological systems have a disadvantage in common which is the use of dictionaries of

roots and other types of words. Furthermore, there is no indication of the implementation of some of these approaches [Khayat 1996].

The use of dictionaries for the primary articles, prefixes, suffixes, and roots makes Hilal's system slow [Al-Uthman 1989]. Also, checking for all affixes in the language which are more than four hundred in number make it impractical. Consider a word like (منتحر mnt.hr) which will be analysed as (من mn) + (تحر t.hr). Since (تحر t.hr) is not an existing root, the system will continue to analyse the word by taking letter (ت t) as a possible prefix which makes the root as (حر .hr). But this is an unacceptable analysis of such a word. Therefore, the algorithm sometimes may give a correct result but not necessarily based on an acceptable analysis.

The greatest advantage of the EXTRACT algorithm is that it does not have a dictionary for all prefixes and suffixes. Also, it does not have to identify these affixes in order to find the word's root. On the other hand, the EXTRACT algorithm is slow and requires large memory space [Al-Shalabi 1996]. Also, a word like وفَا, for example, will not be recognised unless it is written like وفَاا in order to produce its root-pattern representation. In addition, the algorithm does not have the ability to analyse article words.

In ALPNET almost 5000 roots and hundreds of patterns have been stored in separate sub-lexicons. But the Detouring operation that related them in real time was inherently inefficient. Because an Arabic root can combine legally with only a small subset of possible patterns, the system will build and then throw away many superficially plausible stems that were not sanctioned by the lexicon codings [Bessley 1996]. In addition, the system generates a large number of all possible analyses of a given word which makes the parsing of an Arabic sentence a complicated process. Also, it will take long time as the parser will try every combination of possible inflection read for words of the sentence.

All of these systems focus on finding the root form of the given word which

is not suitable in many ways especially in the understanding process and the translation process. As Hilal suggested the root form may serve as a secondary tool [Hilal 1986]. However, there is no need for such analysis and duplication of entries as the surface word form may serve as the basic level of the given word.

Finally, with the exception of the Xerox Arabic morphological analyser, none of these systems are avaliable for use. Even the Xerox system is not licenced for research usage yet.

8.3 ALMHLIL

As has been seen, a strong morphological system allows any noun or verb to be analysed down to one of the three root types: tri-radical, quadri-radical, and quinque-radical. But having the ability to find the root for each word in a given sentence is not the main point in the process of understanding the sentence. In addition, finding the root of the word may result in loosing the original meaning of the given word. Therefore, we decided to build an analyser based on finding the pattern of the words not on finding the root of the word.

After examining Arabic word types we found that not all prefixes, infixes and suffixes belong to the original word. The phonology system of Arabic permits more than one word to be joined together which makes them look like one word. Consider a word like (مدرّسهم modarrisahom = their teacher), for instance, it is written as one word while it is a combination of two different words, the noun (مدرّس modarris = teacher) and the personal pronoun (هم hom = their).

For that reason, ALMHLIL is designed to distinguish between two types of morphemes: internal morphemes and external morphemes. An internal morpheme is called a dependent affix which exists in the word's pattern. It adds more information to the word such as gender and number. In other words, it has a meaning while it is a part of the word but this meaning will disappear if it

is separated from the word. Any affix character which is a part of the word pattern but does not exist in the root of that word can be seen as an example of this type of morpheme.

On the other hand, external morphemes are independent affixes. They have their own meaning even when they are separate from the word. They do not add any information to the given word because they are not a part of the pattern of the given word. Therefore, they can be seen as additional words attached to the given word. Personal pronouns are examples of this kind of morpheme.

If ALMHLIL recognises an external affix, it will be kept apart from the original word. On the other hand, if an internal affix is recognised and the remaining word exists in the lexicon, the system will create a feature structure for the whole word and store it as a temporary entry in the dynamic lexicon. Entries in the dynamic lexicon are then used in subsequent analysis of the current query, see Chapter 9.

All of the words in Table 8.1 are derived from the basic root (درس drs = studied). Some might argue that one should keep the basic root of those words and let the morphological analyser handle the others. The problem with this approach is that the semantic features of each word get lost. The verb (درس drs = studied), for example, takes a subject and/or an object while a verb like (درّس drrs = taught) takes a subject and one or two objects and so on for the other words. Further, there is a trade off here between the number of lexical entries and the number of morphological rules needed.

8.3.1 The Architecture of ALMHLIL

ALMHLIL recognises input words as one of three types: nouns, verbs, and articles. It contains four modules: a verb module, a noun module, a proper noun module, and an article module. ALMHLIL is domain independent and

keeps a high degree of portability by not using any knowledge base. The system has the ability to treat inflected unknown words, i.e. proper nouns. Figure 8.1 shows how the system works.

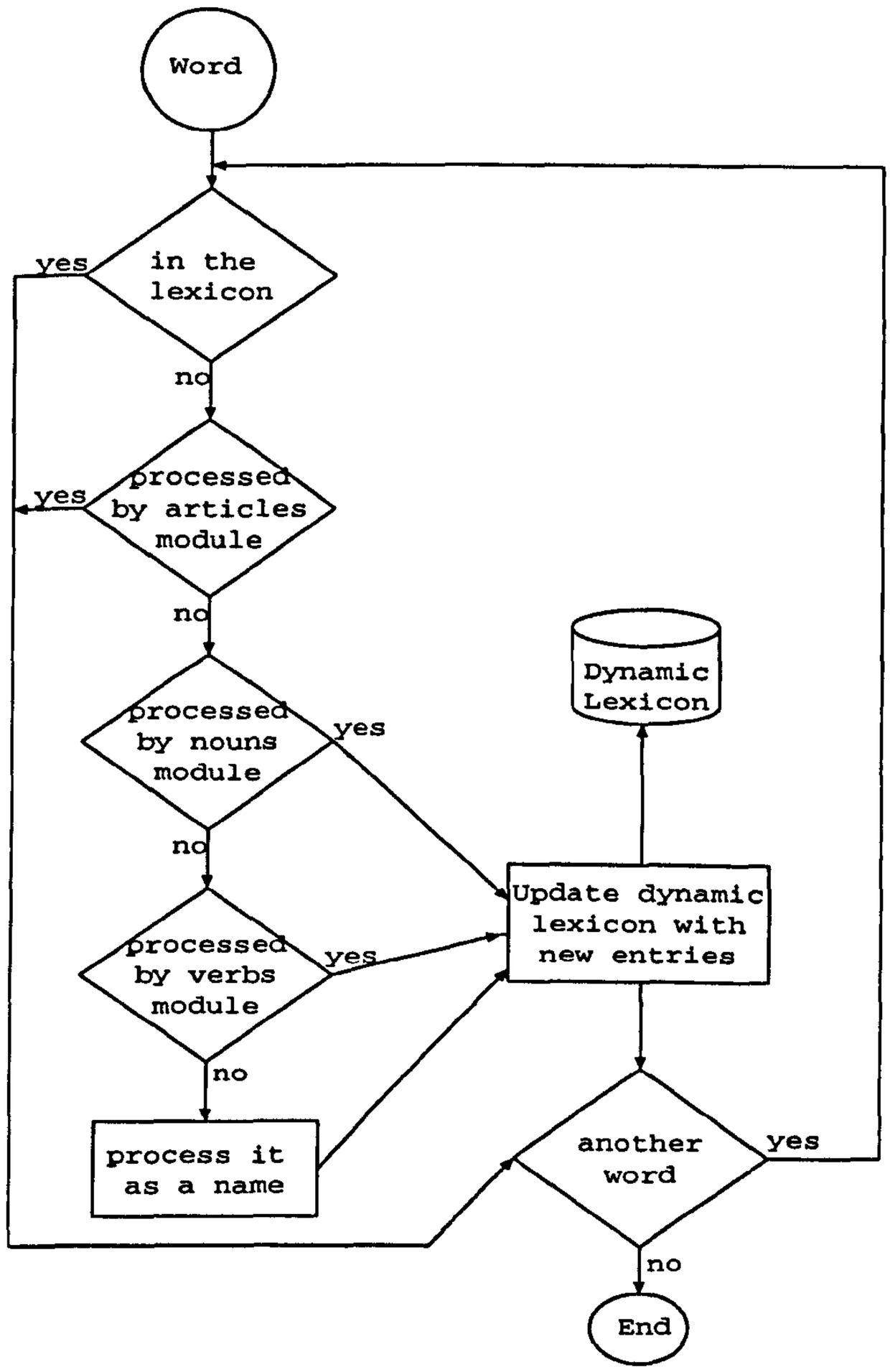

Figure 8.1: ALMHLIL Architecture

ALMHLIL is built based on the following algorithm.

- A word may not be inflected by any means, prefix(es), infix(es) or suffix(es). In this case the word itself is a stem.

- A word may be inflected by one or more independent morpheme prefix(es) and suffix(es). In this case the word will be separated into stem, prefix(es) morpheme, and suffix(es) morpheme.

- A word may be inflected by one or more characters prefix(es), infix(es), and suffix(es); but they are not independent morphemes. In other words, they do not have a meaning as individuals, but they add some information to the stem. In this case, the stem of that word and the extra information conveyed by those characters (number, gender, time, etc.) will be asserted temporarily to the lexicon.

- A word may be inflected by one or more morpheme prefix(es) and suffix(es) and the root of the word is inflected by one or more characters prefix(es), infix(es), and suffix(es) which are not independent morphemes. In this case the word will be separated to its morpheme as above and the inflected root of that word with the extra information given by those inflection characters (number, gender, time, etc.) will be asserted temporarily to the lexicon.

- A word for which there is no stem, because it is a combination of two or more articles as classes of nouns, i.e. pronouns, and so on, prefix(es) and suffix(es). Here the word is divided into its prefix(es) and suffix(es) morpheme but there is no stem morpheme here.

- If the word is not in the lexicon and cannot be recognised by the verb module, the noun module, and the article module, it will be considered as a proper noun and the system will create feature attributes for it then assert it to the lexicon temporarily.

- Any given name will have a temporary entry in the lexicon.

- If a name is inflected by a prefix or a suffix, it will be analysed and a new temprorary entry is given for each possible result of the analysis.

8.3.2 Verb Module

The verb module can handle verbs with tri-radical and quadri-radical roots. As this work is for prototype purposes and the quinque-radical roots are rarely used in modern Arabic, they are not recognised by the system, at least for the time being. It can analyse verbs which have two characters in length to up to twelve characters in length.

The system determines the external affixes, i.e. prefixes and suffixes, and the internal affixes which are those characters that affect the meaning of the word which are augmented in the Arabic word (سَاءلتمونيهَا sA'ltmwnyhA). This module first checks if the verb exists in the lexicon or not. If it is not in the lexicon, then it will be split to its letters for further analysis.

The system uses two directions for analysing Arabic verbs: suffix to prefix and prefix to suffix. The first direction is based on the assumption that there are one or more external suffixes existing in the given verb which need to be recognised. Then the remainder will be checked for existence in the system lexicon. If the remainder does not exist as a word entry in the lexicon, then the system will continue to analyse the word looking for prefixes and infixes.

The second direction is based on the assumption that there are one or more external prefixes to be recognised first in the word and the rest will be checked with the lexicon. If the remainder does not form a word in the lexicon, a further analysis will be used for searching for possible infixes and suffixes to be recognised in the word.

The reason for these two direction of analyses are to examine all possible analyses for the given word. In addition, it will ensure the system reaches the correct analysis for the word. This is true especially when we are looking for the word's pattern level not the word's root level.

A verb can be inflected by one or two external prefixes, i.e. a conjunction

article as in (فضرب fdrb = then hit), a question article like (أنجح 'anĝ.h = did pass), an imperative article as in (ليضرب lydrb = do hit), or by a combination of these as in (أوقال awgal = and did said) and (وليضرب wlydrb = and do hit). It could also be inflected by one or two personal pronouns as external suffixes as in (درسوا drswa = studied they) and (درّسوهم drrswhm = taught they them). These affixes will be isolated from the word and recognised as other words.

The verb may also be inflected by internal prefixes, infixes, or suffixes but in this case these affixes result in adding new information to the given verb. The module will take out these affixes to get the verb in the past singular active pattern form and then check whether this verb exists in the lexicon or not. If the verb pattern exists in the lexicon, the inflected form will be asserted, after creating its feature values, as a new entry in the dynamic lexicon.

Consider, for example a word like يدرّسهم "ydrrshm = teaching them" there are two affixes to be recognised here. The first one is the external suffix which is the plural personal pronoun هم "hm = them". The second one to be recognised is the internal prefix ي "y" to indicate the time of the verb which is present and the gender of the agent. Since the verb درّس "drrs = teach" exists in the lexicon the system will add a new entry in the dynamic lexicon for this word combined with its internal prefix. It will also create the features of the new entry.

Finally, all Arabic verbs are in masculine form but they may become feminine if they are inflected by (ت t) at the end of the past form and at the beginning of the present form. The original verb will be in masculine form and a temporary entry will be added if the system recognises the feminine (ت t).

8.3.3 Noun Module

Nouns, in Arabic, can be classified into three groups: derivational nouns, non-derivational nouns, and proper nouns. The derivational nouns are those derived from verb roots and these have fixed patterns, sometimes called measures. There are certain patterns for each derivational noun. The number of patterns for the derivational nouns are around twelve hundred in total [El-Sadany and Hashish 1989]. The non-derivational nouns are called solid nouns as they do not derive from the roots of verbs. They were abstracted from Arabic speakers many centuries ago.

The system's noun module can treat nouns derived from tri-radical, quadri-radical, and quinque-radical roots. It can analyse nouns which have up to fourteen characters in length. Like the verb module, this module uses two directions for analysing Arabic nouns: suffix to prefix and prefix to suffix.

This module recognises the external affixes of the given noun up to its pattern. Any prefix articles like determiner (ال Al), conjunction (و w, ف f), preposition (ب b), and so on are recognised as different words. Suffixes such as pronouns (هما hmA) are also recognised as other words.

In addition, it determines the number attribute of the noun, i.e. plural, dual, or singular, and the gender attribute, feminine or masculine. It also recognises the case ending sign, i.e. nominative, accusative, and genitive, if it exists.

The analyser can treat nouns which are in the form of regular plural masculine, dual masculine, singular feminine, dual feminine, and regular plural feminine. These forms have a limited number of patterns for their derivation. Therefore, when the system recognises them, it creates a new lexical entry in the dynamic lexicon.

Irregular plural nouns, on the other hand, have an unlimited number of patterns. For this reason the system will have a lexical entry for the irregular

plural form. It will be analysed as it is not derived from its singular form by a rule. Gender, in Arabic, is grammatical not necessarily natural which means nouns may be either feminine or masculine, but not necessarily male or female. Therefore, when the system recognises a word inflected with the gender morpheme, it will assign a masculine or feminine value in the syntactic entry gender attribute. A male or female attribute value will also be assigned to the word's semantic entry but this value is for mammal nouns only.

Sometimes the inflection causes one character of a morpheme to be dropped if it is preceded by another morpheme. A word like (للطَالبين llialbyn), for example, has two prefixes. The first (ل l) is a preposition while the second (ل l) is a determiner. The determiner in Arabic is (ال Al) but the first character (ا A) is dropped in the word because it is preceeded by the (ل l). The system will recognise them.

8.3.4 Name Module

Proper nouns in Arabic, like any other language, can be in any form regardless of the roots or patterns. They are a sequence of strings forming an infinite list of words. Sometimes, they are related syntactically and semantically to a certain root or pattern but that is not the case most of the time.

Arabic names may be inflected by some articles, short vowels, and nunation (تنوين tnween) as prefix and suffixes. Recognising these affixes is very important in querying databases. If tokens like (بدرًا badran), (بدرين badryn), or (ألبدر albadr) are considered as proper nouns, a wrong answer will be produced for the query as the database will have the proper nouns without any affixes. Therefore, to avoid such problems the system is based on analysing any proper nouns to recognise prefix articles, short vowels, and nunations. Beyond that any names will be taken as they are and features will be created with no value assigned to gender and number.

However, proper nouns may be inflected by the case ending marks either the short vowels, (فتحة fatha), (كسرة kasrah), and (ضمة dummah), or the nunation (تنوين tnween), (فتحتين fathatyn), (كسرتين kasratyn), and (ضمتين dummatyn), to indicate the case of the proper noun in a given sentence.

The purpose of these case endings, the short vowels and (تنوين tnween) nunation, are to represent the case of the name, i.e. accusative, genitive, and nominative, but they are not part of the proper nouns. The system will recognise these marks then remove them from the word after creating the features and their values. Sometimes these case ending signs result in adding an extra character such as alf (ا a) in order to put the (فتحتين fathatyn) as in (أحمدًا Ahmedan). This extra character will be also recognised and taken away from the name, so, the name (أحمدًا Ahmedan) will be returned to its original (أحمد Ahmed).

In addition, proper nouns can be inflected by suffixes such as the (ين yn), (ون wn), (اَن an) and so on, to indicate the number. In Arabic, if you have two or more persons called by the same name they can be called by one name in a dual or plural form. For example, if there are two persons called (محمد m.hmd) you can call them in a dual form like (محمدين m.hmdyn) or (محمدَان m.hmdAn) depending on the case of the proper noun.

Although, these are the case signs of proper nouns when they are in plural and dual forms, they cannot be analysed like this all the time. A person may call himself, for example, (محمدين m.hmdyn), or (عليَّان 'lyyAn). If the system analyses these names as they are in a dual or plural form it will lead to an incorrect result. For this reason the morphological analyser will make the entries for this type of name, one taking the whole as one name and the other one without the suffix.

Usually, feminine proper nouns can be distinguished from masculine ones based on certain marks such as the (ة T) of feminine ending. But that is not the case all the time as some have these marks and are used for masculine, for example

(مدحت md.ht), (أسَامة 'sAmT) and so on. In addition, a large number of proper names do not have these marks and such names can be used for both genders such as (ندَى nd_A), and (فجر fgr). Thus, the system will not assign a value for the gender feature for these words.

If the name ends with the letters (ات At), there are three possible cases. The first case is all letters are for one name. The second one is that the name is a feminine name, ending with the feminine (ة T), in a plural form. For example, a name like (فَاطمة fA.tmT) becomes (فَاطمَات fA.tmAt) if there are more than two (فَاطمة fA.tmT). So, the feminine (ة T) of the single form is replaced by the letters (ات At) for the plural form. Therefore, the analyser will take out the last two characters (ات At) and add the feminine (ة T) to return the proper noun to its singular form. But not all plural feminine names which end with these letters (ات At) have the feminine (ة T) in its single form, the single form of a name like (هندَات hndAt) is (هند hnd) not (هندة hndT). Thus, the system will have another possible entry which is the name without the last two letters (ات At) and with no feminine ending (ۀ t).

8.3.5 Article Module

Arabic articles, as recognised by linguists, have three characteristics [Abn-Hesham 1985]:

(a) they do not have a derivation mechanism;

(b) they do not have infix characters;

(c) finally, there is a finite number of them.

An article, however, can have external prefixes and/or suffixes if it is combined with another article or with pronouns, or both.

It should be noticed that only a limited number of articles can have a suffix. Conjunction articles such as (و w) and (ف f) can prefix any other article while articles like (ك k), (ل l), and (ب b) can only prefix articles that refer to nouns.

Because there is no derivation mechanism for Arabic articles and they are limited in number, the article module only works on recognising and identifying articles and pronouns from each other. Therefore, there is no stem to be recognised here. A word like (منهم mnhm), for example, is realised as two words, the preposition (من mn) and the pronoun (هم hm).

8.3.6 Example

The following is an example of how the system works. First, the system gets the Arabic sentence as a list of words. Then, it will try to find the morphemes for each word and its prefix and suffix. Finally, the system combines all of the morpheme's lists into one list to pass it to the parser.

```
Arabic: mn hm alilab walialbat alluyn fy gcm alhacb
        alaly wydrcwn madt prwlwg?

List of morpheme's lists :
[mn,hm,[al,ilab,[]],[[w,al],ialb,at],alluyn,fy,gcm,
 alhacb,alaly,[w,ydrc,wn],madt,prwlwg]

The sentence morphemes list:
[mn,hm,al,ilab,w,al,ialb,at,alluyn,fy,gcm,alhacb,
 alaly,w,ydrc,wn,madt,prwlwg]
```

Note that the list of morphemes which are printed above is for demonstration purposes only - it will not appear during a real run of the system.

8.4 Conclusion

A portable natural language front end system requires a lexicon with words and their syntactic and semantic information. The most characteristic feature of the Arabic language is that it is a highly inflected language. Arabic words can be mostly generated from roots by adding one or more affixes, i.e suffixes, prefixes, infixes or a combination of them.

Although, there are a number of Arabic morphological systems, none of them are available for use. In addition, all of them tend to go down to the basic root level of the word which causes a problem with the word semantics.

ALMHLIL is built to distinguish between two types of morphemes: internal morphemes which are a part of the word's pattern, and external morphemes which are independent words attached to the word but they are not a part of the word's pattern. So, the system focuses on the extraction of morphemes from the various inflexions or forms of any word.

ALMHLIL is domain independent and can be ported to any other Arabic natural language processing system. It has a high degree of portability without requiring the use of knowledge bases and it has the ability to treat unknown words. The system also has the ability to treat inflected Arabic proper nouns without having a database of proper nouns.

§§§§

Chapter 9

The Front End

The Front End of the system takes an Arabic natural language query and generates the equivalent logical query in a form of first order logic by using the Squirrel parser and its semantic interpreter. Therefore, in addition to the Arabic morphological analyser, discussed in the last Chapter, the Arabic Front End contains: syntax and semantic grammar rules, and a lexicon. Figure 9.1 shows these components.

This chapter will discuss the implementation issues relating to the syntactic and semantic grammar which was proposed previously in Chapters 5 and 6. It will also discuss how the grammar and lexicon compilers are used to generate the Arabic grammar rules and lexicon in a Prolog-readable form. The natural language coverage of this system is presented towards the end of the chapter.

9.1 Grammar

Each syntactic rule is associated with a corresponding semantic rule in standard rule-to-rule form, in the following way:

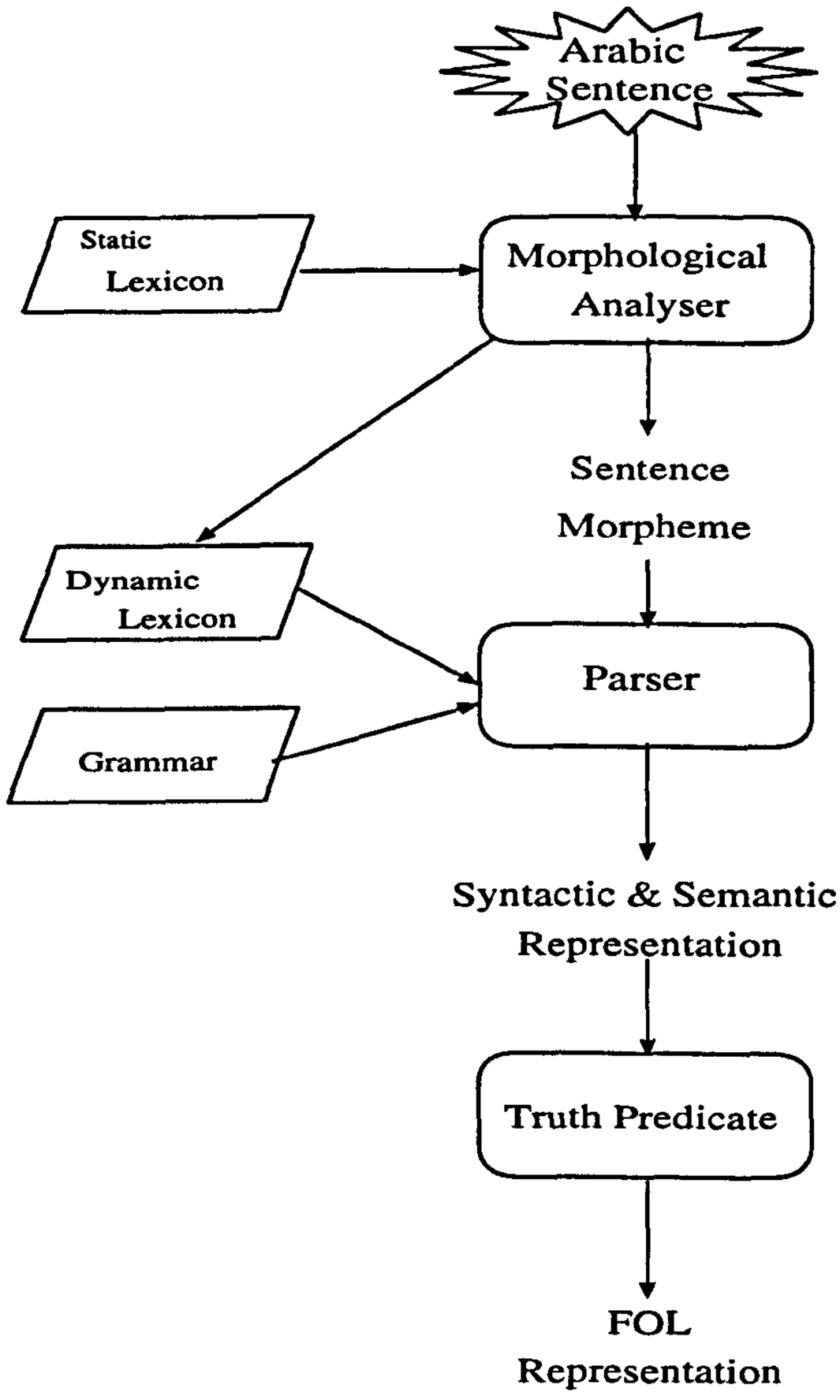

Figure 9.1: Arabic Front End of Squirrel

s	$\longrightarrow$	np , np	$\|\|s\|\|$	=	$\|\|np\|\|(\|\|np\|\|)$
s	$\longrightarrow$	vp	$\|\|s\|\|$	=	$\|\|vp\|\|$
np	$\longrightarrow$	det , n	$\|\|np\|\|$	=	$\|\|det\|\|(\|\|n\|\|)$
vp	$\longrightarrow$	vi	$\|\|vp\|\|$	=	$\|\|vi\|\|$

The format of the syntax and semantic rules are discussed in the following two sections.

9.1.1 Syntactic Rules

The grammar rules are based on the Generalised Phrase Structure Grammar (GPSG) formalism which was proposed earlier in Chapter 5. The following are examples of syntactic rules which are not in prolog format. All rules need to be compiled into a different representation to enable the parser to use term unification, see Section 9.3.

Rule (s3) is for a query which contains two noun phrases: topic (المبتدأ Almbtda') and comment (الخبر Al_hbr). The second rule, rule (s10), is for a query which contains a noun phrase as a topic and a verbal sentence as a comment (الخبر Al_hbr).

```
rule(s3).
s -> [np,np].        b(np).
                     [s,mood]                    = wh.
                     [np(1),case]                = subj.
                     [np(1),agreement]           = [np(2),agreement].
                     [np(1),casetype]            = [np(2),casetype].
                     [np(1),gender]              = [np(2),gender].
                     [np(1),npslash]             = [s,npslash].
                     [np(1),npslash,slabel]      = nil.
                     [np(2),npslash,slabel]      = nil.
                     [np(1),quest]               = yes.
                     [np(2),quest]               = no.
                     [np(2),case]                = obj.

rule(s10).
s -> [np,s].         b(np).
                     [s(1),mood]                 = wh.
                     [np,npslash]                = [s(1),npslash].
                     [np,quest]                  = yes.
```

```
[np,npslash,slabel]          = nil.
[s(2),mood]                  = verbal.
[s(2),neg]                   = no.
[s(2),npslash,agreement]     = [np,agreement].
[s(2),npslash,gender]        = [np,gender].
[s(2),npslash,case]          = [np,case].
[s(2),npslash,casetype]      = [np,casetype].
```

As can be seen, an Arabic sentence may be build up from two noun phrases without any verb, which leads to two phrases with the same category. When a certain category appears more than once in the same rule, the occurrences are distinguished by indexing as in rule(s3).

9.1.2 Semantic Rules

The formal semantics which were proposed in Chapter 6, are based on Montague semantics. That is because we wanted to express a semantics for Arabic in terms of a well known theory. However, semantic rules are implemented based on the Property Theory[1] (PT) formalism, which was introduced by Turner [1988], for two reasons.

Firstly, the PT formalism is essentially based upon that of model-theoretic semantics. It provides alternative treatments to possible worlds without a need for the notion of type information [DeRoeck et al. 1991]. In other words, it can be seen as a substitute for Montague's intensional logic where every expression is explicitly typed. However, Property Theory does have simple type information based on the classical hierarchy of types in which the types are predicates in the language of well-formed formula (wff).

[1] A full explanation of the Property Theory can be found in Turner [1988 1992].

(1) a. $\mathrm{U}(x) \equiv x = x$
b. $\mathrm{P}(x) \equiv P(x)$
c. $\mathrm{R}{\Rightarrow}\mathrm{S}(f) \equiv \forall x(\mathrm{R}(x) \rightarrow \mathrm{S}(fx))$

In the above definition U is the type of everything while P is the type of those objects which are propositions. R⇒S is the type of functions from objects of type R to those of type S, see Turner [1988] for more details. Unlike the notion of Montague semantics types, these notions of types are simple first order predicates and are governed by the proof theory of first order logic. Therefore, PT formalism can be viewed as a flat version of Montague's intensional logic with all the type information and restrictions substituted by the above types.

Secondly, the Squirrel parser and semantic interpreter is based on PT theory. Thus, we need to follow such a theory in order to make use of the system's interpreters.

The corresponding semantic rules for the syntactic grammar rules given above, i.e. `rule(s3)` and `rule(s10)`, will be like:

(2) a. t(np(1)):t(np(2))
b. t(np):t(s(2))

The first rule (2a), for example, represents the semantics of the sentence of `rule(s3)`. This kind of sentence divides into two phrases: the first one a noun phrase which contains the topic (المبتدأ Almbtda'), while the second one contains the comment (الخبر Al_hbr). Its semantic rule states that the semantic value corresponding to an object of syntactic category s is given by applying the semantic value of the object of the syntactic category of the first phrase, i.e. the topic, to that of the object of the second phrase which is the comment. The same can be said for the other rule, rule (2b).

9.2 Lexicon

The system recognises the feature attributes for the lexical entries at two levels: syntactic level and semantic level. Thus, each Arabic word is associated with an initial syntactic value for each feature and category, and a corresponding semantic value.

9.2.1 Syntax Entry

At the syntactic level of every noun, the number, gender, and type of speach are necessary for the syntactic agreement. Also, the case type feature is important for the agreement between the noun and any pronoun refering to it. The following are examples of the lexical entries for nouns, ialb and madt.

```
ialb   n.
       [agreement,num]      = sing.
       [agreement,pers]     = 3.
       [gender]             = masc.
       [casetype]           = human.
       [casesign]           = nil.
       [quest]              = no.
       [nlevel]             = bare.

madt.  n.
       [agreement,num]      = sing.
       [agreement,pers]     = 3.
       [gender]             = fem.
       [casetype]           = nonhuman.
       [casesign]           = nil.
       [quest]              = no.
       [nlevel]             = bare.
```

For a noun lexical entry, the system uses the singular masculine form for regular plural masculine, regular plural feminine, dual masculine, and dual feminine. The irregular plural noun form is used unchanged in the lexical entry because this form cannot be derived using rules. Verbal lexical entries use the single masculine form in past tense. The following is an example of the system's verb lexical entry, drrc:

```
drrc.       v.
                [vtype]              = trans.
                [vform]              = active.
                [vtime]              = past.
                [agreement,num]      = sing.
                [agreement,pers]     = 3.
                [gender]             = masc.
                [vslash,slabel]      = nil.
                [aslash,slabel]      = nil.
```

The system will not keep just the basic root of the word but also its other stems. The reason for that is to keep each word with its semantics. There are two types of lexicon: the static lexicon and the dynamic lexicon. The first one contains the basic lexical entries for the system. The morphological analyser uses this to identify the stems of the word. The dynamic lexicon contains the static lexicon and the new lexical entries built by the morphological analyser. The dynamic lexicon is used by the parser to build the syntax and semantics of the query.

9.2.2 Semantic Entry

As mentioned in Section 3.3, the gender for any animal noun means male or female but for any thing else means either masculine or feminine. Any mammal

common noun is affected by its gender in its referent. For example, a common noun such as (طَالب .tAlb) refers to "male student" while (طَالبَات .tAlbAt) refers to "female student".

(3) من الطَالِب الذي نجح في مَادة بَاسكَال؟
mn Al.tAlib Al_dy nĝ.h fy maAdT bAskAl?
who the student that passed in course pascal.

For example, a query like (3) means that there is a male student and he has passed the Pascal course. Thus, the search in the database should be restricted to the student gender and not to any student who passed Pascal. In the semantic entry, common nouns are grouped into two classes:

A) mammal common noun which takes more than one possible gender thus it is represented with its gender.

B) non-mammal common noun which takes just one possible gender, thus it is represented without its gender.

Therefore, the system recognises the gender attribute in two levels: syntax level and semantic level. In the syntactic level of every noun the gender is recognised as masculine or feminine which is necessary for the syntax agreement. On the other hand, the semantic level is concerned with male or female which will be assigned only to the mammal nouns.

As there are two interpretations for the plural masculine of a mammal common noun (one refers to the male gender alone and the other refers to both genders), we solve that ambiguity by choosing the second interpretation (refer to both genders) as a default and the user can then choose from the answer the correct gender she/he is looking for.

When a proper name is mentioned in a query, the query will not be restricted to a certain gender. Someone may say in Arabic you can tell from the name of the person the gender of that person. But that is not always true, for example

names like (ندي ndy), (فجرfĝr), (نور nwr), etc., can be used for male or female.

Each initial syntactic category is associated with a corresponding semantic value as in the following:

"madt"	n	'madt
"ialb"	n	'ialb:'masc
"fy"	prep	lambda(y, lambda(r, lambda(x, y:lambda(z, 'fy:z:r:x))))
"rcb"	vi	'rcb
"drrc"	vt	lambda(y, lambda(x, y:lambda(z, 'drrc:z:x)))

In the above semantic entries, the semantic values associated with the nouns and intransitive verbs are simple constants. The semantic value associated with preposition "fy" is a three place predicate while for the transitive verb it is a two place predicate, etc.

9.3 Compiling the Grammar and Lexicon

The above representations used in the grammar and lexicon are not directly used by the parser. The grammar and lexicon must be compiled into a different representation to enable the parser to use term unification.

All of the possible paths were expanded in the attributes so that all equations are expressed in terms of the terminal nodes, so the parser can use the more efficient term unification. The possible paths in the attributes of categories are kept in the *gramdef* module. It comprises information about features that each syntactic category may have, for example:

```
category_structure( np,[gender,agreement,casetype,casesign,case,
                        neg,quest,npslash] ).
category_structure( vp,[gender,agreement,neg,npslash,vtime,vform]).
. . . .
```

```
category_structure( v, [gender,agreement,neg,vtype,vtime,vform]).
```

and what other attributes or values these features may have. For the attributes which are declared to be `atomic`, it contains as atomic value only:

```
domain( num,       atomic,  [sing,dual,plur]).
domain( pers,      atomic,  [1,2,3]).
domain( casesign,  atomic,  [acc,nom,gen,nil]).
domain( neg,       atomic,  [yes,no]).
. . . .
domain( vtype,     atomic,  [trans,intrans,bitrans,tritrans]).
```

While if an attribute is declared to be a `feature`, it contains other attributes:

```
domain(npslash,    feature, [slabel,agreement,gender,casetype,
                             case,casesign,quest]).
. . . .
domain(agreement, feature, [num,pers]).
```

The *gramdef* module, also, contains information about the semantic Property Theory type of each category.

```
type_def( s,       p).
type_def( np,      quant).
type_def( vp,      pty).
type_def( n,       pty ).
type_def( v,       quant to pty ).
type_def( propn,   quant ).
. . . .
type_def( coord,   void ).
type_def( neg,     void ).
```

The representation of a sentence, for example, is of type `p` as it is something like a proposition. Verb phrases receive `pty`, the type of properties, whereas noun phrases are `quant`, the type of quantifiers which is `pty⇒p`. The representation of transitive verbs is `quant to pty`. In Property Theory, a category's semantic annotation can belong to more than one type as can be noticed in the above. The type `void` is given for those categories, the co-ordination and negation articles for example, which do not have representation in the PT semantic type.

9.4 Meaning Representation

The expression obtained by the parser is not a logical expression. Thus, after deducing the syntactic structure of the query from the syntactic rules in the grammar, the system invokes the corresponding semantic rules to produce a lambda term for the proposition expressed by this query. Squirrel builds the meaning representation of a query in two stages. The first stage is to represent the query in terms of Property Theory (PT) representation. The second stage is to translate the PT representation into First Order Logic (FOL) representation. These stages are discussed in the following sections.

9.4.1 PT Representation

The initial meaning representation of any query in Squirrel is cast in Property Theory (PT). The use of PT has several advantages. Firstly, PT is an expressive tool from the grammar writer's point of view, because of its lack of a rigid typing structure (see the above rules). Secondly, its preposition treatment is much simplified, as it allows a three-place predicate, one argument of which is a property.

Questions are treated as being constructs which comprise free variables, and

certain words in the lexicon (e.g. *mn, ma, maua,* and so on) are marked to introduce free variables. Since these words may occur more than once, variables are typed and numbered to insure that the introduced free variables are different for each occurrence of this kind of word. Therefore, the word (من *mn* = who) introduces `fva` variable which is typed for *animate*, while (مَا، مَاذَا *ma, maua* = what) introduces the *inanimate* type as `fvi`. As the word *ay* can be bound to animates and inanimates, it introduces `fv`. In the Arabic version of Squirrel, the time and place question tools, (متَى *mty* = when) and (آين *ayn* = where), are typed as `fvt` and `fvp` respectively.

The PT output for the query (من الطلَاب الذين رسبوَا في بَاسكَال؟ mn Al.tlAb Al_dyn rsbwA fy bAskAl? = Who are the students that they failed in Pascal?) is:

```
PT: some(y,every(x,‘ilab: ‘mf:x and‘fy: ‘backal:
                        ‘rcb:x iff x eq y)and‘hw:fva1:y)
```

In the above example, the term `fva1` is a free variable which can range only over *animate* objects. More details about how Property Theory is used in Squirrel are given in DeRoeck et al. [1991].

9.4.2 FOL Representation

Squirrel uses PT representation only as a stepping stone towards a First Order Logic (FOL) representation. After producing the PT representation for the given query, the system then applies the axioms of the truth predicate (which are only expressible in the language of well-formed formula (wff) [DeRoeck et al. 1991]) in order to establish the logical expression (a First Order Logic expression) of this proposition. See [DeRoeck et al. 1991] for axioms of the Truth predicate.

All expressions at this level of meaning representation are domain independent. Property names and relations are derived from the natural language (i.e. Arabic) predicates. The representations obtained at this level of interpretation, for a query like (مَا هي درجَات الطلَاب في قسم كمبيوتر mA hy drĝAt Al.tlAb fy qsm kmbywtr = what grades the students in department Computer), are illustrated in the following:

```
ARABIC: ma hy drjat alilab fy gcm computer.

    PT: some(y,every(x,‘ilab: ‘mf:x iff x eq y)and‘drjat:fvi4:y)
                  and some(a,‘gcm: ‘computer:a and‘fy:a:fvi4)

   FOL: exists(y,all(x,ilab(x,‘mf)<=>x=y)&drjat(y,fvi4))&
                        exists(a,gcm(a,‘computer)&fy(fvi4,a))
```

9.5 Natural Language Coverage

The Arabic fragment described by the current system's grammar covers the following phenomena: generalised quantifiers, relative clauses, yes/no questions, negation, noun/noun modification, count questions, disjunction, and conjunction. Extensive examples of output are in Appendix C to illustrate the natural language capabilities of the system.

§§§§

Chapter 10

The Back End

The Back End of the system takes the output of the Front End to generate the SQL query. It uses information about the given domain, which is stored in the Domain Specific Configuration and the Extended Data Model, and applies it in several translation stages to build up the SQL statement.

This chapter will discuss the customisation process that enables this part to handle Arabic queries. In addition, it will discuss the extensions that have been made to enable it to handle queries for times, places, and counts which are not handled in the original Squirrel.

10.1 Domain Specific Configuration

The Domain Specific Configuration (DSC) contains information about the structure of the database for the current domain of application. It specifies the definitions for each existing relation, its attributes and the domain of each attribute, and its key, as follows:

Relation Definitions

This contains the table names and their attributes together with the domains of the values of those attributes. The following, for example, is the entry for the table `student` and its attributes:

```
student isTABLE [stname:st_name,stnum:st_num,
                 gender:st_gender,address:st_address,
                 degree:st_degree,major:st_major,
                 dept:st_department,advisor:fac_num,
                 registration:st_year].
```

Domain Attributes

Each attribute in a relation is defined with the possible type of value it can have in the given domain. The domain value type can be either a string or a number while the type of that domain, for example, can be animate or inanimate.

```
st_name      isDOMAIN [student![stname], string, both].
st_gender    isDOMAIN [student![gender], string, animate].
st_address   isDOMAIN [student![address], string, inanimate].
. . . .
tch_year     isDOMAIN [teach![tchyear], number, time].
cors_dept    isDOMAIN [course![dept], string, place].
```

The original Squirrel restricted the type of any domain attribute to be either animate or inanimate. However, the domain attribute has been extended to include types for time and place queries.

Relation Key

In relational databases any relation requires one primary key attribute. The relation of student, for example, has the field `stnum`, which means student number, as a key. These keys are used in the optimisation of the queries.

```
student      hasKEY [stnum].
faculty      hasKEY [facnum].
. . . .
course       hasKEY [corsnum].
```

10.2 Extended Data Model

One of the Squirrel characteristics is the Extended Data Model (EDM). It provides the necessary information which maps the predicates in first order logic onto database constructs. For every predicate in the First Order Logic (FOL), which originates from the natural language input words, there is one or more corresponding database meaning associated with it in the EDM.

The information is represented, in the EDM, as a set of equivalences of the form:

$$\pi(\upsilon, \zeta) \leftrightarrow \phi$$

The π is a predicate in the FOL representation having the υ and ζ as individual variables and ϕ as any fragment which replaces it in the domain relational calculus. The following are examples of the mapping rules in the EDM:

```
madt(X) <=> [*,X,*,*,*]:course.
madt(X) <=> [X,*,*,*,*]:course.
drjt(Y,X) <=> [Y,*,*,*,X]:grade.
ialb(X,Y,Z) <=> [X,Y,Z,*,*,*,*,*,*]:student.
```

```
. . . .
rcb(X) <=> exists( bb,[X,bb,*,*,*,*,*,*,*]:student &
                                    [bb,*,*,*,'f]:grade ).
drrc(X,Y) <=> exists(bb,
                        exists(ii, [bb,*,*,X]:teach &
                        [ii,bb,*,*,*]:grade &
                        [Y,ii,*,*,*,*,*,*,*]
                        :student ) ).
```

The first rule says that some X is a course number if it occurs in the second attribute of the course relation. As can be seen from the above rules, some predicates require a join over two or more relations. Also, a FOL predicate may have more than one meaning representation in respect of the current database. In the first two rules, each is presented in a separate rule. Based on the Squirrel ambiguity resolution, all possible interpretations are released for further processing and filtering.

10.3 Translation Stages

The Back End uses the output representation which is released by the Front End in the form of FOL to produce the SQL, as a final representation of the query. As can be seen from Figure 10.1, this is done through several stages of translation as discussed bellow.

10.3.1 Untyped Relational Calculus

The Untyped Relational Calculus (URC) is also called the Universal Relational Calculus in some Squirrel documents. The expressions that appear in the FOL representation have no particular significance, by themselves, in the specific database being used. So, the first step towards producing a database-

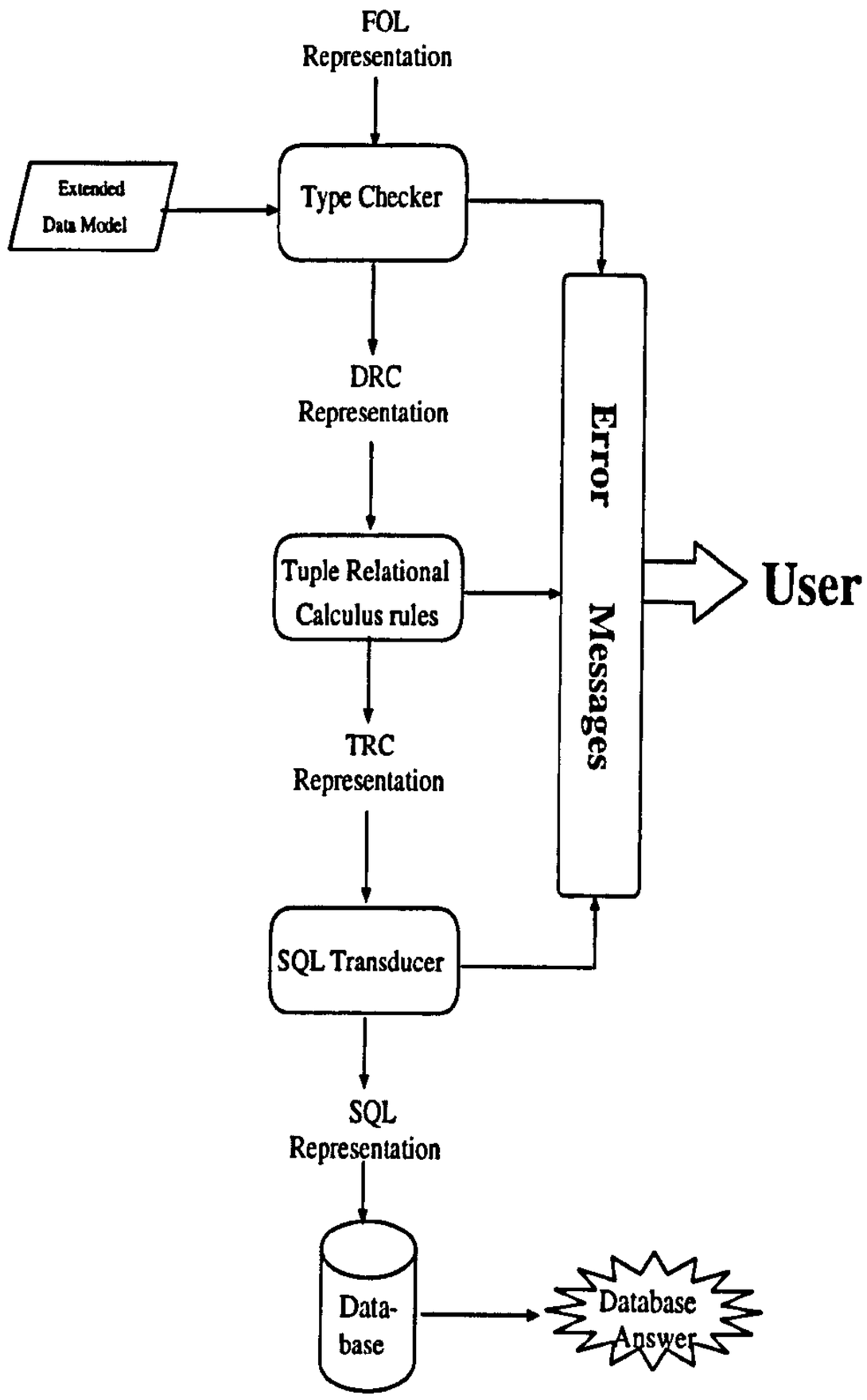

Figure 10.1: Back End of Squirrel

specific SQL query is obtained by relating the meaning of FOL predicates to the corresponding expressions in the database domain.

The translation process from FOL to URC is done by applying the mapping rules in the EDM (Section 10.2) which contains entries in the form:

$$\texttt{FOL} \Longleftrightarrow \texttt{URC}$$

The following is an example of the output obtained by the URC translation process for the query (أَي الطلَاب في مَادة com321؟ 'ay Al.tlAb fy mAdT com321?).

```
URC: {fvi3;exists(y,all(x,[*,x,*,*,*,*,*,*,*]:student<=>
     x=y)&y=fvi3)&exists(a,[‘com321,a,*,*,*]:course&
     [fvi3,a,*,*,*]:grade)}
```

10.3.2 Domain Relational Calculus

Because the variables in the relational calculus query languages are typed, it is important to find the appropriate types for the variables. The Domain Specific Configuration (DSC) involves making the domains of variables explicit and enforcing type knowledge of the structure of the database.

This stage of translation uses information about the database structure which is in the DSC module to enforce type checking. The type checking will act as a filter for the URC presentations. The following is the Domain Relational Calculus (DRC) representation and its checking output for the query (مَا هي درجَات الطلَاب في قسم الكمبيوتر؟ mA hy drĝAt Al.tlAb fy qsm Alkmbywtr?):

```
ARABIC:  ma hy drjat al ilab fy gcm computer.
   URC: {fvi4;exists(y,all(x,[*,x,*,*,*,*,*,*,*]:student<=>x=y)&
        [*,y,*,*,fvi4]:grade)&exists(a,([*,*,*,*,*,‘computer,*,*]:
        faculty&[*,*,*,*,*,a,*,*]:faculty&a= ‘computer)&
        [fvi4,a,*,*,*]:grade)}
   CHK: Query not meaningful with respect to current database
   URC: {fvi4;exists(y,all(x,[*,x,*,*,*,*,*,*,*]:student<=>x=y)&
        [*,y,*,*,fvi4]:grade)&exists(a,([*,*,*,*,*,‘computer,*,*]:
        faculty&[*,*,*,*,*,a,*,*]:faculty&a= ‘computer)&
        exists(bb,[fvi4,bb,*,*,*]:grade&[a,bb,*,*,*]:course))}
CHK: Query not meaningful with respect to current database
URC: {fvi4;exists(y,all(x,[*,x,*,*,*,*,*,*,*]:student<=>x=y)&
     [*,y,*,*,fvi4]:grade)&exists(a,([*,*,*,*,*,‘computer,*,*]:
     faculty&[*,*,*,*,*,a,*,*]:faculty&a= ‘computer)&
```

```
        [fvi4,*,*,*,*,*,a,*,*]:student)}
   CHK: Query not meaningful with respect to current database
   URC: {fvi4;exists(y,all(x,[*,x,*,*,*,*,*,*,*]:student<=>x=y)&
        [*,y,*,*,fvi4]:grade)&exists(a,([*,*,*,*,*,'computer,*,*]:
        faculty&[*,*,*,*,*,a,*,*]:faculty& a= 'computer)&
        [fvi4,*,*,*,*,a,*,*]:faculty)}
   CHK: Query not meaningful with respect to current database
. . . .
. . . .
   URC: {fvi4;exists(y,all(x,[*,x,*,*,*,*,*,*,*]:student<=>x=y)&
        [y,*,*,*,fvi4]:grade)&exists(a,([*,*,*,*,*,*,'computer,*,*]:
        student&[*,*,*,*,*,*,a,*,*]:student&a= 'computer)&
        exists(bb,[bb,*,*,*,fvi4]:grade&[*,bb,*,*,*,*,a,*,*]:
        student))}
   DRC: {fvi4:grade![grade];exists(y:student![stnum],
         all(x:student![stnum],[*,x,*,*,*,*,*,*,*]:student<=>
         x=y)&[y,*,*,*,fvi4]:grade)&exists(a:student![dept],
         ([*,*,*,*,*,*,'computer,*,*]:student&[*,*,*,*,*,*,a,*,*]:
         student&a= 'computer)&exists(bb:student![stnum],
         [bb,*,*,*,fvi4]:grade&[*,bb,*,*,*,*,a,*,*]:student))}
```

As can be seen, the only differences between the syntax of DRC and URC is that domains are explicitly associated with variables. The variables y, x, a and bb in the above URC become y:student![stnum], x:student![stnum], a:student![dept] and bb:student![stnum] respectively.

10.3.3 First Optimisation (OP1)

This stage is carried out once the query has been translated into DRC. It is concerned with the removal of tuple membership clauses which have been made redundant by the explicit association of domains with variables. Consider the

following DRC query which corresponds to the question (مَا أَسمَاء المَوَاد؟ mA 'asmA' AlmwAd?):

```
DRC: {fvi1:course![title];exists(y:course![corsnum],
     all(x:course![corsnum],[*,x,*,*,*]:course<=>
                    x=y)&[fvi1,y,*,*,*]:course)}
```

The variable x is associated with the domain which is the projection on the course number attribute corsnum of the course relation. As can be seen, the tuple membership clause is redundant as x and y are variables for the same attribute. The OP1 stage performs optimisation by converting the DRC expression into clausal form and removing the redundant clauses. Therefore, the above example produces the following clause set:

$$\{\langle *, x, *, *, * \rangle \in course\}$$
$$\{\langle fvi1, y, *, *, * \rangle \in course\}$$

Removing the redundant clause produces the single clause:

$$\{\langle fvi1, skf1, *, *, * \rangle \in course\}$$

Then the query is formed by reconstructing that single clause:

```
OP1: {fvi1:course![title];exists(skf1:course!
                [corsnum],all(x6:course![corsnum],
                 x6=skf1&[fvi1,skf1,*,*,*]:course))}
```

10.3.4 Tuple Relational Calculus (TRC)

In the DRC representation the object attributes are constrained to occur in certain positions in particular relations, while SQL's quantifiers range over relations. SQL does not need to know where a particular attribute appears

in the tables as it picks up the appropriate attribute using a field name. The TRC representation is close to SQL since variables must range over relations.

```
DRC: {fva2:student![stname];exists(y:student![stname],
     all(x:student![stname],[x,*,'masc,*,*,*,*,*,*]:student&
     x= 'bdr<=>x=y)&y=fva2)}
```

The TRC output for the above DRC query, which is for query "من هو الطَالب بدر؟", is:

```
TRC: {fva2!stname;fva2:student;exists(skf1:student,
       all(x7:student,(x7!stname=skf1!stname=>exists(tuple1:
           student,tuple1!stname=x7!stname&tuple1!gender=masc))&
           (x7!stname=skf1!stname=>x7!stname=bdr)&(exists(tuple2:
            student,tuple2!stname=x7!stname&tuple2!gender=masc)&
            x7!stname=bdr=>x7!stname=skf1!stname)&skf1!stname=
            fva2!stname))}
```

10.3.5 Second Optimisation (OP2)

The translation of DRC relation membership clauses always results in extra existentially quantified variables in the TRC expression. Consider the following TRC representation which corresponds to the question (مَا أَسمَاء المَوَاد؟ mA 'asmA' AlmwAd?):

```
TRC: {fvi1!title;fvi1:course;exists(skf1:course,
       all(x6:course,x6!corsnum=skf1!corsnum&
       exists(tuple1:course,tuple1!title=fvi1!title&
              tuple1!corsnum=skf1!corsnum)))}
```

In the TRC query, above, the existentially quantified variable, `skf1` has become redundant. This stage performs optimisation by translating the TRC expression into clausal form and applying the para-modulation inference rule, which substitutes one term for another which is equal, until the redundant variables have been eliminated. The above TRC query is converted to the following clause sets:

$$\{x6.corsnum = skf1.corsnum\}\ \{tuple.title = fvi1.title\}$$

$$\{tuple1.corsnum = skf1.corsnum\}$$

Skolem constants have been formed by prefixing variable names with *sk*. As can be seen, all occurrences of $skf1.corsnum$ can be replaced by $x6.corsnum$. This substitution reduces the clause sets and produces the following simplified OP2 for the above TRC query:

```
OP2: {fvi1!title;fvi1:course;all(x6:course,
        exists(sk2:course,sk2!title=fvi1!title&
                sk2!corsnum=x6!corsnum))}
```

10.3.6 SQL

This is the final stage of the Back End. It has a direct mapping between the optimised TRC and SQL expressions. Because the Tuple Relation Calculus is a generic query language, which is expressed in a set-theoretic notation, the translation of this stage is the most straightforward.

Applying this stage of translation to the optimised TRC of the qury مَا أَسمَاء المَوَاد؟ mA 'asmA' AlmwAd?), above, produces the following SQL statement:

```
SQL: SELECT DISTINCT fvi1.title
```

```
FROM course fvi1
WHERE 1=ALL(SELECT 0
            FROM course x6
            WHERE NOT EXISTS(SELECT *
                             FROM course sk2
                             WHERE (sk2.title=fvi1.title AND
                                    sk2.corsnum=x6.corsnum)))
```

The only data that can be abstracted from the relation are the fields which are represented by the free variables. In the above example, the free variable `fvi1.title` is translated as the only field to be retrieved from the tuple-that is the purpose of `DISTINCT`.

Yes/no questions which contain no free variables to be abstracted, are translated in a different way. Squirrel defines a unary relation in the database called *ANSWER* which contains an attribute called *answer* as well. This attribute can be associated only with a unique value which is the constant `YES`. The following, for instance, is the SQL for the query (أ رسب أحمد في مَادة برولوق؟ 'a rsb 'a.hmd fy mAdT brwlwq?):

```
SQL: SELECT DISTINCT a.answer
     FROM answer a
     WHERE EXISTS
       (SELECT *
        FROM course sk3
        WHERE EXISTS
          (SELECT *
           FROM student sk4
           WHERE EXISTS
             (SELECT *
              FROM grade sk5
              WHERE (sk3.title='prolog' AND
```

```
(sk4.stname='ahmed' AND
 (sk5.grade='f' AND
  (sk5.stnum=sk4.stnum AND
   sk5.corsnum=sk3.corsnum))))))
```

If the answer of the above query is true with respect to the current database, the system returns the answer relation, which contains the value YES. If it is not true, the system returns the empty relation which means NO.

The original Squirrel is unable to deal with count questions, i.e. how many...?, which does not refer to a specific data item to be abstracted from the database nor to a logical query, i.e. yes/no question. This type of query refers to the count of the tuples that satisfy the query conditions. However, the back end is extended to handle this kind of query. For example, the SQL statement for a query (كم طَالبًا في مَادة برولوق؟ km .tAlbA fy mAdT brwlwq?), which means "How many students are there on the course Prolog?", is:

```
SQL: SELECT COUNT (*)
     FROM student fvn3
     WHERE EXISTS
       (SELECT *
       FROM student sk3
       WHERE EXISTS
         (SELECT *
          FROM course sk4
          WHERE EXISTS
            (SELECT *
FROM student sk5
WHERE EXISTS
  (SELECT *
   FROM grade sk6
   WHERE (sk3.stname=fvn3.stname AND
```

```
(sk3.gender='masc' AND
 (sk4.title='prolog' AND
  (sk5.stname=fvn3.stname AND
   (sk6.stnum=sk5.stnum AND
    sk6.corsnum=sk4.corsnum)))))))
```

10.4 Pragmatics

Natural languages are very complex and different from any formal language. The meaning of the words cannot be formally defined, they are ambiguous, and the interpretation of natural language is pragmatic. Therefore the automated parsing of natural languages is not a straightforward task. Work in natural language interfaces has developed from intra-sentential syntactic and semantics analysis to levels where pragmatic analysis is necessary [Kaplan and Bresnan 1982].

Morris studied these three analysis levels and defined the relationship btween them. Syntax is the way signs are related to other signs while semantics is the way signs are related to corresponding objects in the world. Pragmatics, however, is concerned with how signs relate to the users of the sign system or language [Morris 1938].

Pragmatically, the following query (1), for example, may have a different meaning according to the system domain as the user may be concerned with a student called Badr or a member of staff called Badr and so on.

1) من بدر؟
   mn Badr?
   who is Badr?

Any ambiguity in the sense of more than one meaning for the above query will be solved by building SQL statment for each possible read for the query in

respect to the current database.

§§§§

Chapter 11

System Evaluation

Evaluation is very useful as a development aid even if you know that your system cannot do very much or anything like as much as you want it to [Sparck Jones and Galliers 1996].

This chapter will propose the evaluation methodology for the system then discuss the evaluation results. But before that it will give an overview of evaluation in the natural language interface area.

11.1 Background

Before discussing our methodology of evaluation, it is better to overview the evaluation methods which have been used in this area of research.

Database Interface evaluation has been a concern since the early seventies. [Woods 1973] describes the informal testing of the LUNAR system. The informal evaluation consisted of running a demonstration of the system at the 2nd annual Lunar Science Conference twice a day for three days. The geologists attending the conference were invited to ask the system questions and he

then analysed the system's responses.

The Transformational Question Answering (TQA) system was an experimental NLI to a town planning database. It was evaluated by testing the system in operation with its intended end users. TQA was placed in operation for evaluation from late 1977 through 1979. It had two logging facilities, one was a verbatim record of all output on the user terminal and the other one was a comprehensive trace of the system flow whilst it processed each query. The results for a whole year were statistically analysed to produce measures per month of number of questions, completed, aborted, parsing failure, etc. [Sparck Jones and Galliers 1996].

Jarke et al. [1985] describe a comparative evaluation between natural language and an artificial query language, SQL, as to the practical usefulness of natural language in such a context.

Whittaker and Walker [1989] describe the evaluation of a restricted NL as an interface for a database in comparison with a menu interface system. They contrast their approach in terms of the interpretation of results with Jarke's study.

The Airline Travel Information System (ATIS) is a database of flights and information on aircraft, stop, connections, meals, etc. It is being used as a common test and evaluation base by teams participating in the Spoken Language Systems Conferences (SLS conferences). Questions are collected initially as raw data using 'Wizard scenarios'.

TSNLP project (Test Suite for Natural Language Processing) [Balkan et al. 1995] provides a methodology to construct substantial amounts of test data in three European languages.

11.2 Evaluation Methodology

From the above, with exception of ATIS and TSNLP, all the evaluations were done by end users. The purpose of Woods and TQA evaluation was to monitor the performance of the system while the purpose of Jarke et al and, Whittaker and Walker was to compare the performance of the NLI against the other interfaces. ATIS used Wizard of Oz (WOZ) simulations which are difficult, labour intensive, and time consuming plus the subjects are convinced that they are dealing with a computer system not a human wizard. That means it will result in unrealistic behaviour of both the system and the users. Data collection for the ATIS application was in some cases done in controlled WOZ experiments without sufficient regard for its implications [Sparck Jones and Galliers 1996]. TSNLP has produced substantial multi-purpose and multi-user test suites but it is restricted to the French, English and German languages.

The TSNLP cannot be used to test the Arabic Front-End system because of the language restriction on TSNLP. The same can be said for ATIS which was for the English airline travel domain. The purpose of the Arabic Front-End system is not to compare the performance of natural language against other interfaces, thus Jarke et al., and Whittaker and Walke methodologies are not appropriate as well.

Therefore, the evaluation methodology of the Arabic Front-End system will be as follows:

A) Ask a number of Arabic native speakers by e-mail to send possible queries for student-faculty database and then these queries will be passed to the system to see the results.

B) Let a number of subjects use the system and monitor the results.

C) Apply the system to another domain to test its portability.

Methods A and B are used to test the system performance while method C is

used to test the system portability.

11.3 Performance Tests

This test was carried out into two stages. The first stage is to test the system against a collection of queries. The other one is use subjects to query the system. These stages are discussed in the following sections.

11.3.1 Collection Test

For this type of test, we managed to collect 150 queries, about the student-faculty database domain, from a number of Arabic speakers. These queries were stored in a file and processed by the system off line.

Table 11.1: System performance

	Total	Succeed	Fail
No. of queries	150	130	20
Percentage	100%	86.67%	13.33%

Table 11.1 shows the overall results of the system. 130 queries were completed successfully which makes the overall performance of the system 86.67%. This compares well with other natural language interface systems. The results of LUNAR, for example, indicate that 10% of inputs resulted in parsing or semantic problems and 12% of inputs failed due to coding errors in the system. This means that the system could process only about 78% of its inputs.

The system failed to generate SQL statements for 20 queries. However, it managed to translate most of them to some degree as shown in Table 11.2. 35% of them cannot be translated beyond the first optimisation (OP1) translation

Table 11.2: Analyses of the failed queries

Last translation obtained	Syntax	FOL	OP1	Not parsed
No. of queries	3	4	7	6
Percentage	15%	20%	35%	30%

stage. It could generate only the first order logic (FOL)for 20% of the failed queries. At the parsing stage only 15% which parsed syntactically without semantic representation. Only 30% of the failed queries could not be parsed by the system.

Table 11.3: The Back End and Front End performance

	Front End performance	Back End performance
No. of queries	141	130
Percentage	94.0%	92.20%

In general, the Front End of the system could generate the FOL for 141 queries out of 150 queries which makes its performance 94.0% as shown in Table 11.3. The Back End could handle 130 queries out of 141 FOL queries, taking out the queries which failed at the Front End part, makes its performance 92.20%.

11.3.2 Subjects Test

The other performance test is obtained by users. Four Arabic speaker subjects who know the SQL language have been asked to play with system. All results of each subject are stored in a separate file for analysis. Table 11.4 shows the result where each subject is given a number.

The total number of queries submitted by the users is 102 queries. The system managed to generate SQL for about 52.94% of them. 8.82% of the input queries

Table 11.4: The results of the users' test

Subject No.	Total Queries	Failed to parse	Translated to FOL only	Translated to SQL
Subject 1	22	8 36.36%	- -	14 63.64%
Subject 2	27	13 48.15%	3 11.11%	11 40.74%
Subject 3	23	5 21.74%	4 17.39%	14 60.87%
Subject 4	30	13 43.33%	2 6.67%	15 50%
Total	102 100%	39 38.24%	9 8.82%	54 52.94%

could only be translated to FOL. This leaves about 38.24% of the input queries as unparsed queries.

It is obvious that the performance of the system in this test is low compared with the result of the previous test. This does not reflect the actual performance of the system as most of the causes of the failed queries are due to subject errors.

By analysing the failed queries, see Table11.5, we found that the major cause for them to fail is typing errors. Subject 3, for example, in that table shows that typing errors were the only reasons for most of the unparsed queries. Also, 69.23% of the failed queries of Subject 4 arose because of typing errors. In this test, typing errors were behind 64.1% of the failed queries, 25 queries out of 39 unparsed queries.

The second major cause are syntactic errors. 37.5% of the failed queries by Subject 1 were syntactically incorrect. Syntax errors were the cause of 17.95% of the unparsed queries of this test. These are user errors and should not be used as an indication of the system performance.

Table 11.5: The cause of failure of parsing

Subject No.	Typing Errors	Syntax Errors	Word not in the Lexicon	No grammar rule	Total fail
Subject 1	3 37.5%	3 37.5%	2 25%	- -	8
Subject 2	8 61.54%	2 15.38%	2 15.38%	1 7.69%	13
Subject 3	5 100%	- -	- -	- -	5
Subject 4	9 69.23%	2 15.38%	- -	2 15.38%	13
Total	25 64.1%	7 17.95%	4 10.26%	3 7.69%	39 100%

Typing errors are a common problem in human/machine interactions. Also, the use of Latin characters to represent Arabic was behind these errors as subjects give most of their attention in finding the characters and do not verify their entries. In addition, some subjects mentioned that they were intending to test the system by queries which are syntactically incorrect to see the system response.

In this test, some queries contained words not in the system's lexicon. Also, some queries could not be parsed because there were no grammar rule to handle them. 10.26% of the unparsed queries were because of the lack of lexical entries and 7.69% were because of the lack of grammar rules. This is because we are building a prototype and not a complete system. Of course, these can be fixed if we add more entries in the lexicon and add more grammar rules.

11.4 Portability Test

For testing its portability, the system was ported to access an example of a library database. The data model of the database is relational. The database

contains four entities: borrowers, items, loans and reservation. All the relations between them are one to many as shown in Figure 11.1.

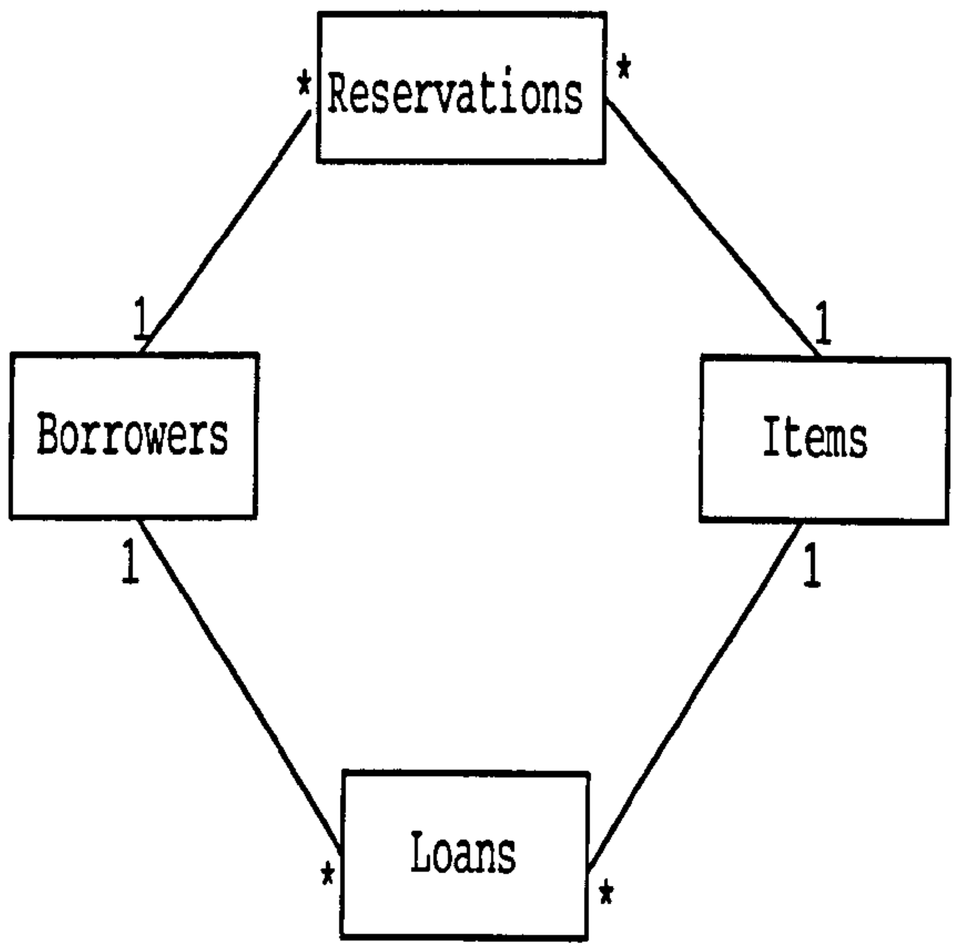

Figure 11.1: Data structure of the new domain

All of these entities will be kept in different tables. The following represents the database tables structure.

Borrowers

bor_name	bor_num	bor_gender	bor_address	bor_year

Items

item_title	item_num	item_author	item_place	item_loantype

Loans

bor_num	item_num	bor_date	due_date	return_mark

Reservation

bor_num	item_num	res_date	note_mark

The portability process can be achieved by replacing the domain dependent information which is stored in the Back End. Information about the structure of the database of a given domain of application, is kept in the Domain Specific Configuration (DSC). The DSC specifies the definitions for each existing relation, its attributes and the domain of each attribute, and its key. Therefore, a new DSC needs to be established in order to port the system to the new domain.

Also, the portability process requires an update for the Extended Data Model (EDM) which provides the necessary information which maps the predicates in first order logic onto database constructs. For every predicate in the First Order Logic (FOL) there is one or more corresponding meanings associated with it in the EDM of the database domain.

As can be seen, there is no need to change the Front End or the translation process of the Back End. This makes the portability process much easier and requires little time and effort. Samples of this domain's queries are given in Appendix B.

11.4.1 Portability Evaluation

After porting the system to the new domain, it is important to evaluate the portability of the system and its performance on the new domain. Therefore, we tested the system with 48 queries about the new domain which is the library database. The result is presented in Table 11.6.

From Table 11.6, the system managed to parse and generate the SQL statments for 39 queries of the new domain. This makes the performance of the system about 81.25% for the new domain. The Front End of the system could parse and generate the FOL for 41 queries out of 48 queries. The Back End of the system cannot translate two queries from the FOL to SQL.

Table 11.6: The results of the new domain's test

	Total Queries	Failed to parse	Translated to FOL	Translated to SQL
NO. of queries	48	7	41	39
Percentage	100%	14.58%	85.42%	81.25%

The system failed to parse about 14.58% of the input queries. By analysing the unparsed queries, see Table 11.7, we found that four of them failed because of the typing errors while three of them failed because there is no grammar rule to parse them. In this test, there is no query that failed because of syntax errors or because there is no lexical entry.

Table 11.7: The cause of failure of parsing

Typing Errors	Syntax Errors	Word not in the Lexicon	No grammar rule	Total fail
4	-	-	3	7
57.14%	-	-	42.86%	100%

Comparing this result, Table 11.7, with the result of Table 11.5, we found that the typing errors are the first cause of parsing failure in the both domains. The second cause of failure for the new domain is the lack of grammar rules while in Table 11.5 the second cause was the syntax errors and the lack of grammar rules came as the fourth cause.

If we ignore the typing errors queries, the performance of the system would be 39 queries out of 44 queries which is 88.64% and the Front End would be 41 queries out of 44 queries which means 93.75% successfuly treated. This evaluation shows that the performance of the system on the new domain is quite colser to the performance of the system for the previous domain which is 94.0% where there are no typing errors (see Table 11.3).

11.5 Squirrel Problems

One problem with Squirrel is the interpretation of "and". Squirrel as it stands has one interpretation for the "and" conjunction which is the logical "and". Of course this is not always true. For instance:

Who passed Prolog and failed in C++?	(S) level	should receive "and"
The students and the teacher went for lunch	(NP) level	should receive "and"
List professors and lecturers	(N) level	should receive "or"

Thus the appropriate interpretation for "and" in an NL query depends on the type of objects that are coordinated. In order to get the right interpretation we need contextual information which is incompatible with the high degree of portability of Squirrel.

Another problem is a pragmatic problem. The query answering mechanism of Squirrel is to retrieve one single field from the tuple of the table for wh questions or return a boolean answer for yes/no questions. That means the answer of query Q1 and Q2 is A1 and A2 as follow:

Q1: who is Mike?

A1: 96343551

Q2: what are the grades of students in Java?

A2: 91

87

78

95

.

etc.

When the user sends a query like Q1 pragmatically he may means one of the following interpretation:

I1: what is the ID number of Mike?

I2: Give me all the details you have about Mike?

The system chooses the first interpretation I1 to answer Q1 as in A1. If the user intends the second interpretation I2 he needs to use a separate query for every single item of data about the student (i.e. a query for student's address, a query for the student's department, a query for the student's advisor, etc.). The reason for that is that Squirrel's back end as it stands does not have the ability to retrieve the whole tuple of the table.

The system answers query Q2 by presenting a list of student grades only without any information about to whom this grade belongs (i.e. student name or student number). That is an unhelpful form of answer because pragmatically the user needs the names to go with these grades.

Finally, the back end of Squirrel sometimes generates more than one correct SQL statement. This happens when there is more than one mapping rule satisfied in the translation process from FOL to URC (see Section 10.2).

§§§§

Chapter 12

Conclusion and Future Work

Natural Language Interface (LNI) systems simplify the interaction between computer and user by allowing users to communicate with a computer without learning specialised programming languages. Non-technical users who do not know any artificial language must be able to operate such systems. These kinds of systems handle user queries and responses in a conversational style instead of via commands. A large number of NLI systems is available based on the English language.

Research on Arabic natural language processing is in its early stages compared with what has been done for other languages. Existing systems had made a good start in treating Arabic interrogative sentences. Each system tries to use one or more computational linguistic theories to parse Arabic. However, there has been little work done to build a domain independent meaning representation for Arabic interrogative sentences.

In the present work, we have used GPSG to analyse Arabic statements. The focus is mainly to provide syntactic analyses based on correct Arabic linguistic principles. We have shown that GPSG is a powerful syntactic theory not only for Indo-European languages but also for Semitic languages such as Arabic.

Formal semantics is very important in question answering systems because it expresses the meaning of the user's question in terms of high level world concepts, which are independent of the database structure. Thus it is not difficult to port systems based on this approach to other databases or knowledge domains. We have proposed and implemented treatments for Arabic statements to build well-formed expressions based on formal semantics.

We have proposed a semantic classification for Arabic sentences, based on the type of function they have. From this classification we have shown that the semantics of Arabic sentences cannot always be built by applying the semantics of one part to the other one. Some sentences contain open formulae which need to be bounded before the application process. In addition, the application process needs to be done by a special rule at the sentence level as there is no form of formula in the sentence parts. Well-formed propositions are produced for Arabic fragments without the need for the idea of a copular deletion.

The proposed treatments have been proved and tested by building a prototype system. The prototype is implemented using Squirrel, one of the NLI existing systems. The decision has been made to use that system based on the achievement of a high degree of portability.

An Arabic morphological analyser, called ALMHLIL, has been built to distinguish between two types of morphemes: internal morphemes which are a part of the word's pattern, and external morphemes which are independent words attached to the word but which are not a part of the word's pattern. So, the system focuses on the extraction of morphemes from the various inflexions or forms of any word. Although ALMHLIL is used with the Squirrel system, it is domain independent and can be ported to any other Arabic natural language processing system.

One problem we faced during the development of the system is that there is no general Arabic lexicon available to use. So, we built our own lexicon for the domains we need. We may expand the lexicon to make it domain independent in the near future. Another problem was during the evaluation stage as there

is no corpus or collection of Arabic queries to test the system. We created our own collection to test the system. This collection may be also expanded in the future to make it more general and domain independent for any question answering system.

The system has been evaluated through two different domains. Unfortunately, we cannot compare the evaluation results of the system with the previous Arabic systems as there are no evaluation figures mentioned to them. The results were 86.67% of the input queries of the first domain and 88.64% of the second domain were successfully translated to SQL. This compares well with other natural language interface systems such as LUNAR, where it could process only about 78% of its inputs.

It is obvious from the evaluation results that the major problem of the unparsed queries is the typing errors. This happened because we are using a Latin characters to represent the Arabic characters. This problem can be minimized if the system runs under an Arabic environment and used the Arabic characters. Also, there is a need for more grammar rules and more lexical entries.

The most possibility for further extension of the system would be to make it handle a dialogue as there is no such a system doing this for Arabic. Consider, for example, the following queries:

Q1) من يدرّس مَادة بَاسكَال؟
mn ydrrs mAdT bAskAl.
Who is teaching course Pascal?

Q2) هل هو في قسم الحَاسب؟
hl hw fy qsm Al.hAsb
Is he in Computer department?

Q3) هل يدرّس مَادة آخرَى؟
hl ydrrs mAdT 'A_hr_A.
Does he teach another course?

Both queries Q2 and Q3 are refering to the person of the answer to Q1. For this type of queries the system needs a module for coreference resolution to be built. With this module the system should keep the queries so it can refers to them later whenever there is a query which contains pronouns to be resolved. The coreference resolution can be fitted in our system after the generation of the PT representation to resolve anaphora before building the FOL representation.

Another possible extension to the system is to try to make the system work as a multilingual interface, i.e. Arabic/English. This could be achieved by joining the Arabic version of Squirrel with the English one. Of course there is a need for modules to ensure that the system will give the same result for both languages.

§§§§

Bibliography

M. Abdullatif. في بنَاء الجملة العربية *fy bena al jumlh al3rabyah.* Dar Algalam, Kiwait, 1982.

J. Abn-Hesham. مغني اللّبيب عن كتب الاَعَاريب *mugni allabeeb 3n kutub ala3areeb.* Dar Al-Feker, Byrrot, six edition, 1985.

A. Abu-Arafah. *A Grammar for the Arabic Language Suitable for Machine Parsing and Automatic Text Generation.* PhD thesis, Computer Science Department, Illinois Institute of Technology, Chicago, IL, USA, 1995.

S. Al-Fedaghi and F. Al-Anzi. A new algorithm to generate arabic root-pattern forms. In *Proceedings of the 11th National Computer Conference and Exhibition*, Dharan, Saudi Arabia, 1989.

S. Al-jabri. A syntactic analyzer for arabic. Master's thesis, Computer Science and Engineering Dept., King Fahd University of Petroleum and Minerals., 1989.

S. Al-jabri. *Arabic Word Generation from Semantic Descriptions.* PhD thesis, Department of Artificial Intelligence, University of Edinburgh, 1997.

B. Al-Johar and J. McGregor. A logical meaning representation for arabic. In *Proceedings of the 15th National Computer Conference and Exhibition*, Dharan, Saudi Arabia, 1997.

B. Al-Johar and J. McGregor. An arabic natural language interface to a database system. In *the 6th International Conference and Exhibition on Multilingual computing , ICEMCO-98*, pages 4.2.1–4.2.9. Cambridge University, 1998.

M. Al-Khonaizi, A. Al-Zobaidie, and M. Al-A'ali. Al-khazoon: An arabic language text database system. In E. Dittres, editor, *Processing Arabic*, 8,

pages 107–114, Holland, 1995. Nijmegen University, Institute for the Languages and Cultures of the Middle East.

H. Al-Muhtaseb and M. Khayat. Natural arabic understanding system (naus). In *The Proceedings of the Regional Conference on Informatics and Arabization*, Tunis, 1988.

S. Al-Safran and M. Khayat. An arabic sentence generator. In *Proceedings of the 3rd National Computer Conference*, Riyadh, Saudi Arabia, 1992.

A. Al-Sawadi and M. Khayat. An end-case analyzer of arabic sentences. *King Saud University Journal (Computer Division)*, 8(1), 1995. Riyadh, Saudi Arabia.

R. Al-Shalabi. *Design and Implementation of an Arabic Morphologyical System to Support Nautral Language Processing.* PhD thesis, Computer Science Department, Illinois Institute of Technology, Chicago, IL, USA, 1996.

H. Al-Shishiny. A formal description of the arabic verbal sentence in definite clause grammar terms. 8, Holland, 1995. Nijmegen University, Institute for the Languages and Cultures of the Middle East.

A. Al-Uthman. A morphological analyzer for arabic. Master's thesis, Computer Science and Engineering Dept., King Fahd University of Petroleum and Minerals, 1989.

J. Allen. *Natural Language Understanding.* The Benjamin/Cumming Publishing Company, Inc., 1995.

A. Alneami. *Design and Implementation of an English to Arabic Machine Translation (MEANA MT).* PhD thesis, Department of Computer Science, University of Sheffield, 1997.

A. Alneami and J. McGregor. Morphological auto-analyser and generator for arabic nouns. In E. Dittres, editor, *Processing Report Nijmegen*, 8, Holland, 1995. Nijmegen University, Institute for the Languages and Cultures of the Middle East.

H. Alshawi, D. M. Carter, J. Eijck, and R. C. Moore. Overview of the core language engine. Technical report, SRI International Cambridge Computer Science Research Centre, University of Cambridge, 1988.

H. Alshawi, B. Gamback, and M. Rayner. Translation by quasi logical form transfer. In *Proceedings of the 29th Annual Meeting of the ACL*, 1991.

K. Amaireh. في التحليل اللّغوي: منهج وصفي تحليلي *fy Alt.hlyl Alll.gwy: mnhj w.sfy t.hlyly.* مكتبة المنَار mktbT AlmnAr, Jourdn, 1987.

I. Androutsopoulos. Masque/sql a natural language front-end for relational databases - usr's manual. Technical report, Department of Artificial Intelligence, University of Edinburgh, 1992.

I. Androutsopoulos, G. D. Ritchie, and P. Thanisch. Natural language interface to database- an introduction. *Natural Language Engineering*, 1(1):29–82, 1995.

Murtadha Bakir. *Aspects of Clause Structure in Arabic - A Study in Word Order Variation in Literary Arabic.* PhD thesis, Indiana University, IN, USA, 1980.

L. Balkan, D. Arnold, and F. Fouvry. Test suits for evaluation in natural language engineering. In *Language Engineering Convention*, London, 1995.

F. Barros. *A Treatment of Anaphora in Portable Natural language Front Ends to Data Bases.* PhD thesis, University of Essex, 1995.

F. Barros and A. DeRoeck. Portable natural language front ends - a review. Research report csm-194, Dept. of Computer Science, University of Essex, 1993.

A. Beeston. *The Arabic Language Today.* Hutchinson Ltd., 1970.

P. Bennett. *A course in Generalized Phrase Structure Grammar.* London: UCL Press, 1995.

K. Bessley. Finite-state description of arabic morphology. In *the Second Conference on bilingual Computing in arabic and english, Literary and Linguistic Computing*, cambridge University, 1990.

K. Bessley. Arabic finite-state morphological analysis and generation. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, pages 89–94, Denmark, 1996.

K. Bessley. Arabic morphological analysis on the internet. In *the 6th International Conference and Exhibition on Multi-lingual computing , ICEMCO-98*, pages 3.1.1–3.1.10. Cambridge University, 1998.

K. Bessley, T. Buckwalter, and S. Newton. Two-level finite-state analysis of arabic morphology. In *the First Conference on bilingual Computing in arabic and english, Literary and Linguistic Computing*, cambridge University, 1989.

J. Binot. Natural language processing and logic. In A. Thayse, editor, *From Natural Language Processing to Logic for Expert Systems : A Logic Based Approach to Artificial Intelligence.* John Wiley and Sons Ltd., England, 1991.

B. Boguraev, R. Garigliano, and J. Tait. Editorial. *Natural Language Engineering*, 1(1), 1995.

A. Burton. *Sublanguage of English for database query in a managerial environment.* PhD thesis, Sunderland Polytechnic,UK, 1991.

N. Chomsky. *Aspects of the Theory of Syntax.* MIT Press, USA, 1965.

E. Codd. A relational model for large share data banks. *Communications of the ACM*, 13(6):53–70, 1970.

P. Cohen. The role of natural language in a multimodel interface. Technical note 514, Computer Dialogue Laboratory, SRI International, 1991.

A. Copestake and K. Jones. Natural language interface to database. *The Knowledge Engineering Review*, 5(5), 1990.

D. Dahl. Pundit – natural language interfaces. In J. Siekmann, editor, *Lecture Notes in Artificial Intelligence*, 636. Springer-Verlag, 1993.

A. DeRoeck, C. Fox, R. Lowden, R. Turner, and B. Walls. A natural language system based on formal semantics. In *Proceedings of International Conference on Current Issues in Computational Linguistics*, pages 268–281. University of Sayns, 1991.

E. Ditters. *A Formal Approach to Arabic Syntax: the Noun Phrase and the Verb Phrase.* PhD thesis, Nijmegen University, Nijmegen: Luxor, Netherlands, 1992.

D. Dowty, R. Wall, and S. Peters. *Introduction to Montague semantics.* D.Reidel Publishing Company, 1981.

A. El-Dessouki, A. El-Dessouki, Nazif, and M. Ahmed. An expert system for understanding arabic sentences. In *The 10th National Conderence*, Jeddah, Saudai Arabia, 1988.

A. El-Dessouki, A. El-Dessouki, Nazif, and M. Ahmed. An atn approach for understanding arabic sentences. In *The 11th National Conderence*, Dhahran, Saudai Arabia, 1989.

S. El Kareh, P. Cote, and M. Maamouri. Development of computer tools to analyze arabic questions. In *Proceedings of The 12th National Conference*, Riyadh, Saudi Arabia, 1990.

T. El-Sadany and M. Hashish. An arabic morphological system. *IBM Systems Journal*, 28(4):600–601, 1989.

A. Fassi Fehri. A realistic syntax of arabic. In Raymond Descout, editor, *Applied Arabic Linguistics, and Signal and information processing.* Hemisphere Publishing Corporation, 1987.

A. Fassi Fehri. Agreement in arabic, binding and coherence. In *Agreement in Natural Language.* Center for the Study of Language and Information, Stanford University, 1988.

C. Fillmore. The case for case. In E. Bach and T. Harms, editors, *Universals in Linguistic Theory.* 1968.

C. Fox. Squirrel documentation. Technical report, Department of Computer Science, University of Essex, 1995.

B. Gamback and M. Rayner. The swedish core language engine. Technical report R92:13, SRI International, Cambridge Computer Science Research Center, 1992.

G. Gazdar. Unbounded dependencies and coordination structure. *Linguistic Inquiry*, 12:155–184, 1981.

G. Gazdar, E. Klein, G. Pullum, and I. Sag. *Generalized phrase structure grammar.* Oxford: Blackwell, 1985.

B. Grosz, D. Appelt, P. Martin, and F. Pereira. Team: An experiment in the design of transportable natural-language interfaces. *Artificial Intelligence, An international Journal*, 32(2):173–241, 1987.

L. Harris. Robot: A high performance natural language interface for data base query. In Leonard Bolc, editor, *Naltural Language Based Computer Systems*, pages 285–318. The Macmillan Press Ltd., 1980.

H. Hassan. *GQuery- A Natural Language Query System for Geographical Databases.* PhD thesis, School of Engineering and Applied Sciences, University of Sussex, 1988.

A. Hassn. النّحو الوَافي *Alnn.hw AlwAfy.* دَار المعَارف, Egypt, 1975.

E. Hendrix, G. G.and Sacerdoti, D. Sagalowicz, and J. Slocum. Developing a natural language interface to complex data. *ACM Trans. Database System*, 3(2):105–147, 1978.

Y. Hilal. Arabic morphological generation. In *Proceedings of the 9th National Computer Conference*, Riyadh, 1986.

P. Humphrey. Natural language processing at eds. *EDS Technical Journal*, 1992.

M. Jarke, J. Turner, E. Stohr, Y. Vassiliou, N. White, and K. Michielsen. A field evaluation of natural language for data retrieval. *IEEE Transactions on Software Engineering*, SE-11(1):97–113, 1985.

T. Johnson. *Natural Language Compututing: the Commercial Applications.* London: OVUM., 1985.

J. Kaplan. Cooperative responses from a portable natural language database query system. In Michael Brady and Robert Berwick, editors, *Computational Models of Discourse.* The Massachusetts Institute of Technology, 1983.

M. Kaplan and J. Bresnan. Lexical functional grammar: a formal system for grammatical representation. In J. Bresnan, editor, *The Mental Representation of Grammatical Relations.* The Massachusetts Institute of Technology, 1982.

M. Khayat. Directions in natural arabic understanding. In *Proceedings of Second Conference on Arabic Computational Linguistics*, pages 472–488, Kuwait, 1989.

M. Khayat. Understanding natural arabic. In *The First KFUPM Workshop on Information and Computer Science: Machine Translation*, pages k1–k4, King Fahd University of Petroleum and Minerals, Saudi Arabia, 1996.

M. Khayat and Al-Muhtaseb. Knowledge representation in natural language system. In *Proceedings of The 10th National Conference*, pages 667–675, Jeddah, Saudi Arabia, 1988.

K. Koskenniemi. Two-level model for morphological analysis. In *Proceedings of IJCAI*, pages 683–685, 1983.

G. Lopes. Transforming english interfaces to other natural languages: An experiment with portuguese. In *Proceedings of the 22th Annual Meeting of the ACL*, pages 8–10, 1984.

B. G. Lowden, B. Wall, A. De Roeck, and C. Fox. An approach to paraphrasing logical query languages in english. *Database Technology*, 4(4):227–233, 1991-1992a.

B. G. Lowden, B. Wall, A. De Roeck, and C. Fox. Modal reasoning in relational systems. *Database Technology*, 4(4):236–244, 1991-1992b.

P. Martin, D. Appelt, and F. Pereira. Transportability and generality in a natural-language interface system. In B. Grosz, K. Jones, and B. Webber, editors, *Readings in Natural Language Processing*. Morgan Kaufmann Publishers, Inc. USA, 1983.

P. Mc Kevitt, D. Partridge, and Y. Wilks. Why machines should analyse intention in natural language dialogue. Technical report, Departement of Computer Science, University of Exter, 1992.

M. Md Sap and McGregor. Natural language interface to databases: State of the art. Technical report, Departement of Computer Science, University of Strathclyde, Glasgow, 1992.

S. Mehdi. Arabic language parser. *Man-Machine Studies*, (25):593–611, 1986.

S. Mehdi. *Computer Interpretation of Arabic*. PhD thesis, University of Exeter, 1987.

F. Mohammad, K. Nasser, and H. Harb. A knowledge based arabic question answering system (aqas). *Aquarterly Publication of the Association for Computing Machirey (ACM), Special interest group on Artificial Intlligence, Sigart Bulletin*, 4(4), 1993.

K. Morick. Customer requirements for natural language system: Results of an inquiry. *IJMMS*, 21:401–414, 1984.

C. Morris. Foundations in a theory of signs. In R. Carnap, editor, *Encyclopedia of Unified Science*. The University Chicago press, 1938.

A. Narayanan and L. Hashem. A three-level finite-state model for arabic morphology. In *the 5th International Conference and Exhibition on Multi-lingual computing , ICEMCO-92*, 1992.

C. Pereira and H. Warren. Definite cluase grammars for language analysis-a survey of the formalism and comparison with augmented transition networks. *Artificial Intelligence*, 13:231–278, 1980.

S. Pulman, H. Alshawi, and D. Carter. Clare: A combined language and reasoning engine. In *JFIT conference*, 1993.

M. Rayner, D. Carter, and P. Bouillon. Adapting the core language engine to french and spanish. Technical Report Crc061, SRI International, Cambridge Computer Science Research Center, 1996.

P. Roochnik. Arabatn: A morpho-syntactic parser for arabic. In *the First Conference on bilingual Computing in arabic and english, Literary and Linguistic Computing*, cambridge University, 1989.

G. Saad. Transitivity causation and passivization a semantic-syntactic study of the verb in classical arabic. In *Monograph no 4*, London: Kengan Paul Intrnational, 1982.

P. Sells. *Lecture on Contemporary Syntactic Theories.* Center for the Study of Language and Information, Stanford University, 1986.

S. Shapiro, editor. *Encyclopedia of Artificial Intellience*, volume I, II. John Wiley and Sons, Inc., New York, 1992.

S. Shieber. *An Introduction to Unification-Based Approaches to Grammar.* Center for the Study of Language and Information, Stanford University, 1986.

M. Smith, R. Ganigliano, R. Morgan, S. Shiu, and S. Jaruis. Lolita: A natural language engineered system. Technical report, Laboratory for Natural Language Engineering, Dept. of Computer Science, University of Durham, 1994.

J. Snow. *A Grammar of Modern Written Arabic Clauses.* PhD thesis, University of Michigan, MI, USA, 1965.

K. Sparck Jones and J. Galliers. *Evaluating Natural Language Processing Systems: An Analysis and Review.* Springer-Verlag, Berlin, 1996.

F. Steurs. Generalized phrase structure grammar. In F. Droste and J. Joseph, editors, *Linguistic Theory and Grammatical Description.* John Benjamins Publishing Company, USA, 1991.

M. Templeton and J. Burger. Problems in natural language interface to dbms with examples from eufid. In *the 1st Conference on Applied Natural Language Processing*, California, 1983.

R. Turner. Properties, propositions and semantic theory. In *Proceedings of the workshop on computational linguistics and formal semantics*, 1988.

R. Turner. Properties, propositions and semantic theory. In R. Michael and J. Roderick, editors, *Computational linguistics and formal semantics.* Cambridge University press, 1992.

M. Wallace. *Communicating with Databases in Natural Language.* Ellis Horwood Limited, England, 1984.

D. Warren and F. Pereira. An efficient easily adaptable system for interpreting natural language queries. *Computational Linguistics*, 8, 1982.

S. Whittaker and M. Walker. Comparing two user-oriented databases query languages: A field study. Technical report hpl-isc-89-060, Hewlett Packard Laboratories, Bristol, 1989.

T. Winograd. *Language As A Cognitive Process.* Addison-Wesley, London, 1983.

W. Woods. Procedural semantics for a question-answering machine. In *Proceedings of the Fall Joint Computer Conference*, pages 457–471, New York, NY, AFIPS, 1968.

W. Woods. Progress in nlu - an application to lunar geology. In *AFIPS Conference Proceedings*, 1973.

W. Wu and D. Dilts. Integrating diverse cim data bases: The role of natural language interface. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(6):1331–1347, 1992.

A. Yamani and A. Al-Zobaidie. Long-distance dependecies and coordination phenomena in arabic interrogatives: an lfg trearment. In *Proceedings of the Second International Conference on Artificial Intelligence Applications*, Cairo, Egypt, 1994.

A. Yamani and A. Al-Zobaidie. Computational linguistics approach to question-answering system for arabic. In *Proceedings of the 5th International Conference and Exhibition on Multi-lingual computing, ICEMCO-96*, Cambridge University, 1996.

H. Yusuf. *Word Order Variation in Arabic: A Generalized Phrase Structure Grammar Analysis.* PhD thesis, School of Oriental and African Studies, University of London, 1983.

F. Ziadeh and R. Winder. *An Introduction to Modern Arabic.* Printceton University Press, 1957.

Appendix A

The System's Arabic Characters

The Arabic alphabet with the Latin equivalent used in the system is listed in the following table.

Arabic Letter	System's character	Name of Letter
أ	a	Alif
ب	b	Baa
ت	t	Taa
ث	v	Thaa
ج	j	Jeem
ح	e	Hha
خ	x	Kha
د	d	Dal
ذ	u	Thal

Continue next page

Arabic Letter	System's character	Name of Letter
ر	r	Raa
ز	z	Zain
س	c	Seen
ش	o	Sheen
ص	s	Saad
ض	p	Dhad
ط	i	Tta
ظ	q	Zha
ع	3	Ain
غ	9	Ghain
ف	f	Faa
ق	g	Qaf
ك	k	Kaf
ل	l	Lam
م	m	Meem
ن	n	Noon
ه	h	Haa
و	w	Waw
ي	y	Yaa

Appendix B

Sample of Queries

The following are a sample of queries which can be processed by the system.

Query: أَ رسب أَحمد في بيسك ام بَسكَال؟

'a rsb 'a.hmd fy bysk am baskAl?

Query: أَ لم يرسب علي؟

'a lm yrsb 'ly?

Query: هل بندر نَاجح؟

hl bndr nAĝ.h?

Query: من يدرّس مَادة بَسكَال في قسم الحَاسب؟

mn ydrrs mAdT baskAl fy qsm al.hAsb?

Query: مَا الموَاد الّتي لم يدرسهَا بدر؟

mA almwAd allty lm ydrshA bdr?

Query: مَا هي درجَات مَادة قوَاعد البيَانَات؟

mA hy drĝAt mAdT qwA'd AlbyAnAt?

Query: من في قسم الحَاسب الآلي؟

mn fy qsm Al.hAsb AlAly?

Query: كم طَالب في مَادة الاَتصَالات؟

km .tAlb fy mAdT AlAt.sAlAt?

Query: مَاذَا يدرس الطَالب محمّد؟

mA_dA ydrs Al.tAlb m.hmmd?

Query: من درّسوَا مَادة نظم المعلومَات؟

mn drrswA mAdT n.zm Alm'lwmAt?

Query: أَ نجح علي في الي٤٥٣ و في الي٧٨٣؟

'a nĝ.h 'ly fy Aly354 w fy Aly387?

Query: هل رسب علي و محمود؟

hl rsb 'ly w m.hmwd?

Query: متَى تخرّج أَحمد؟

mt_A t_hrrĝ 'a.hmd?

Query: مَاذَا يدرّس عبدَاللّطيف؟

mA_dA ydrrs 'bdAl-l.tyf?

Query: في أَي مَادة الطَالب محمود؟

fy 'ay mAdT al.tAlb m.hmwd?

Query: من الّذي درّس مَادة مقدمة الحَاسب؟

mn all_dy drrs mAdT mqdmT Al.hAsb?

Query: يدرّس بدر في أَي قسم ؟

ydrs bdr fy 'ay qsm?

Query: مَا اسمَاء كل المدرّسين؟

mA asmA' kl Almdrrsyn?

Query: متَى درّس الاَستَاذ عبدلعزيز مَادة كوبول؟

mt_A drrs AlAstA_d 'bdl'zyz mAdT kwbwl?

Query: كم عد د الطلَاب في آلي٩٩٣؟

km 'd d Al.tlAb fy 'Aly399?

Query: أَعطني أَرقَام الطلَاب.

'a‘.tny 'arqAm Al.tlAb.

Query: قَاءَمة بأَسمَاء الطلَاب.

qA'ymT b'asmA' Al.tlAb.

Query: من استعَار كتَاب الذكَاء الصنَاعي؟

mn Ast‘Ar ktAb Al_dkA' Al.snA‘y?

Query: أَين موقع كتَاب برولوق؟

'ayn mwq‘ ktAb brwlwq?

Query: متي استعَار بدر كتَاب قوَاعد البينَات؟

mty Ast‘Ar bdr ktAb qwA‘d AlbynAt?

Query: كم كتَاب عند محمّد؟

km ktAb ‘nd m.hmmd?

Query: مَا الكتَاب الّذي استعَاره بدر؟

mA AlktAb All_dy Ast‘Arh bdr?

Query: هل اعَاد احمد كتَاب بَاسكَال؟

hl A‘Ad A.hmd ktAb bAskAl?

Query: من حجز كتَاب بيسك؟

mn .hĝz ktAb bysk?

Appendix C

Examples

C.1 Generalised Quantifiers

```
ARABIC: ma hy drjat al ilab fy gcm computer?
        What it grades the students in department computer?

Morph: [ma,hy,drjat,al,ilab,fy,gcm,computer]
FOL: exists(x,ilab(x,'mf)&drjat(x,fvi2))&exists(a,gcm(a,'computer)
     &fy(fvi2,a))
URC:{fvi2;exists(x,[*,x,*,*,*,*,*,*,*]:student&[x,*,*,*,fvi2]:grade)
    &exists(a,([*,*,*,*,*,*,'computer,*,*]:student&[*,*,*,*,*,*,a,
    *,*]:student&a='computer)&exists(bb,[bb,*,*,*,fvi2]:grade&
    [*,bb,*,*,*,*,a,*,*]:student))}
DRC:{fvi2:grade![grade];exists(x:student![stnum],[*,x,*,*,*,*,*,*,*]
    :student&[x,*,*,*,fvi2]:grade)&exists(a:student![dept],
    ([*,*,*,*,*,*,'computer,*,*]:student&[*,*,*,*,*,*,a,*,*]:
    student&a='computer)&exists(bb:student![stnum],[bb,*,*,*,fvi2]
    :grade&[*,bb,*,*,*,*,a,*,*]:student))}
OP1:{fvi2:grade![grade];exists(skf3:student![stnum],exists(skf2:
```

```
    student![dept],exists(skf1:student![stnum],[skf1,*,*,*,fvi2]:
    grade&[*,*,*,*,*,*,'computer,*,*]:student&skf2='computer&
    [skf3,*,*,*,fvi2]:grade&[*,skf3,*,*,*,*,skf2,*,*]:student)))}
TRC:{fvi2!grade;fvi2:grade;exists(skf3:student,exists(skf2:student,
    exists(skf1:student,exists(tuple1:grade,tuple1!stnum=skf1
    !stnum&tuple1!grade=fvi2!grade)&exists(tuple2:student,tuple2
    !dept=computer)&skf2!dept=computer&exists(tuple3:grade,tuple3
    !stnum=skf3!stnum&tuple3!grade=fvi2!grade)&exists(tuple4:
    student,tuple4!stnum=skf3!stnum&tuple4!dept=skf2!dept))))}
OP2:{fvi2!grade;fvi2:grade;exists(sk3:student,exists(sk4:grade,
    exists(sk5:student,exists(sk6:grade,exists(sk7:student,
    sk4!stnum=sk3!stnum&sk4!grade=fvi2!grade&sk5!dept=computer
    &sk6!grade=fvi2!grade&sk7!stnum=sk6!stnum&sk7
    !dept=computer)))))}
OP3:{fvi2!grade;fvi2:grade;exists(sk3:student,exists(sk4:grade,
    exists(sk5:student,exists(sk6:grade,exists(sk7:student,sk4
    !stnum=sk3!stnum&sk4!grade=fvi2!grade&sk5!dept=computer&sk6
    !grade=fvi2!grade&sk7!stnum=sk6!stnum&sk7!dept=computer)))))}
SQL:SELECT DISTINCT fvi2.grade FROM grade fvi2 WHERE EXISTS(
    SELECT * FROM student sk3 WHERE EXISTS(SELECT * FROM grade
    sk4 WHERE EXISTS(SELECT * FROM student sk5 WHERE EXISTS
    (SELECT * FROM grade sk6 WHERE EXISTS(SELECT * FROM
    student sk7 WHERE (sk4.stnum=sk3.stnum AND (sk4.grade=
    fvi2.grade AND (sk5.dept='computer' AND (sk6.grade=
    fvi2.grade AND (sk7.stnum=sk6.stnum AND sk7.dept
    ='computer'))))))))));

ARABIC: kl ialb drc madt prolog.
every student studied cource prolog.

Morph: [kl,ialb,drc,madt,prolog]
FOL: all(x,ialb(x,'masc)=>exists(a,madt(a,'prolog)&drc(x,a)))
URC:all(x,[*,x,'masc,*,*,*,*,*,*]:student=>exists(a,['prolog,
```

```
    a,*,*,*]:course&[x,a,*,*,*]:grade))
DRC:all(x:student![stnum],[*,x,'masc,*,*,*,*,*,*]:student=>exists
    (a:course![corsnum],['prolog,a,*,*,*]:course&[x,a,*,*,*]:grade))
OP1:all(x4:student![stnum],exists(skf1:course![corsnum],(
    [*,x4,'masc,*,*,*,*,*,*]:student=>['prolog,skf1,*,*,*]:course)
    &([*,x4,'masc,*,*,*,*,*,*]:student=>[x4,skf1,*,*,*]:grade)))
TRC:all(x4:student,exists(skf1:course,(exists(tuple1:student,
    tuple1!stnum=x4!stnum&tuple1!gender=masc)=>exists(tuple2:course,
    tuple2!title=prolog&tuple2!corsnum=skf1!corsnum))&(exists(tuple3:
    student,tuple3!stnum=x4!stnum&tuple3!gender=masc)=>exists(tuple4:
    grade,tuple4!stnum=x4!stnum&tuple4!corsnum))))
OP2:all(x4:student,exists(sk2:course,exists(sk3:grade,all(tuple3:
    student,all(tuple1:student,(tuple1!stnum=x4!stnum&tuple1!gender=
    masc=>sk2!title=prolog)&(tuple3!stnum=x4!stnum&tuple3!gender=
    masc=>sk3!stnum=x4!stnum)&(tuple1!stnum=x4!stnum&tuple1!gender=
    masc&tuple3!stnum=x4!stnum&tuple3!gender=masc=>sk3!corsnum=
    sk2!corsnum)))))
OP3:all(x4:student,exists(sk2:course,exists(sk3:grade,all(tuple3:
    student,all(tuple1:student,(tuple1!stnum=x4!stnum&tuple1!gender=
    masc=>sk2!title=prolog)&(tuple3!stnum=x4!stnum&tuple3!gender=
    masc=>sk3!stnum=x4!stnum)&(tuple1!stnum=x4!stnum&tuple1!gender=
    masc&tuple3!stnum=x4!stnum&tuple3!gender=masc=>sk3!corsnum=
    sk2!corsnum)))))
SQL: SELECT DISTINCT a.answer FROM answer a WHERE 1=ALL(SELECT 0
    FROM student x4 WHERE NOT EXISTS(SELECT * FROM course sk2 WHERE
    EXISTS(SELECT * FROM grade sk3 WHERE 1=ALL(SELECT 0 FROM student
    tuple3 WHERE NOT 1=ALL(SELECT 0 FROM student tuple1 WHERE NOT
    ((NOT (tuple1.stnum=x4.stnum AND tuple1.gender='masc') OR
    sk2.title='prolog') AND ((NOT (tuple3.stnum=x4.stnum AND
    tuple3.gender='masc') OR sk3.stnum=x4.stnum) AND (NOT
    (tuple1.stnum=x4.stnum AND (tuple1.gender='masc' AND
    (tuple3.stnum=x4.stnum AND tuple3.gender='masc'))) OR
    sk3.corsnum=sk2.corsnum)))))));
```

C.2 Conjunction

```
ARABIC: mn alilab w alialbat fy madt prolog?
        who the students male and the students female in course prolog?
Morph: [mn,al,ilab,w,al,ialbat,fy,madt,prolog]
FOL:(exists(x,ilab(x,‘mf)&be(x,fva2))&exists(x,ialbat(x,‘fem)&
    be(x,fva2)))&exists(a,madt(a,‘prolog)&fy(fva2,a))
URC:{fva2;(exists(x,[*,x,*,*,*,*,*,*,*]:student&x=fva2)&exists(x,
    [*,x,‘fem,*,*,*,*,*,*]:student&x=fva2))&exists(a,[‘prolog,a,*,*,*]:
    course&[fva2,a,*,*,*]:grade)}
DRC:{fva2:student![stnum];(exists(x:student![stnum],[*,x,*,*,*,*,*,*,*]:
    student&x=fva2)&exists(x:student![stnum],[*,x,‘fem,*,*,*,*,*,*]:
    student&x=fva2))&exists(a:course![corsnum],[‘prolog,a,*,*,*]:
    course&[fva2,a,*,*,*]:grade)}
OP1:{fva2:student![stnum];exists(skf3:course![corsnum],exists(skf2:
    student![stnum],exists(skf1:student![stnum],skf1=fva2&[*,skf2,
    ‘fem,*,*,*,*,*,*]:student&skf2=fva2&[‘prolog,skf3,*,*,*]:course&
    [fva2,skf3,*,*,*]:grade)))}
TRC:{fva2!stnum;fva2:student;exists(skf3:course,exists(skf2:student,
    exists(skf1:student,skf1!stnum=fva2!stnum&exists(tuple1:student,
    tuple1!stnum=skf2!stnum&tuple1!gender=fem)&skf2!stnum=fva2!stnum&
    exists(tuple2:course,tuple2!title=prolog&tuple2!corsnum=skf3!corsnum)
    &exists(tuple3:grade,tuple3!stnum=fva2!stnum&tuple3!corsnum=skf3!
    corsnum))))}
OP2:{fva2!stnum;fva2:student;exists(sk3:student,exists(sk4:student,
    exists(sk5:course,exists(sk6:grade,sk3!stnum=fva2!stnum&sk4!
    gender=fem&sk5!title=prolog&sk6!stnum=fva2!stnum&sk6!corsnum=sk5!
    corsnum&sk4!stnum=fva2!stnum))))}
OP3:{fva2!stnum;fva2:student;exists(sk5:course,exists(sk6:grade,fva2!gender
    =fem&sk5!title=prolog&sk6!stnum=fva2!stnum&sk6!corsnum=sk5!corsnum))}
SQL: SELECT DISTINCT fva2.stnum FROM student fva2 WHERE EXISTS(SELECT *
```

```
FROM course sk5 WHERE EXISTS(SELECT * FROM grade sk6 WHERE
(fva2.gender='fem' AND (sk5.title='prolog' AND (sk6.stnum=fva2.stnum
AND sk6.corsnum=sk5.corsnum)))));
```

C.3 Negation

```
ARABIC: ma almwad allty lm ydrcha Badr?
        what the course that not study it Badr?

Morph: [ma,al,mwad,allty,lm,ydrc,ha,Badr]
FOL: exists(x,(madt(x)& ~ydrc(x,'badr))&be(x,fvi1))
URC: {fvi1;exists(x,([*,x,*,*,*]:course& ~exists(bb,[bb,x,*,*,*]:
     grade&['badr,bb,*,*,*,*,*,*,*]:student))&x=fvi1)}
DRC: {fvi1:course![corsnum];exists(x:course![corsnum],([*,x,*,*,*]:course&
     ~exists(bb:student![stnum],[bb,x,*,*,*]:grade&['badr,bb,*,*,
      *,*,*,*,*]:student))&x=fvi1)}
 OP1:{fvi1:course![corsnum];exists(skf1:course![corsnum],all(x10:
     student![stnum],~ ([x10,skf1,*,*,*]:grade&['badr,x10,*,*,*,*,*,*,*]:
     student)&skf1=fvi1))}
 TRC:{fvi1!corsnum;fvi1:course;exists(skf1:course,all(x10:student,
     ~(exists(tuple1:grade,tuple1!stnum=x10!stnum&tuple1!corsnum=skf1!
     corsnum)&exists(tuple2:student,tuple2!stname=badr&tuple2!stnum=
     x10!stnum))&skf1!corsnum=fvi1!corsnum))}
 OP2:{fvi1!corsnum;fvi1:course;all(tuple1:grade,all(tuple2:student,
     all(x10:student,~(tuple1!corsnum=fvi1!corsnum&tuple1!stnum=x10!
     stnum&tuple2!stname=badr&tuple2!stnum=x10!stnum))))}
 OP3:{fvi1!corsnum;fvi1:course;all(tuple1:grade,all(tuple2:student,
     all(x10:student,~(tuple1!corsnum=fvi1!corsnum&tuple1!stnum=x10!
     stnum&tuple2!stname=badr&tuple2!stnum=x10!stnum))))}
 SQL: SELECT DISTINCT fvi1.corsnum FROM course fvi1 WHERE 1=ALL
      (SELECT 0 FROM grade tuple1 WHERE NOT 1=ALL(SELECT 0 FROM student
```

```
tuple2 WHERE NOT 1=ALL(SELECT 0 FROM student x10 WHERE NOT NOT
(tuple1.corsnum=fvi1.corsnum AND (tuple1.stnum=x10.stnum AND
(tuple2.stname='badr' AND tuple2.stnum=x10.stnum)))));
```

C.4 Noun/noun Modification

```
ARABIC: mn hw mrod alialbt rgm 9600234?
        who he advise the student number 9600234?

Morph: [mn,hw,mrod,al,ialbt,rgm,9600234]
FOL: exists(x,rgm(x,‘9600234,‘ialb: ‘fem)&mrod(x,fva1,‘masc))
URC:{fva1;exists(x,[x,‘9600234,‘fem,*,*,*,*,*,*]:student&exists(bb,
    [x,*,*,*,*,*,*,bb,*]:student&[fva1,bb,‘masc,*,*,*,*,*]:faculty))}
DRC:{fva1:faculty![facname];exists(x:student![stname],[x,‘9600234,
    ‘fem,*,*,*,*,*,*]:student&exists(bb:faculty![facnum],[x,*,*,*,
    *,*,*,bb,*]:student&[fva1,bb,‘masc,*,*,*,*,*]:faculty))}
 OP1:{fva1:faculty![facname];exists(skf2:faculty![facnum],exists(skf1:
    student![stname],[skf1,‘9600234,‘fem,*,*,*,*,*,*]:student&[skf1,*,*,*,
    *,*,*,skf2,*]:student&[fva1,skf2,‘masc,*,*,*,*,*]:faculty))}
 TRC:{fva1!facname;fva1:faculty;exists(skf2:faculty,exists(skf1:student,
    exists(tuple1:student,tuple1!stname=skf1!stname&tuple1!stnum=9600234&
    tuple1!gender=fem)&exists(tuple2:student,tuple2!stname=skf1!stname&
    tuple2!advisor=skf2!facnum)&exists(tuple3:faculty,tuple3!facname=
    fva1!facname&tuple3!facnum=skf2!facnum&tuple3!gender=masc)))}
 OP2:{fva1!facname;fva1:faculty;exists(sk3:student,exists(sk5:faculty,
    exists(sk4:student,sk3!stnum=9600234&sk3!gender=fem&sk5!facname=fva1!
    facname&sk5!gender=masc&sk5!facnum=sk4!advisor&sk4!stname=sk3!stname)))}
 OP3:{fva1!facname;fva1:faculty;exists(sk3:student,exists(sk5:faculty,
    exists(sk4:student,sk3!stnum=9600234&sk3!gender=fem&sk5!facname=fva1!
    facname&sk5!gender=masc&sk5!facnum=sk4!advisor&sk4!stname=sk3!stname)))}
SQL: SELECT DISTINCT fva1.facname FROM faculty fva1 WHERE EXISTS(SELECT *
```

```
FROM student sk3 WHERE EXISTS(SELECT * FROM faculty sk5 WHERE
EXISTS(SELECT * FROM student sk4 WHERE (sk3.stnum=9600234 AND
(sk3.gender='fem' AND (sk5.facname=fva1.facname AND (sk5.gender='masc'
AND (sk5.facnum=sk4.advisor AND sk4.stname=sk3.stname)))))));
```

C.5 Relative Clause

```
ARABIC: mn alilab alluyn rcbwa fy prolog?
        who the students that failed in Prolog?

Morph: [mn,al,ilab,alluyn,rcb,wa,fy,prolog]
FOL: exists(x,(ilab(x,‘mf)&fy(x,‘rcb,‘prolog))&be(x,fva1))
URC:{fva1;exists(x,([*,x,*,*,*,*,*,*,*]:student&exists(bb,[x,bb,*,*,‘f]:
    grade&[‘prolog,bb,*,*,*]:course))&x=fva1)}
DRC:{fva1:student![stnum];exists(x:student![stnum],([*,x,*,*,*,*,*,*,*]:
    student&exists(bb:course![corsnum],[x,bb,*,*,‘f]:grade&[‘prolog,bb,*,
    *,*]:course))&x=fva1)}
OP1:{fva1:student![stnum];exists(skf2:course![corsnum],exists(skf1:student!
    [stnum],[skf1,skf2,*,*,‘f]:grade&[‘prolog,skf2,*,*,*]:course&skf1=fva1))}
TRC:{fva1!stnum;fva1:student;exists(skf2:course,exists(skf1:student,
    exists(tuple1:grade,tuple1!stnum=skf1!stnum&tuple1!corsnum=skf2!corsnum&
    tuple1!grade=f)&exists(tuple2:course,tuple2!title=prolog&tuple2!corsnum=
    skf2!corsnum)&skf1!stnum=fva1!stnum))}
OP2:{fva1!stnum;fva1:student;exists(sk3:grade,exists(sk4:course,sk3!grade=
    f&sk4!title=prolog&sk4!corsnum=sk3!corsnum&sk3!stnum=fva1!stnum))}
OP3:{fva1!stnum;fva1:student;exists(sk3:grade,exists(sk4:course,sk3!grade=
    f&sk4!title=prolog&sk4!corsnum=sk3!corsnum&sk3!stnum=fva1!stnum))}
SQL: SELECT DISTINCT fva1.stnum FROM student fva1 WHERE EXISTS(SELECT *
     FROM grade sk3 WHERE EXISTS(SELECT * FROM course sk4 WHERE
     (sk3.grade='f' AND (sk4.title='prolog' AND (sk4.corsnum=sk3.corsnum
     AND sk3.stnum=fva1.stnum)))));
```

C.6 Prepositional Attachment

```
ARABIC:  mn ydrc fy ay madt?
         who study in which course?

Morph:  [mn,ydrc,fy,ay,madt]
FOL: madt(fvi1)&fy(fva1,'ydrc,fvi1)
URC: {fvi1,fva1;[*,fvi1,*,*,*]:course&[fva1,fvi1,*,*,*]:grade}
DRC:{fvi1:course![corsnum],fva1:student![stnum];[*,fvi1,*,*,*]:course&
     [fva1,fvi1,*,*,*]:grade}
OP1: {fvi1:course![corsnum],fva1:student![stnum];[fva1,fvi1,*,*,*]:grade}
TRC:{fvi1!corsnum,fva1!stnum;fvi1:course,fva1:student;exists(tuple1:
     grade,tuple1!stnum=fva1!stnum&tuple1!corsnum=fvi1!corsnum)}
OP2:{fvi1!corsnum,fva1!stnum;fvi1:course,fva1:student;exists(tuple1:
     grade,tuple1!stnum=fva1!stnum&tuple1!corsnum=fvi1!corsnum)}
OP3:{fvi1!corsnum,fva1!stnum;fvi1:course,fva1:student;exists(sk1:
     grade,sk1!stnum=fva1!stnum&sk1!corsnum=fvi1!corsnum)}
SQL: SELECT DISTINCT fvi1.corsnum, fva1.stnum FROM course fvi1,
     student fva1 WHERE EXISTS(SELECT * FROM grade sk1 WHERE
     (sk1.stnum=fva1.stnum AND sk1.corsnum=fvi1.corsnum));
```

C.7 Yes/no Question

```
ARABIC: a drrc alactau ahmed madt prolog fy 1996?
        is taught the lecturer Ahmed course Prolog in 1996?

Morph:  [a,drrc,al,actau,ahmed,madt,prolog,fy,1996]
FOL: exists(x,actau(x,'ahmed,'masc)&exists(a,madt(a,'prolog)&
       fy(x,'drrc:a,'1996)))
URC: exists(x,['ahmed,x,'masc,*,*,*,*,*]:faculty&exists(a,['prolog,a,*,*,*]:
       course&[a,*,'1996,x]:teach))
```

```
DRC: exists(x:faculty![facnum],['ahmed,x,'masc,*,*,*,*,*]:faculty&exists
       (a:course![corsnum],['prolog,a,*,*,*]:course&[a,*,'1996,x]:teach))
OP1: exists(skf2:course![corsnum],exists(skf1:faculty![facnum],['ahmed,skf1,
      'masc,*,*,*,*,*]:faculty&['prolog,skf2,*,*,*]:course&
       [skf2,*,'1996,skf1]:teach))
TRC: exists(skf2:course,exists(skf1:faculty,exists(tuple1:faculty,
      tuple1!facname=ahmed&tuple1!facnum=skf1!facnum&tuple1!gender=masc)&
      exists(tuple2:course,tuple2!title=prolog&tuple2!corsnum=skf2!corsnum)&
      exists(tuple3:teach,tuple3!corsnum=skf2!corsnum&tuple3!tchyear=1996&
      tuple3!facnum=skf1!facnum)))
OP2: exists(sk3:faculty,exists(sk2:faculty,exists(sk4:course,exists(sk1:
       course,exists(sk5:teach,sk3!facname=ahmed&sk3!facnum=sk2!facnum&
       sk3!gender=masc&sk4!title=prolog&sk4!corsnum=sk1!corsnum&sk5!
       corsnum=sk1!corsnum&sk5!tchyear=1996&sk5!facnum=sk2!facnum))))
OP3: exists(sk3:faculty,exists(sk2:faculty,exists(sk4:course,exists(sk1:
      course,exists(sk5:teach,sk3!facname=ahmed&sk3!facnum=sk2!facnum&sk3!
      gender=masc&sk4!title=prolog&sk4!corsnum=sk1!corsnum&sk5!corsnum=sk1!
       corsnum&sk5!tchyear=1996&sk5!facnum=sk2!facnum))))
SQL: SELECT DISTINCT a.answer FROM answer a WHERE EXISTS(SELECT *
       FROM faculty sk3 WHERE EXISTS(SELECT * FROM faculty sk2 WHERE EXISTS
      (SELECT * FROM course sk4 WHERE EXISTS(SELECT * FROM course sk1 WHERE
      EXISTS(SELECT * FROM teach sk5 WHERE (sk3.facname='ahmed' AND
      (sk3.facnum=sk2.facnum AND (sk3.gender='masc' AND (sk4.title='prolog'
      AND (sk4.corsnum=sk1.corsnum AND (sk5.corsnum=sk1.corsnum AND
      (sk5.tchyear=1996 AND sk5.facnum=sk2.facnum)))))))))));
```

C.8 Count Question

```
ARABIC: km ialb fy madt prolog?
        how many student in course Prolog?
```

```
Morph: [km,ialb,fy,madt,prolog]
FOL: ialb(fvn4,'masc)&exists(a,madt(a,'prolog)&fy(fvn4,a))
URC:{fvn4;[*,fvn4,'masc,*,*,*,*,*,*]:student&exists(a,['prolog,a,*,*,*]:
     course&[fvn4,a,*,*,*]:grade)}
DRC:{fvn4:student![stnum];[*,fvn4,'masc,*,*,*,*,*,*]:student&exists(a:
    course![corsnum],['prolog,a,*,*,*]:course&[fvn4,a,*,*,*]:grade)}
    OP1:{fvn4:student![stnum];exists(skf1:course![corsnum],[*,fvn4,'masc,*,*,*,
     *,*,*]:student&['prolog,skf1,*,*,*]:course&[fvn4,skf1,*,*,*]:grade)}
    TRC:{fvn4!stnum;fvn4:student;exists(skf1:course,exists(tuple1:student,
     tuple1!stnum=fvn4!stnum&tuple1!gender=masc)&exists(tuple2:course,
     tuple2!title=prolog&tuple2!corsnum=skf1!corsnum)&exists(tuple3:grade,
     tuple3!stnum=fvn4!stnum&tuple3!corsnum=skf1!corsnum))}
    OP2:{fvn4!stnum;fvn4:student;exists(sk2:student,exists(sk3:course,
     exists(sk4:grade,sk2!stnum=fvn4!stnum&sk2!gender=masc&sk3!title=
     prolog&sk4!stnum=fvn4!stnum&sk4!corsnum=sk3!corsnum)))}
    OP3:{fvn4!stnum;fvn4:student;exists(sk3:course,exists(sk4:grade,fvn4!
     gender=masc&sk3!title=prolog&sk4!stnum=fvn4!stnum&sk4!corsnum=
     sk3!corsnum))}
    SQL: SELECT COUNT (*)  FROM student fvn4 WHERE EXISTS(SELECT * FROM
         course sk3 WHERE EXISTS(SELECT * FROM grade sk4 WHERE
         (fvn4.gender='masc' AND (sk3.title='prolog' AND
         (sk4.stnum=fvn4.stnum AND sk4.corsnum=sk3.corsnum)))));
```

C.9 Command statment

```
ARABIC: aib3 3nawyn alilab.
        List the students' addresses.

Morph: [aib3,3nawyn,al,ilab]
FOL: exists(y,all(x,ilab(x,'mf)<=>x=y)&3nwan(y,fv1))
URC:{fv1;exists(y,all(x,[*,x,*,*,*,*,*,*,*]:student<=>x=y)&
```

```
     [*,y,*,fv1,*,*,*,*,*]:student)}
DRC: {fv1:student![address];exists(y:student![stnum],all(x:student!
       [stnum],[*,x,*,*,*,*,*,*,*]:student<=>x=y)&
       [*,y,*,fv1,*,*,*,*,*]:student)}
OP1: {fv1:student![address];exists(skf1:student![stnum],all(x22:student!
       [stnum],x22=skf1&[*,skf1,*,fv1,*,*,*,*,*]:student))}
TRC: {fv1!address;fv1:student;exists(skf1:student,all(x22:student,x22!
       stnum=skf1!stnum&exists(tuple1:student,tuple1!stnum=skf1!stnum&
       tuple1!address=fv1!address)))}
OP2: {fv1!address;fv1:student;exists(sk1:student,all(x22:student,
       exists(sk2:student,x22!stnum=sk1!stnum&sk2!stnum=sk1!stnum&sk2!
       address=fv1!address)))}
OP3: {fv1!address;fv1:student;exists(sk1:student,all(x22:student,
       exists(sk2:student,x22!stnum=sk1!stnum&sk2!stnum=sk1!stnum&sk2!
       address=fv1!address)))}
SQL: SELECT DISTINCT fv1.address FROM student fv1 WHERE EXISTS(SELECT *
     FROM student sk1 WHERE 1=ALL(SELECT 0 FROM student x22 WHERE NOT
     EXISTS(SELECT * FROM student sk2 WHERE (x22.stnum=sk1.stnum AND
     (sk2.stnum=sk1.stnum AND sk2.address=fv1.address))))

ARABIC: a3iny acmaa jmy3 alacatut.
        Give me the name of all faculty.

Morph: [a3iny,acmaa,jmy3,al,acatut]
FOL: exists(y,all(x,actau(x,'masc)<=>x=y)&acm(y,fv1))
URC:{fv1;exists(y,all(x,[*,x,'masc,*,*,*,*,*]:faculty<=>x=y)&
     [fv1,y,*,*,*,*,*,*]:faculty)}
DRC: {fv1:faculty![facname];exists(y:faculty![facnum],all(x:faculty!
       [facnum],[*,x,'masc,*,*,*,*,*]:faculty<=>x=y)&[fv1,y,*,*,*,*,*,*]:
       faculty)}
OP1: {fv1:faculty![facname];exists(skf1:faculty![facnum],all(x30:faculty!
     [facnum],(x30=skf1=>[*,x30,'masc,*,*,*,*,*]:faculty)&
     ([*,x30,'masc,*,*,*,*,*]:faculty=>x30=skf1)&
```

```
      [fv1,skf1,*,*,*,*,*,*]:faculty))}
TRC: {fv1!facname;fv1:faculty;exists(skf1:faculty,all(x30:faculty,(x30!
      facnum=skf1!facnum=>exists(tuple1:faculty,tuple1!facnum=x30!facnum&
      tuple1!gender=masc))&(exists(tuple2:faculty,tuple2!facnum=x30!
      facnum&tuple2!gender=masc)=>x30!facnum=skf1!facnum)&exists(tuple3:
      faculty,tuple3!facname=fv1!facname&tuple3!facnum=skf1!facnum)))}
 OP2: {fv1!facname;fv1:faculty;exists(sk1:faculty,all(x30:faculty,exists
      (sk2:faculty,exists(sk3:faculty,all(tuple2:faculty,(x30!facnum=sk1!
      facnum=>sk2!facnum=x30!facnum)&(x30!facnum=sk1!facnum=>sk2!gender=
      masc)&(tuple2!facnum=x30!facnum&tuple2!gender=masc=>x30!facnum=sk1!
      facnum)&sk3!facname=fv1!facname&sk3!facnum=sk1!facnum)))))}
 OP3: {fv1!facname;fv1:faculty;exists(sk1:faculty,all(x30:faculty,exists
      (sk2:faculty,exists(sk3:faculty,all(tuple2:faculty,(x30!facnum=sk1!
      facnum=>sk2!facnum=x30!facnum)&(x30!facnum=sk1!facnum=>sk2!gender=
      masc)&(tuple2!facnum=x30!facnum&tuple2!gender=masc=>x30!facnum=sk1!
      facnum)&sk3!facname=fv1!facname&sk3!facnum=sk1!facnum)))))}
 SQL: SELECT DISTINCT fv1.facname FROM faculty fv1 WHERE EXISTS(SELECT *
      FROM faculty sk1 WHERE 1=ALL(SELECT 0 FROM faculty x30 WHERE NOT
      EXISTS(SELECT * FROM faculty sk2 WHERE EXISTS(SELECT * FROM
      faculty sk3 WHERE 1=ALL(SELECT 0 FROM faculty tuple2 WHERE NOT (
      (NOT x30.facnum=sk1.facnum OR sk2.facnum=x30.facnum) AND ((NOT
       x30.facnum=sk1.facnum OR sk2.gender='masc') AND ((NOT
      (tuple2.facnum=x30.facnum AND tuple2.gender='masc') OR
       x30.facnum=sk1.facnum) AND (sk3.facname=fv1.facname AND
       sk3.facnum=sk1.facnum)))))))))
```