M.Sc. Engg. Thesis

# An HPSG Analysis on Declension of Nominals and Verbs in Arabic Grammar

by
Mahmudul Hasan Masum

Submitted to

Department of Computer Science and Engineering
in partial fulfilment of the requirments for the degree of
Master of Science in Computer Science and Engineering

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology (BUET)
Dhaka 1000

January 2013

The thesis titled "**An HPSG Analysis on Declension of Nominals and Verbs in Arabic Grammar**," submitted by Mahmudul Hasan Masum, Roll No. 100705074P, Session October 2007, to the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, has been accepted as satisfactory in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Engineering and approved as to its style and contents. Examination held on December 22, 2012.

## Board of Examiners

| | |
|---|---|
| 1. ______________________________ | |
| Dr. M. Sohel Rahman | Chairman |
| Associate Professor | (Supervisor) |
| Department of Computer Science and Engineering | |
| BUET, Dhaka 1000 | |
| 2. ______________________________ | |
| Dr. Abu Sayed Md. Latiful Hoque | Member |
| Professor & Head | (Ex-officio) |
| Department of Computer Science and Engineering | |
| BUET, Dhaka 1000 | |
| 3. ______________________________ | |
| Dr. Muhammad Masroor Ali | Member |
| Professor | |
| Department of Computer Science and Engineering | |
| BUET, Dhaka 1000 | |
| 4. ______________________________ | |
| Dr. Md. Monirul Islam | Member |
| Assistant Professor | |
| Department of Computer Science and Engineering | |
| BUET, Dhaka 1000 | |
| 5. ______________________________ | |
| Dr. Mohammad Nurul Huda | Member |
| Associate Professor | (External) |
| Department of Computer Science & Engineering | |
| United International University, Bangladesh | |

# Candidate's Declaration

It is hereby declared that this thesis or any part of it has not been submitted elsewhere for the award of any degree or diploma.

Mahmudul Hasan Masum
Candidate

# Contents

# List of Figures

# List of Tables

# Acknowledgments

*In the name of Allah, most Gracious, most Compassionate.*

I express my heart-felt gratitude to my supervisor, Dr. M. Sohel Rahman for his constant supervision of this work. He helped me a lot in every aspect of this work and guided me with proper directions whenever I sought one. His patient hearing of my ideas, critical analysis of my observations and detecting flaws in my thinking and writing have made this thesis a success.

I would also want to thank the members of my thesis committee for their valuable suggestions. I thank Dr. Abu Sayed Md. Latiful Hoque, Dr. Muhammad Masroor Ali, Dr. Md. Monirul Islam and specially the external member Dr. Mohammad Nurul Huda.

I also thank one of my friend Md. Sadiqul Islam, who was my partner in the early works of my research. We jointly wrote a paper on HPSG formalism of verbal noun and got the paper published in HPSG conference ( [22]), which is the only international conference of HPSG. We have also submitted part of this research work on journal Transactions on Asian Language Information Processing (TALIP). Also parts of this thesis are published in ACIT'2009 ( [23]) and ACIT'2012 ( [31]). Verbal declension part is accepted in ICECE'2012 ( [32]).

I also thank Professor Ivan A. Sag of Stanford University, Prof. Dr. Stefan Müller of Institut ü Deutsche und Niederlndische Philologie and Frank M. Richter of Universität Tbingen for their kind help in HPSG framework application and also in implementation phase.

In this regard, I remain ever grateful to my family, who always exists as sources of inspiration behind every success of mine I have ever made.

# Abstract

Natural language processing (NLP) deals with computational linguistic modeling of large coverage of vocabulary of human languages. Among the living languages, Semitic languages can construct numerous lexemes by demonstrating rich morphology, which is an important branch of NLP. Derivation and inflection, for example, are two such morphological operations by which Semitic languages can generate numerous inflected or derived lexemes respectively. Declension is one kind of inflection to construct one form from another. Among Semitic languages, Arabic is very rich in grammatical declension of nouns and verbs. In classical Arabic, noun lexemes are declined by nine distinct ways and verb lexemes are declined by four ways. Modeling the morphological effect of such a rich declension system is a challenging problem and is essential for intelligent and automated processing of Arabic language. But this declension phenomenon of Arabic nouns and verbs has not been captured yet by computational modeling.

In this thesis, we analyze the declension system of Arabic nouns and verbs and design lexical type hierarchy by which the declension type of any noun or verb lexeme will be determined from the lexical type. We develop an algorithm to determine declension type of a noun lexeme. We also show construction rules to capture the morphological and syntactic effect of declension types dynamically. We also analyze Nominal definiteness and present construction rules to generate definite lexemes from indefinite. We use Head-Driven Phase Structure Grammar (HPSG) which provides a versatile, multidimensional, constraint-based architecture for supporting morphological, syntactic and semantic features of a language. We show implementation of lexical type hierarchy and construction

rules in TRALE, an implementation platform which was developed specially for the grammars of HPSG. We believe our work effectively extends the capabilities of existing HPSG framework for supporting declension of Nominals and verbs in Arabic.

# Chapter 1

# Introduction

Natural language processing (NLP), which is a combination several well established research areas such as computer science, artificial intelligence and linguistics, deals with interaction between human language and computer. It is an enormous research field consisting of several branches, for example, linguistics, artificial intelligence, philosophy etc. There are lots of research tasks in NLP and some of these have significant contributions in several real-world applications, namely speech recognition, text to speech generation, automatic summarization, language translation, natural language interfaces to computer systems, optical character recognition, question answering, e-mail filtering, intelligent search engines and many more. For formalizing NLP, Head-driven Phrase Structure Grammar (HPSG) [40] has a significant position as it integrates all the essential linguistic layers. There are significant reasons behind choosing HPSG formalism of Arabic declension. Section 1.1 describes our motivation for this thesis followed by problem statement (Section 1.2), scope of work (Section 1.3) and major contributions (1.4).

## 1.1 Motivation

Semitic languages exhibit rich morphological operations for construction of lexicons. We can have a large coverage of vocabulary in these languages by computational linguistic modeling of their morphology. In this thesis we focus on Arabic for morphological analysis. It is one of the best instances of morphology among the living languages. Arabic is the mother tongue of more than two hundred and twenty million people and it ranks fourth by number of native speakers ( [29]). Despite these facts, the morphological analysis of Arabic language is relatively a new research field.

Grammatical declension is one kind o morphological operation and resides at the heart of Arabic grammar ( [24]). Grammatical declension is known in Arabic as I'rab. By definition, declension is the process of disambiguating the grammatical roles of words by slightly changing their end vowels. Details about I'rab is discussed in Section 2.2.

For modeling Arabic morphology, we have chosen Head-driven Phrase Structure Grammar (HPSG) which is an attractive tool for capturing complex linguistic constructs. It combines the best ideas from its predecessors: Generalized Phrase Structure Grammar (GPSG) [18], Lexical Functional Grammar (LFG) [8] and Government and Binding theory (GB) [12]. It plays a vital role in NLP because of its flexibility for adding different data types for grammatical description, changing individual components of a grammar and applying the framework to new languages. We have worked on Sign Based Construction Grammar (SBCG) [47] version of HPSG.

## 1.2 Problem Statement

In classical Arabic, there are nine ways to represent nominal declension and based on these there are sixteen classes of nouns. These are addressed from Type 1 to Type 9 ($T_n1$ - $T_n9$). Each of these sixteen classes is not partitioned because some of these noun classes have conditions depending on its placement in a phrase and declension type is applicable based on fulfillment of these conditions. As an example, declension type $T_n9$ will not appear in lexeme level. It is applicable to sound masculine plural if it is used as possession of first person singular number. Otherwise, in lexeme level sound masculine plural will use $T_n6$ declension type. Thus same lexical type follows two different declension types based on its position in phrase or sentence. Furthermore, in some cases, these conditions are not explicitly mentioned in these sixteen noun classes. As an example, a singular triptote noun lexeme which is not pseudo sound but ends with *waw* is not addressed in the classification of sixteen types. Another example is أَبٌ (*abun* ) which is a triptote singular sound and follows $T_n1$ declension in lexeme level. But if it is used in a phrase then it becomes أَبُو (*abuw* ) and follows $T_n4$ declension. Thus, these sixteen classes of nouns is a complex system to identify right declension type of a noun lexeme. Our objective is to classify Arabic noun lexemes in such a way that each of these classes form a partition. This classification will capture these conditions computationally and eventually, identification of a declension type will be easier.

## 1.3 Scope of the Work

The diversity and importance of Arabic nominals is much broader than that of their counterparts in other languages. In Arabic, modifiers, such as adjectives and adverbs, are also treated as nominals. Like others, Arabic nominals show two types of morpho-syntactic operations: derivation and inflection. *Derivation* is the primary means of forming Arabic nouns. In Arabic, nouns can be derived from verbs or other nouns. On the other hand, *Inflection* refers to the variation in the form of a word, typically by means of an affix, that expresses a grammatical contrast which is obligatory for the stems in some given grammatical context. As an example, *'speakers'* is inflected from the stem *'speaker'*. This inflection is necessary if *'speaker'* is used in plural form. Here a suffix, *'s'* is used for the inflection. The word *'speakers'* is not a stem. But its category is the same as the category of *'speaker'*. Thus, this process is different from derivation as syntactic category does not change here. Formation of dual or plural from singular, formation of feminine gender from masculine and declension are some examples of inflection.

Declension is the process of disambiguating the grammatical roles of words by slightly changing their end vowels. Arabic declension has some unique features. In this thesis we limit our discussion of inflection on declension only.

In this thesis, we capture declension phenomena of Arabic grammar. Lexical construction by declension is significant part of this thesis. This includes construction of genetives and accusatives from nominative lexemes. Also for verb, from indicative lexemes, jussive and subjunctive lexemes can be constructed. For lexical construction of 16 noun classes of classical Arabic we consider *singular* noun classes that captures lexical constructions. This is because there is no regular pattern for declension of plural nouns. Analyzing plural declension needs further research. Phrasal construction is not included in the scope of this thesis as it is a vast area and needs further research. For verbs, we have also considered *singular* verb classes. We also analyze on lexical construction for definiteness, that is, construction of definite lexemes from indefinite lexemes.

## 1.4 Contributions

Our contributions in this research are as follows:

- We show the HPSG type hierarchy of Arabic noun and verb lexemes. This hierarchy maps noun and verb lexeme type to declension type.

- We propose Attribute Value Matrix (AVM) for Arabic nouns and verbs which captures morphological, syntactic and semantic effects.

- Classical sixteen categories is very complex system to identify declension types of nouns. To make it simple, we devise an algorithm which identifies declension type of a noun lexme. That is, this algorithm will be used to find the position of noun lexeme in type hierarchy. This will also show the completeness of this classification.

- Different types of nouns follow different declension types. We analyze this phenomenon and propose lexical construction rules for particular declension types of nouns and verbs. This will help to avoid numerous lexical entries. Because thousands of lexical entries will be replaced by only few rules. Hence lexical entry database will be smaller and performance will be improved.
- We design lexical construction rules to construct definite lexeme from indefinite.
- We verify the partial type hierarchy and construction rules in TRALE, which is a freeware platform developed in Prolog for HPSG implementation and validation.

## 1.5 Organization

Rest of the document is organized as follows. Chapter 2 provides preliminary discussion required for this thesis. It first discusses basic idea of linguistics and its different subfields. Then it discusses Arabic declension system and its effect on grammar. Lastly it discusses preliminaries of HPSG.

Chapter 3 is our contribution chapter. Firstly this chapter presents our proposed type hierarchy and mapping of type hierarchy to declension types. We also propose AVMs for both Arabic nouns and verbs. We provide an algorithm to identify declension type of a noun lexeme. Finally we propose construction rules to capture declension and definiteness.

We have implemented our research in TRALE platform as mentioned in Chapter 4. It starts with TRALE introduction and preliminaries. After that, it describes implementation procedures. In Appendix detail implementation has been presented.

Chapter 5 draws conclusion by mentioning our concrete contributions followed by future direction for further research.

# Chapter 2

# Background and Related Works

This chapter serves as a background for rest of the thesis, particularly related with linguistics, Arabic declension and HPSG. Hence this chapter is very important to understand the thesis. Section 2.1 provides background idea of linguistics by describing major subfields. Section 2.2 provides Arabic declension, its significance and different declension types. Here we start to use Arabic alphabet. A table for transliteration is given in Appendix A. We give HPSG basics in Section 2.3. Lastly, we present the state of the research works on HPSG modeling with emphasis on the Arabic language. For exhaustive studying and capturing related background information of this chapter we have used different publications [15, 16, 20, 24, 30, 38, 44–47, 49].

## 2.1 Linguistic Background

Linguistic means scientific and systematic study of human language. It is an interface between science and humanities. Linguistics has the following subfields ( [30]):

- Phonology: The study of specific sounds that make up words.
- Morphology: The study of word structures and variations.

- Syntax: How words are arranged into sentences.
- Semantics: The meaning of words.
- Pragmatics: How sentences are used to communicate messages in specific contexts.
- Discourse analysis: The highest level of analysis, looking at texts.

Among these, this thesis only highlights morphology, syntax and semantics. So we need to know some more details about these three subfields.

### 2.1.1 Morphology

Morphology is the subfield of linguistics which focuses on the study of formations of word in a language. It includes study of *morpheme* which is the smallest indivisible unit of a language that retains meaning. For example, the word "imperfections" is composed of four morphemes: *im + perfect + ion + s*. The root, *perfect*, is transformed from an adjective into a noun by the addition of *ion*, made negative with *im*, and pluralized by *s*. There are two types of morphemes: bound morpheme and free morpheme. **Bound morpheme** is the morpheme that cannot occur without any other morpheme. In our example *s* is a bound morpheme because it cannot be used alone. Bound morphemes are called **affix**. **Free morpheme** can occur alone. For example here *perfect* is free morpheme. **Root** means smallest meaningful word from where all affixes are removed and that cannot be analyzed further.

Using morphology new words can be formed from existing by any of two operations:

- Inflection: Inflection is variation in word form with an affix which is mandatory to express a grammatical context in sentence. For example "He *writes*". Here *write* is used with inflectional affix *-s*. Here we cannot write sentence like "He write" because it will be grammatically wrong. Again "They are writers": here *s* is inflectional affix.

- Derivation: By derivation new word is created from existing word by changing grammatical category (e.g verb to noun). For example "'He is a *writer*". Here *writer* is derived from word *write* with affix *-r*.

Based on two morpho-syntactic operations there are two types of affixes: inflectional affix and derivational affix.

A **stem** is the root or roots of a word, together with any derivational affixes, to which inflectional affixes are added. A root is also a stem. For example, *write* is a root and also a stem. Derivational affix *-r* has been added to *write* to form *writer.* So *writer* is also a stem. We can add inflectional affix *-s* to *writer* to convert to *writers* to express certain grammatical context.

A complete set of related word forms is called **linguistic paradigm**.

### Declension

Declension is one of the morphological features by which one form converts to another. It is the process of disambiguating grammatical roles of a word. It is the inflection of nouns, pronouns, adjectives or articles to express person, number, gender or case.

For example: alumnus is a singular masculine gender. Its declension for different numbers and genders is shown in Table 2.1.

| Gender/Number | **Singular** | **Plural** |
|---|---|---|
| **Masculine** | alumnus | alumni |
| **Faminine** | alumna | alumnae |

Table 2.1: Declension of *alumnus*

Again, declension of a pronoun according to case and person is shown in Table 2.2.

| Case/Person | **First** | **Second** | **Third** |
|---|---|---|---|
| **Nominative** | I | you | he |
| **Accusative** | me | you | him |
| **Genitive** | my | your | his |
| **Reflexive** | myself | yourself | himself |

Table 2.2: Declension of pronoun

### 2.1.2 Syntax

Syntax is the study of the structure of sentences. To know about syntax we need to get idea about some syntactic terms such as sentence, phrase, clause, grammatical categories etc.

#### Sentence, Clause and Phrase

In natural language, a sentence is an expression. It is composed of some words that indicate minimal syntactic relation between the words.

A **clause** is the smallest grammatical unit that can express a complete proposition. It is a group of words that is either a whole sentence or is a part of a sentence.

A phrase is a key grammatical unit. A phrase expresses one complete element of a proposition. It will be made up of one or more words and occupy a particular syntactic slot within its clause or sentence, e.g. as subject, predicate or object. The word that determines syntactic type of a phrase is called a **head**. Phrases are classified according to their head. For example **Noun phrase** is a phrase whose head is either noun or pronoun, **Verb phrase** is a phrase whose head is verb and **Prepositional phrase** is a phrase whose head is preposition.

**Grammatical Category**

Grammatical Category is a class of units such as noun, verb, prepositional phrase, finite clause and features such as case, mood, countability, gender, number. These may in turn be subcategorized into kinds of noun, case, mood etc. Here we discuss about the two important features case and mood.

**Grammatical case** means change in forms to indicate relative relation or role or grammatical function in a phrase, clause or sentence. For example, in the sentence "I have lost my pen" there are three cases or roles:

- Subject: I
- Object: pen
- Possessor: my

There are different types of cases which are listed in Table 2.3.

Different languages have different numbers of cases. For example Indo-European languages has 8 cases, Sanskrit has 6 different cases and Arabic has 3 cases.

Case is often marked by inflection. That is, in some language each case has a specific inflected form by which the case is identified. This is not true for all languages though.

**Grammatical mood** indicates relation of verbs to reality or intent in speaking. Possible moods are described in Table 2.4:

Different languages have different numbers of moods. Many languages indicate distinctions of moods by inflection of the verb form.

### 2.1.3 Semantics

Semantics is the study of meaning in language. In particular, it is the study of how meaning is structured in sentences, phrases, and words.

| **Case** | **Significance** | **Example** |
|---|---|---|
| Nominative | Subject | **Belal** has bought the doll |
| Accusative | Direct object | Belal has bought **the doll** |
| Genitive | Possession | Belal has bought **Matin's** doll |
| Dative | Indirect object | Belal has bought the doll **for Salam** |
| Locative | Place/time related | Belal has bought the doll **at Super market** |
| Ablative | Movement from something or cause | Belal has bought the doll **from Matin** |
| Instrumental | Instrument used to perform action | Belal wrote it **by hand** |
| Vocative | Addressing | Hello **Sir**! Listen to me. |

Table 2.3: Example of cases

| **Mood** | **Significance** | **Example** |
|---|---|---|
| Indicative | State of factuality and reality | Karim is good |
| Interrogative | Asking questions | Will you go? |
| Imperative | Command/request or prohibitions | Please help me |
| Conditional | Dependent upon another condition | He **will come** if I go |
| Optative | Hope/wishes | May you live long |
| Jussive | Pleading, wish, command or purpose etc (1st & 3rd person) | He shall help |
| Potential | Probability | He may do it |

Table 2.4: Example of moods

Semantics is very much related with reference that are used for agreement. Some of these are-

1. **Index agreement**: It arises when indices are required to be token identical. That is, the value of semantic index of a lexicon needs to similar with the same value of semantic index of other lexicon.

2. **Syntactic agreement**: There are some strictly syntactic objects (e.g. case, verb form, mood). That is, a lexicon has some syntactic requirements and these requirements can be fulfilled by other lexicons which has certain syntactic object values.

3. **Pragmatic agreement**: It arises when contextual background assumptions are required to be consistent.

In many languages, agreements are semantic rather than being sytactic only. For example, *the faculty is voting itself a raise* and *the faculty are voting themselves a raise.* In the two sentences, same *faculty* is used in two different numbers and this needs to be captured in semantic level. Like English, agreement in Arabic language is not syntactic; rather it is semantic. Which properties of referents are encoded by agreement features is subject to cross-linguistic variation, but common choices include person, number, gender, shape, humanness, animate/inanimate.

There are two types of semantic relationships holding between predicates and their arguments. The semantic role of the subject is *actor*, and indicates the entity responsible for the event. The semantic role of the object is *undergoer*, and indicates the entity which experiences the state or change of state described by the verb ( [44, 45]). In other words, the argument structure of an English transitive verb requires two arguments: the first argument (i.e., the subject) must be a semantic agent, and the second argument (i.e., the object) must be a semantic undergoer.

## 2.2 Arabic Declension

### 2.2.1 Grammatical States

In Arabic, there are three grammatical states of nouns i.e. three cases ( [46]):

- The state of رَفع (*rafʿ* ) - nominative case
- The state of نَصَب (*naṣab* ) - accusative case
- The state of جرّ (*ğrr* ) - genitive case

Table 2.5 shows example of three cases for same noun كِتَاب (*kitāb* ).

| **Case** | **Sentence** | **Word form** |
|---|---|---|
| Nominative | هَذَا كِتَابٌ (*hadā kitābun* )<br>Meaning: This is a book | كِتَابٌ (*kitābun* ) |
| Accusative | وَضعَ بِلَالٌ كِتَابًا (*waḍḥa bilālun kitāban* )<br>Meaning: Bilal put a book | كِتَابًا (*kitāban* ) |
| Genitive | هَذَا صُفهَتُ كِتَابٍ (*hadā ṣofhatu kitābin* )<br>Meaning: This is a page of a book | كِتَابٍ (*kitābin* ) |

Table 2.5: Different cases for same word *kitāb*

There are three states of Arabic verbs i.e. three moods ( [46]):

- The state of رَفع (*rafʿ* ) - Indicative mood
- The state of نَصب (*naṣab* ) - Subjunctive mood
- The state of جَزَم (*ǧazam* ) - Jussive mood

Table 2.6 shows example of three moods for same verb يَضرِبُ (*yaḍribu* )

| **Mood** | **Sentence** | **Word form** |
|---|---|---|
| Indicative | هُوَ يَضرِبُ (*huwa yaḍribu* )<br>Meaning: He beats | يَضرِبُ (*yaḍribu* ) |
| Subjunctive | هُوَ يُرِيدُ ان يَضرِبَكَ (*huwa yuriydu ’n yaḍribaka* )<br>Meaning: He wants to beat you | يَضرِبَ (*yaḍriba* ) |
| Jussive | أَخبِرنِي ان يَضرِبكَ (*aḫbirniy in yaḍribka* )<br>Meaning: Inform me if he beats you | يَضرِب (*yaḍrib* ) |

Table 2.6: Different moods for same word *yaḍribu*

### 2.2.2 Arabic Declension

Grammatical declension is known in Arabic as I’rab (إِعرَاب - *iʿrāb* ). By definition, i’rab is the process of disambiguating the grammatical roles of words by slightly changing their end vowels. According to declension, we can classify Arabic lexemes into two types -

declinable (مُعرَب - *muʿrab* ) and indeclinable (مَبني - *mabny* ). If a word experiences declension it is called declinable, and if it does not experience declension, or experiences it but does not show it, it is called indeclinable.

### 2.2.3 Role of Declension in Arabic Grammar

Declension plays more significant role in Arabic than most other languages. Because in Arabic, subject, object, predicate everything is determined by the end vowel. As an example, in English, subject and object are determined by sequence of words in a sentence. For example, "Zayeed beat Belal" and "Belal beat Zayeed". In the former sentence, Zayeed is the subject and Belal is the object whereas in the latter sentence, 'Belal' is the subject and 'Zayeed' is the object. But in Arabic, ضربَ زيدٌ بِلَالًا (*ḍrba zydun bilālan* ) and ضربَ بِلَالًا زيدٌ (*ḍrba bilālan zydun* ) these two sentences have same meaning - *Zayeed beat Belal.* In both sentences, زيدٌ (*zydun* ) is the subject as it is ended with short form of وَاو (*wāw* ) and بِلَالًا (*bilālan* ) is the object as it is ended with short form of اَلف (*alif* ). Thus end vowel implies grammatical cases for nominals in Arabic.

Let us see another example - ضربَ زيدٌ بِلَالًا بِالخشبةِ (*ḍrba zydun bilālan biālḫšbti* - *Zayeed beat Belal by stick*). In this sentence, زيدٌ (*zydun* ) is in nominative case. So here no change occurs and the ending remains the same. But بِلَالٌ (*bilālun* ) is in accusative case. So بِلَالٌ (*bilālun* ) declines to بِلَالًا (*bilālan* ). On the other hand ضربَ (*ḍrba* ) is indeclinable. So its ending vowel always remains the same. الخشبةٌ (*ālḫšbtun* ) is in genitive case. So الخشبةُ (*ālḫšbtu* ) declines to الخشبةِ (*ālḫšbti* ). Table 2.7 shows these declension forms.

| **Nominative form** | **Case in sentence** | **Declined form** |
|---|---|---|
| زيدٌ (*zydun* ) | Nominative | زيدٌ (*zydun* ) |
| بِلَالٌ (*bilālun* ) | Accusative | بِلَالًا (*bilālan* ) |
| الخشبةٌ (*ālḫšbtun* ) | Genitive | الخشبةِ (*ālḫšbti* ) |

Table 2.7: Different declensions in sentence ضربَ زيدٌ بِلَالًا بالخشبةِ

## 2.2.4 Case Marking

In Section 2.2.3 we have seen that grammatical function of a word in a sentence is determined by the end vowel. It is called case marking. For subject and object there are separate case marking. Each case marker corresponds to one of the three different cases that described in Section 2.2.1. There are three case markers:

1. Damma (ـُ - *u* )

2. Kasra (ـِ - *i* )

3. Fatha (ـَ - *a* )

For some cases, the marker is nunated. For example, for indefinite noun, Damma is pronounced as *un* (ٌ) instead of *u* (ُ), Kasra is pronounced as *in* (ٍ) instead of *i* (ِ), Fatha is pronounced as *an* (ً) instead of *a* (َ). As an example اَلمَكتُبُ (*almaktubu* - “the book”) is definite which is ended with *u* (ُ). On the other hand, مَكتُبٌ (*maktubun* - “book”) is indefinite and it is ended with *un* (ٌ).

### Declension Types

Arabic declension can be classified according to three dimensions based on different forms of vowels which are used for declension. These dimensions are described below.

- **Visibility of declension**: This dimension determines whether all possible forms of vowels are explicitly shown on the final letter of a noun for all three cases. Along this dimension, declension can be classified into three categories: Visible declension, Partially invisible declension and Completely invisible declension.

  There are nouns which cannot explicitly show one or more vowels on their final letters. In this case, it is assumed that if the noun were able to show vowels on its final letter, then it would do so. So, here assumed vowels are used to reflect the declension type. For example, حُسنٰى (*ḥusnā* ) cannot show ُ (*u* ) and ِ (*i* ) on its final letter because of the presence of ى (*ā* ) (Ya maqsura). So, حُسنٰى (*ḥusnā* ) follows *completely invisible declension.* On the other hand, حَسَنٌ (*ḥasanun* ) can show all short forms of vowels on its final letter: حَسَنٌ (*ḥasanun* ) is in nominative case; when it is in accusative case it becomes حَسَنًا (*ḥasanan* ) and when it is in genitive case it becomes حَسَنٍ (*ḥasanin* ). So, حَسَنٌ (*ḥasanun* ) follows *visible declension.* There are some declension types where some forms are visible and rest are invisible. We call it *partially invisible.* For example, قَا ضِيًا (*qā ḍiy* ) cannot show ُ (*u* ) and ِ (*i* ) on its final letter because of the presence of ى (*ā* ) (Ya maqsura) in Genitive and Nominative case.

- **Vowel form used to decline**: This dimension determines whether the vowel used for declension is in short or long form. There are some words which use وَاو (waw) to reflect the nominative case, أَلف (*alf* ) to reflect accusative and يَاء (*yāʾ* ) to reflect the genitive case. For example, ذُو (*ḏuw* ) is in nominative case, ذَا (*ḏā* ) is in accusative case and ذِي (*ḏiy* ) is in genitive case. On the other hand, حَسَنٌ (*ḥasanun* ) shows declension with a short vowel.

- **Completeness of declension**: This dimension determines whether all vowels are used in the declension forms. There are words which do not use all vowels to show all cases; rather same vowel is used to reflect accusative and genitive cases. For example, مَسَاجِدُ (*masāǧidu* ) uses َ (*a* ) to reflect both accusative and genitive cases. Here ُ (*u* ) is not used. For this reason this is an example of incomplete declension. On the other hand, حَسَنٌ (*ḥasanun* ) is an example of complete declension as it can use all forms of short vowels to reflect the declensions. Notably, if a declension type is partially or completely invisible, then the completeness dimension will not be applicable to that declension type.

Table 2.8 shows examples of the above three dimensions of declensions [24]. Through inheriting different combination of divisions of these dimensions, nine ways can be found by which grammatical cases are represented. For example, visible complete declension with long vowel, partially invisible with long vowel, invisible declension with these short vowel etc. These nine declension types can be expressed as $T_n1$, $T_n2$, $T_n3$, …, $T_n9$, as shown in Table 2.9. In traditional Arabic, nouns are categorized into sixteen classes which are shown in Table 2.9. These noun classes can be declined by these nine declension types as shown in Table 2.9.

| Dimensions | Partitions | Genitive | Accusative | Nominative |
|---|---|---|---|---|
| | Visible declension | حَسَنٍ<br>*ḥasanin* | حَسَنًا<br>*ḥasanan* | حَسَنٌ<br>*ḥasanun* |
| Visibility | Completely invisible declension | حُسنٰى<br>*ḥusnā* | حُسنٰى<br>*ḥusnā* | حُسنٰى<br>*ḥusnā* |
| | Partially invisible declension | قَا ضِي<br>*qā ḍiy* | قَا ضِيًا<br>*qā ḍiyan* | قَا ضِي<br>*qā ḍiy* |
| Vowel form | Declension with short vowel | حَسَنٍ<br>*ḥasanin* | حَسَنًا<br>*ḥasanan* | حَسَنٌ<br>*ḥasanun* |
| | Declension with long vowel | ذِي<br>*ḏiy* | ذَا<br>*ḏā* | ذُو<br>*ḏuw* |
| Completeness | Complete declension | حَسَنٍ<br>*ḥasanin* | حَسَنًا<br>*ḥasanan* | حَسَنٌ<br>*ḥasanun* |
| | Incomplete declension | مَسَاجِدَ<br>*masāǧida* | مَسَاجِدَ<br>*masāǧida* | مَسَاجِدُ<br>*masāǧidu* |

Table 2.8: Different dimensions of declension

Some grammatical terms in this table are explained below.

- *Triptote* refers to words that take all three short vowel case endings, where each one differentiates a particular case. For example, زَيدٌ (*zaydun* ) is in nominative case and زَيدًا (*zaydan* ) and زَيدٍ (*zaydin* ) are in accusative and genitive respectively.

- *Diptote* only exhibits two case markers: ـُ (*u* ) for nominative and ـَ (*a* ) for both

| Types | Declension | Noun class | Genitive | Accusative | Nominative |
|---|---|---|---|---|---|
| Type 1 | $T_1$ | 1. Triptote sound singular<br>2. Singular noun pseudo sound<br>3. Triptote broken plural | ـٍ | ـً | ـٌ |
| Type 2 | $T_2$ | 4. Sound feminine plural | ـٍ | ـٍ | ـٌ |
| Type 3 | $T_3$ | 5. Diptote without prefixed by definite markness أَل<br>or not a possessed in a possessive phrase | ـَ | ـَ | ـُ |
| Type 4 | $T_4$ | 6. أَب أَخ حم هن فم ذو possessed<br>not towards first person singular number possessor | ي | ا | و |
| Type 5 | $T_5$ | 7. Dual noun<br>8. كلَا and كلتَا possessed towards pronoun<br>9. اِثنَان and اِثنَتَان | ي | ي | ا |
| Type 6 | $T_6$ | 10. Sound masculine plural<br>11. Multiple of ten between twenty and ninety<br>12. أُولو (plural of possessor) | ي | ي | و |
| Type 7 | $T_7$ | 13. Noun possessed towards<br>first person personal pronoun<br>14. ending with ya maqsoora (ى) | ـٍ | ـً | ـٌ |
| Type 8 | $T_8$ | 15. Noun ending with ي | ـٍ | ـً | ـٌ |
| Type 9 | $T_9$ | 16. Sound masculine plural<br>possessed towards personal pronoun | ي | ي | و |

Table 2.9: Noun classes according to 9 declension types

genitive and accusative. For example, قَوافِلُ (*qa''filu* ) is in nominative case and قَوافِلَ (*qa''fila* ) is used for both genitive and accusative case.

- *Ending type* denotes type of end letter- whether consonant or vowel. *Sound nouns* are those where end letters are consonants, for example, زَيدٌ (*zaydun* ). For an *unsound noun*, end letter is a vowel (ا (*ā* ) or و (*w* ) or ي (*y* )). For example, قَا ضِي (*qā ḍiy* ) is a unsound noun.

- *Pseudo sound* ends with و (*w* ) or ي (*y* ) and there is sakin [2] on the letter before the last letter. It is actually unsound, but it follows declension like a sound noun. For example, دَ لوٌ (*da lwun* ).

In Arabic, possessive phrase is called *Mudaf-Mudaf Ilayh*, where possessed is called *Mudaf* and possessor is called *Mudaf Ilayh*. For example, كِتَابُاللَّهِ (*kitābu'l-lahi* - 'The book of Allah'). كِتَابُ (*kitābu* - 'book') is the possessed and اَللَّهِ (*al-lahi* - 'Allah') is the possessor.

There are 4 possible ways by which Arabic verbs can be declined and 5 classes of Arabic verbs. Mapping of these 5 classes to 4 declension types is shown in Table 2.10.

| Types | Verb class | Jussive | Subjunctive | Indicative |
|---|---|---|---|---|
| Type 1 | 1. Sound singular | jawazim | ـَ | ـُ |
| Type 2 | 2. Unsound by و<br>3. Unsound by ي not ending with ن | Extinction ل | ـَ | ـُ |
| Type 3 | 4. Unsound by أ | Extinction ل | ـَ | ـُ |
| Type 4 | 5. Imperfect ending with ن | Extinction ن | Extinction ن | ن |

Table 2.10: Verb classes according to 4 declension types

[2]Sakin is absence of short vowel on a letter

## 2.3 HPSG Preliminaries

### 2.3.1 HPSG and Other Grammars

To formalize natural language a grammar is required. Probable grammars for natural language processing are ( [49]):

- Generalized Phrased Structure Grammar (GPSG) ( [18])
- Lexical Functional Grammar (LFG) ( [8])
- Head Driven Phrase Structure Grammar (HPSG) ( [40])

Among the above, HPSG is very different from GB, GPSG, LFG and other contemporary grammar models in architecture as well as formal and linguistic content. HPSG is the immediate successor to GPSG.

HPSG is flexible for adding different data types for grammatical description, changing individual components of a grammar, applying the framework to new languages and further development of the overall framework. Moreover, grammar model, universal grammar and particular grammars are expressed in a uniform powerful formalism.

### 2.3.2 Sign Based Construction Grammar

In this thesis we use Sign Based Construction Grammar (SBCG) ( [47]) version of HPSG. SBCG is an attempt to adapt ideas developed in HPSG after long research. It is an alternative to derivational (movement-based) theories of grammar. It synthesizes ideas developed in HPSG for its theoretical foundations.

### 2.3.3 Feature Structure

A feature structure is essentially a set of attribute-value pairs. For example, a probable value of attribute *gender* is *masculine*. A value of an attribute can be atomic/single symbol or another feature structure. Feature's value can be any of the four possible types ( [20]):

- Atomic sort or single value
- A feature structure
- A set of feature structures
- List of feature structures

Attribute value pairs can also be expressed using **Attribute value matrix (AVM)**. In Figure 2.1 we can see example of a AVM. There are two columns in the matrix. First is for attributes (which is also called features) and another is its value. In the example we can see values of feature of CAT (category) is *noun*, that is a single symbol. The value is atomic. But value of feature AGR (agreement) is another AVM or feature structure. The inner feature structure contains three features person, gender and number and value of these are *1st*, *masc* (masculine) and *sing* (singular) respectively.

$$\begin{bmatrix} \text{CAT} & noun \\ \text{AGR} & \begin{bmatrix} \text{PERSON} & 1st \\ \text{GENDER} & masc \\ \text{NUMBER} & sing \end{bmatrix} \end{bmatrix}$$

Figure 2.1: Example of AVM

### Sort Description

The type of each feature structure can be expressed by sort description. Sort description is an atomic value and written at top left position of a feature structure. For example, sort description for third person singular nouns can be $3rd - sg - noun - lex$. Figure 2.2 shows a feature structure with sort description *sort1*.

$$\begin{bmatrix} \textit{sort1} & \\ \textsc{Feature1} & \textit{value1} \\ \textsc{Feature2} & \textit{value3} \end{bmatrix}$$

Figure 2.2: Feature structure with sort description ‘sort1’

**Structure Sharing**

The main explanatory mechanism in HPSG is that of structure-sharing, equating two features as having the exact same value (token-identical). An example of structure sharing is shown in Figure 2.3. Here in Feature4 is sharing value with Feature3B and Feature5 is sharing value with Feature2. That means, the value of Feature4 will be same as the value of Feature3B, i.e., $value3b$.

$$\begin{bmatrix} \textit{sort1} & \\ \textsc{Feature1} & \textit{value1} \\ \textsc{Feature2} & \boxed{2}\ \textit{value3} \\ \textsc{Feature3} & \begin{bmatrix} \textsc{Feature3a} & \textit{value3a} \\ \textsc{Feature3b} & \boxed{1}\ \textit{value3b} \end{bmatrix} \\ \textsc{Feature4} & \boxed{1} \\ \textsc{Feature5} & \boxed{2} \end{bmatrix}$$

Figure 2.3: Example of structure sharing

### 2.3.4 Sign

SBCG describes language in terms of constraints on linguistic expressions which is called *sign*. Sign has PHON, MORPH, SYN and SEM feature to express phonological, morphological, syntactic, semantic subfields of linguistics respectively. A typical sign has been shown in Figure 2.4.

The value of PHON is phonological phrase $\phi - phr$. The values of other three features is feature structure of type of corresponding object i.e. morphological-object (*morph-obj*),

$$\begin{bmatrix} \text{PHON} & \phi - phr \\ \text{MORPH} & \textit{morph-obj} \\ \text{SYN} & \textit{syn-obj} \\ \text{SEM} & \textit{sem-obj} \end{bmatrix}$$

Figure 2.4: HPSG sign

syntax-object (*syn-obj*) and semantic-object (*sem-obj*).

**ARG-ST** (Argument structure) is a feature of sign which lists syntactico-semantic arguments. For example, for transitive verb *donate* has an ARG-ST list as follows: $< NP, NP, PP >$. Here first NP is verb's subject, second NP is verb's direct object and the last one is prepositional phrase.

SYN feature structure includes features like CAT (category), VAL (valence) and MRKG (marking). CAT is a complex feature which includes CASE to indicate cases of nouns, MOOD to indicate mood of verbs, VFORM to specify morpho-syntactic category of a verb, AUX (auxiliary) to indicate whether a verb is an auxiliary and VOICE for verb.

VAL expresses degree of saturation. The value of MRKG can be *unmk* (unmarked) in case of unmarked signs or any of *that, whether, than, det* etc. For example, *that Pat wrote*: its MRKG value is *that* but *Pat wrote*: its MRKG value is *unmk*.

An example of syn-obj is shown in Figure 2.5

SEM feature structure includes two features: INDEX and FRAMES. INDEX is to indicate the referent of an expression. For a noun phrase (NP) it indicates the subject and for verb phrase it indicates the verb.

In HPSG, the semantic information is expressed in Minimal Recursion Semantics (MRS), as developed in CSLI's Linguistic Grammars Online (LinGO) project [15, 16]. Most semantic information in MRS is contained under the feature FRAMES. That is, FRAMES is to indicate predications that together determine meaning of a sign. Value

$$\begin{bmatrix} \textit{syn-obj} & \\ \textsc{cat} & \begin{bmatrix} \textit{verb} & \\ \textsc{mood} & \textit{subj} \\ \textsc{vf} & \textit{fin} \\ \textsc{aux} & + \end{bmatrix} \\ \textsc{val} & \textit{none} \\ \textsc{mrkg} & \textit{unmk} \end{bmatrix}$$

Figure 2.5: Example of syntactic object

of FRAMES is list of frames. A *frame* is an elementary scene in which certain semantic roles are specified and specific participants are assigned to them. As an example, in eating frame, participants are an actor (who does the eating) and the food (which is eaten). SIT (situation) is used to indicate verb index in a frame. Figure 2.6 shows an example of *sem-obj*.

$$\begin{bmatrix} \textit{sem-obj} & \\ \textsc{index} & i \\ \textsc{frames} & \left\langle \begin{bmatrix} \textit{eat-fr} & \\ \textsc{sit} & s \\ \textsc{actor} & i \\ \textsc{undergoer} & j \end{bmatrix} \right\rangle \end{bmatrix}$$

Figure 2.6: Example of semantic object

### 2.3.5 Construct

Derivational and inflectional constructions fit uniformly into a two-level mode, one that is articulated in terms of a mother and its daughter(s). For example, the verbal word whose form is *laughed* is constructed from the verbal lexeme whose form is *laugh*. The resulting mother-daughter configuration is a construct ( [47]).

In order to express constructional generalizations in a systematic way, it is useful, as indicated above, to model constructs as feature structures of the form sketched in Figure 2.7.

$$\begin{bmatrix} \text{MTR} & \textit{sign} \\ \text{DTRS} & \textit{list(sign)} \end{bmatrix}$$

Figure 2.7: HPSG construct

The value of MTR (mother) is a sign which is constructed from a given construct. The value of DTRS (daughters) is a list of signs which are used to form the mother.

### 2.3.6 The Sign Principle

Signs in SBCG are licensed by a general grammatical principle ( [47]) which can be formulated as follows ( [48]): Every sign must be lexically or constructionally licensed, where:

- A sign is lexically licensed only if it satisfies some lexical entry
- And a sign is constructionally licensed only if it is the mother of some construct

A *lexical entry* is thus a feature structure description that describes a class of lexical signs: lexemes or words that is lexical entry is used to formulate a sign.

## 2.4 Related Works

HPSG analysis of Semitic language is comparatively a new area of research. Few research works address the problem of HPSG modeling for Semitic languages. Among these languages, HPSG modeling of Hebrew is not new but it lacks its coverage on morphology. In

2000, Nathan Vaillette presented a paper on Hebrew relative clauses [51]. In this paper, he nicely modeled the phrasal construction rules to capture Hebrew relative clauses. He did not put emphasis on morphological operation.

Morphology of Sierra Miwok and French were modeled in HPSG by phonological realization [6]. The author also showed how nonconcatenative morphology can be captured by his framework. He further mentioned the idea how consonant and vowel melody forms the word in Arabic. But he did not show any construction rule for any language.

An HPSG formalism of morphologically complex predicate is outlined in [13]. Here the author mostly focused on syntax and semantics of causative construction. He used lexical rule with semantic frames to capture morphological effect. As Japanese is an Agglutinative language, the morphology used here is concatenative morphology.

Intricate nature of Arabic morphology attracts several series of research projects [1, 9, 50]. These research projects are mainly based on development of toolkit for Arabic morphological analysis. These projects are not based on compiler development, rather these are dedicated for morphological analyzer which designs and implements finite state morphological models. From linguistic perspective, these models describe rules of lexicon development and derive lexicons.

Riehemann modeled concatenative morphology in German and English using HPSG formalism in 1998 [42,43]. In that paper, the author captured the morphological derivation by a special feature called MORPH-B, which means morphological base. MORPH-B feature serves the purpose of derivation. MORPH-B feature can also be used to capture nonconcatenative morphology. In 2001, Riehemann extended the previous work and added nonconcatenative morphology for Hebrew verbal nouns [43].

In 2006, an HPSG analysis of Arabic broken plurals and gerunds were presented [27]. Main assumption in that work revolves around the Concrete Lexical Representations (CLRs) located between an HPSG type lexicon and phonological realization. Here, HPSG sign was represented using CLR function and not by AVM. This function put more emphasis on phonology instead of morpho-syntactic operations.

HPSG modeling of Arabic triliteral strong verbs was proposed in 2008 [3–5]. In these papers, the authors have shown regular morphology of Arabic verbs. The authors designed the SBCG AVM of Arabic verbs. The authors also designed several constructions of verb lexeme and morphologically complex predicates (MCP). The authors did not propose any implementation methodologies to implement the construction rules proposed in their works.

In [36], HPSG formalization for Arabic nominal sentences has been presented. That formalization covers seven types of simple Arabic nominal sentences while taking care of the agreement aspect. The formalization presented in [36] has been implemented using the Linguistic Knowledge Building (LKB) ( [14]) system. Additionally, Mutawa et al.'s work is based on the assumption that agreement information in Arabic arises from syntactic rules. However in Section 3.4, we have established that agreement in Arabic is not always syntactic and the agreement feature needs another feature, humanness (HUM), which is not mentioned in that work.

In [21], authors has studied the typology of the Arabic relative sentences and proposed an Arabic HPSG Grammar. That work has specified an Arabic lexicon and proposed the grammar in Type Description Language (TDL) ( [28]).

An in-depth analysis of declensions of Arabic nouns has been presented in [23]. Here the authors discussed different dimensions of declension. But they did not show or discuss any mapping of lexical types to declension types, which is necessary to implement the declension phenomenon in HPSG.

Thus to the best of our knowledge, the rich declension phenomenon of Arabic Nominals has not yet been explored in the literature. This motivates us to do research in this particular area.

# Chapter 3

# HPSG Formalism

In this chapter, we show the type hierarchy of Arabic noun lexemes and verb lexemes. We propose AVM for Arabic nouns and verbs. Then we show the mapping from type hierarchy to DEC and VDEC feature for nouns and verbs respectively. We show the algorithm which identifies the declension type of a noun lexeme from the type hierarchy. From this algorithm, we show that any type of noun lexemes must have one distinct declension type in a particular condition. We also show construction rules for each case or mood and definiteness to eliminate lexical entries.

## 3.1 Type Hierarchy

In this section, we discuss about type hierarchy for both nouns and verbs.

### 3.1.1 Type Hierarchy for Noun Lexemes

We can classify Arabic nouns based on several dimensions. For example:

- Number
- Derivation

- Gender
- Declension
- Ending type

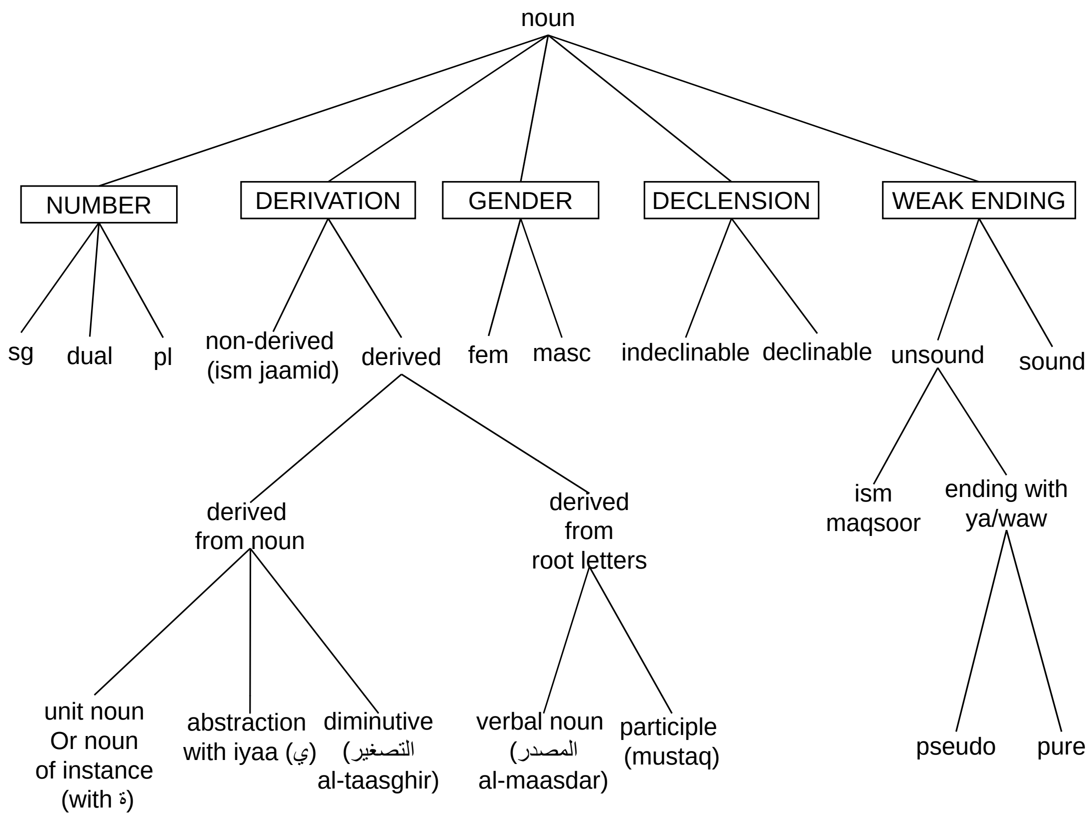


Figure 3.1: Lexical type hierarchy of noun lexemes

The type hierarchy of Arabic noun lexemes is shown in Figure 3.1 based on this dimensions. *Derivation* is explored by [22]. In fact, classification along *derivation* is not subjected to declension. Among these dimensions, *end letter type* deserves some explanation. The first level of classification along this dimension is as follows:

- Sound (end letter is consonant)

- Unsound (end letter is vowel i.e ا (*ā* ) or و (*w* ) or ي (*y* )). It can be classified as follows:
  - Ism maqsoor (alif ending)
  - Ending with ya/waw. It can be further classified to:
    - Pseudo sound (ending with ي (*y* ) or و (*w* ) and its letter has sakin)
    - Pure unsound.

### 3.1.2 Type Hierarchy for Verb Lexeme

Arabic verbs can be classified into two dimensions based on declension pattern. These are:

- Number: Three possible values of number are: sg, dual and pl to denote singular, dual or plural respectively.
- Ending type: Ending type can be sound or unsound. If it is unsound then we can again divide into two types: ending with أ (*a* ) and ending with ي (*y* ) or و (*w* )

Type hierarchy of Arabic verb lexemes is shown in Figure 3.2 based on these dimensions.

## 3.2 Mapping Type Hierarchy to Declension Type

In this section, we map our type hierarchy to declension type for both nouns and verbs. From the type hierarchy, we first form lexical types that formed by multiple inheritances. Then we map each lexical type to a particular declension type.

### 3.2.1 Noun Type Hierarchy to Declension Type Mapping

As discussed in Section 2.2.4, there can be nine possible declension types of noun lexemes. Figure 3.3 shows the mapping of these declension types from lexical type hierarchy. Note

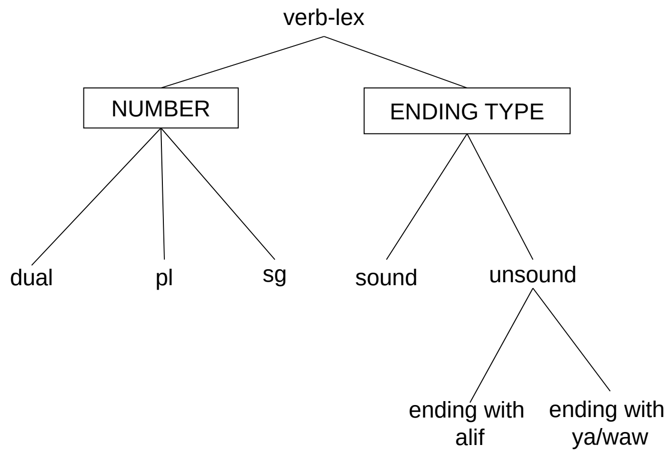


Figure 3.2: Lexical type hierarchy of verb lexemes

that, derivation has no effect on declension. So dimension along derivation is not shown in Figure 3.3. This figure shows subtypes which are formed by multiple inheritances indicated by dotted lines. For example, $triptote - sound - sg - noun - lex$ is type of a noun lexeme which is a subtype of triptote, sound and singular. Declension type for each sub type is shown inside parenthesis. Declension type of lexical type $triptote - sound - sg - noun - lex$ is $T_n1$ which indicates that lexical type $triptote - sound - sg - noun - lex$ follows declension type 1.

For simplicity, we have not mentioned lexical type for other subtypes though we have shown corresponding declension types. We can also observe that there are three lexical types which follow declension type 1. These are - $triptote - sound - sg - noun - lex$, $triptote - pseudo - sg - noun - lex$ and $triptote - broken - pl - noun - lex$. Notably, $T_n4$ and $T_n9$ declension types are only found in phrase levels. That is why, these are not shown in this mapping.

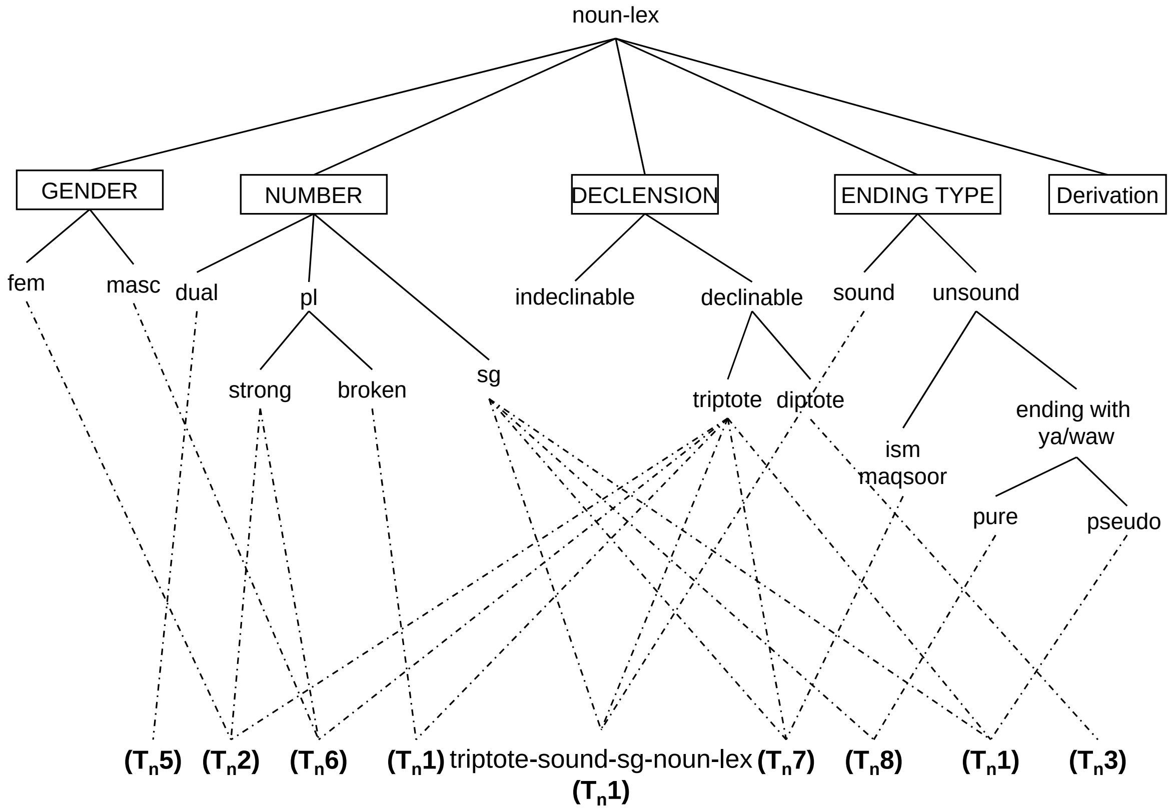


Figure 3.3: Mapping of declension type from type hierarchy of noun lexeme

### 3.2.2 Verb Type Hierarchy to Declension Type Mapping

Mapping of declension type from type hierarchy of verb lexeme is shown in Figure 3.4. Here we can see mapping to 4 declension types that are mentioned in Table 2.10. Type of verb lexemes is formed by multiple hierarchies and each type has a particular declension type. For example, type $sound - sg - verb - lex$ is formed from sound and singular by multiple inheritance and it follows declension type $T_v1$.

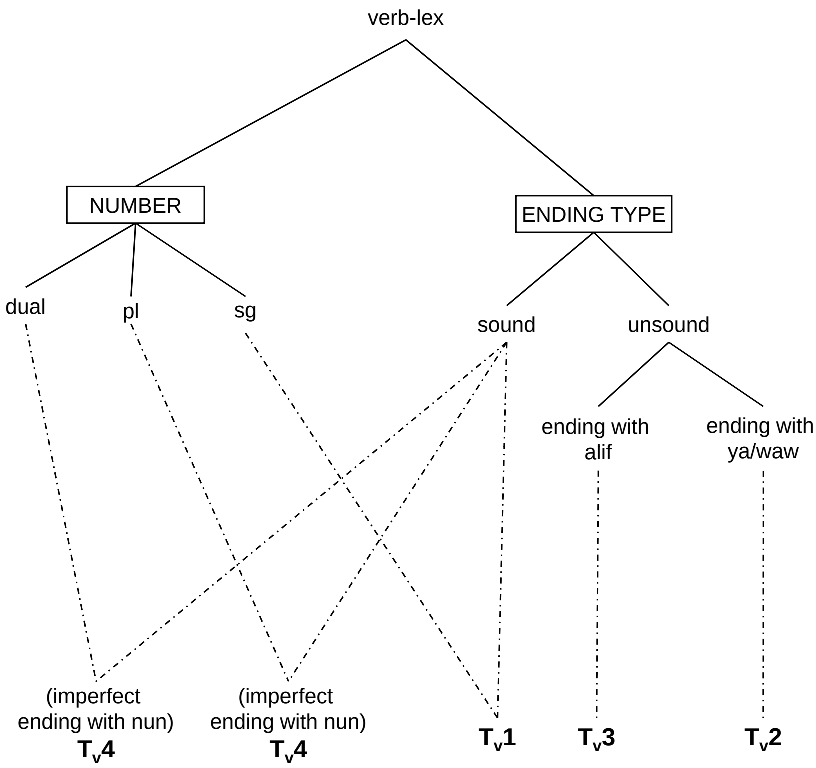


Figure 3.4: Mapping of declension types from type hierarchy of verb lexemes

## 3.3 Algorithm to Find Declension Type of Nouns

In Section 3.2, we have shown mapping of declension types from lexical type hierarchy. From this mapping, we devise an algorithm which determines the declension type of a noun lexeme. The flowchart of this algorithm is shown in Figure 3.5. This will be an offline method by which the declension type of a lexeme will be identified.

From this flowchart, it is clear that each noun lexeme must have a declension type. This is because, at each decision maker, the noun lexemes are subjected to two new partitions. Thus, all noun lexemes are completely partitioned. In other words, every noun lexeme must have a declension type which can be determined from this flow chart.

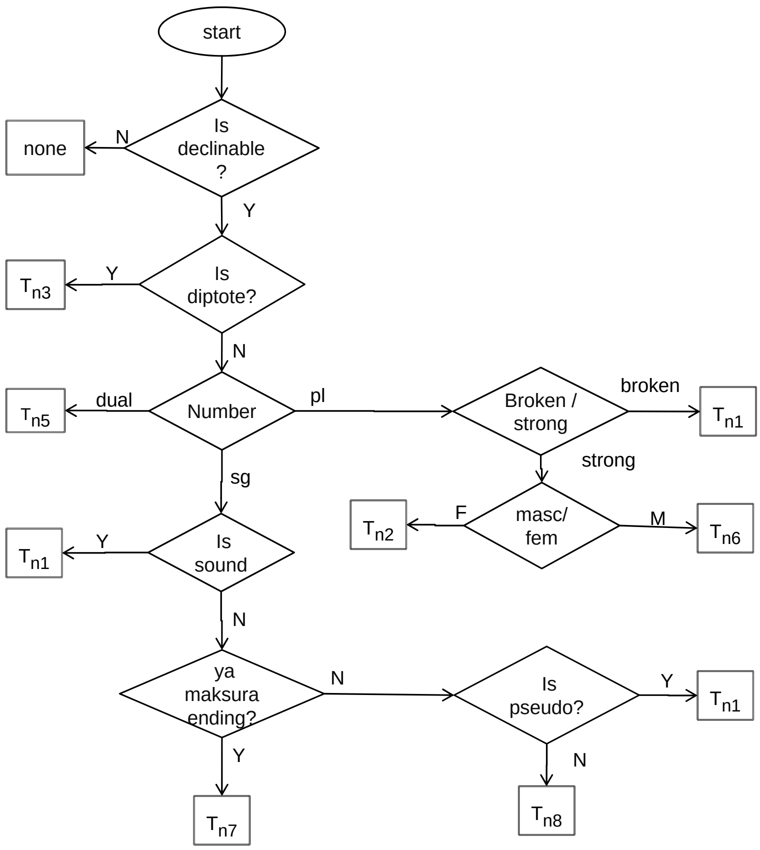


Figure 3.5: Algorithm to find the declension type of a noun lexeme

## 3.4 Arabic AVM

In this section we first mention SBCG AVM for English nouns and then extend it for Arabic. Then we will propose AVM for Arabic verbs.

### 3.4.1 AVM for Arabic Noun

We modify the SBCG feature geometry for English and adopt it for Arabic. The SBCG AVMs for nouns in English [47] and in Arabic are shown in Figure 3.6 and Figure 3.7, respectively.

$$\begin{bmatrix} \textit{noun-lex} & \\ \text{PHON} & [] \\ \text{FORM} & [] \\ \text{ARG-ST} & list(sign) \\ \text{SYN} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \textit{noun} & \\ & \\ \text{CASE} & \ldots \\ \text{SELECT} & \ldots \\ \text{XARG} & \ldots \\ \text{LID} & \ldots \end{bmatrix} \\ \text{VAL} & list(sign) \\ \text{MRKG} & mrk \end{bmatrix} \\ \text{SEM} & \begin{bmatrix} \text{INDEX} & i \\ \text{FRAMES} & list(frame) \end{bmatrix} \end{bmatrix}$$

Figure 3.6: AVM for English noun

The PHON deals with phonology and hence, it is not related to morphology and its effect. For this reason, this feature is out of the scope of this thesis. Three main function features - MORPH, SYN and SEM are discussed here.

The MORPH feature captures the morphological information of signs and replaces the FORM feature of English AVMs. This feature is similar to MORPH feature used for Hebrew verbal nouns [43]. The value of the feature FORM is a sequence of morphological

$$
\begin{bmatrix}
\textit{noun-lex} & \\
\textsc{phon} & [\,] \\
\textsc{morph} & \begin{bmatrix} \textsc{root} & list(letter) \\ \textsc{skeleton} & list(letter) \\ \textsc{dec} & \ldots \end{bmatrix} \\
\textsc{arg-st} & list(sign) \\
\textsc{syn} & \begin{bmatrix} \textsc{cat} & \begin{bmatrix} \textit{noun} & \\ & \\ \textsc{case} & \ldots \\ \textsc{def} & \ldots \\ \textsc{select} & \ldots \\ \textsc{xarg} & \ldots \\ \textsc{lid} & \ldots \end{bmatrix} \\ \textsc{val} & list(sign) \\ \textsc{mrkg} & mrk \end{bmatrix} \\
\textsc{sem} & \begin{bmatrix} \textsc{index} & \begin{bmatrix} \textsc{person} & \ldots \\ \textsc{number} & \ldots \\ \textsc{gender} & \ldots \\ \textsc{hum} & \ldots \end{bmatrix} \\ \textsc{frames} & list(frame) \end{bmatrix}
\end{bmatrix}
$$

Figure 3.7: AVM for an Arabic noun (extended from English noun AVM)

objects (formatives); these are the elements that will be phonologically realized within the sign's PHON value [47]. On the other hand, MORPH is a function feature. It does not only contain these phonologically realized elements but also contains their origins. MORPH contains three features - ROOT, SKELETON and DEC.

1. ROOT feature contains root letters for the following cases:

   (a) The root is characterized as a part of a lexeme, and is common to a set of derived or inflected forms

   (b) The root cannot be further analyzed into meaningful units when all affixes are removed

   (c) The root carries the principal portion of the meaning of the lexeme

   In other cases, the value of this feature is empty.

2. SKELETON contains not only stem but also inflected word. That is, it contains all the letters that constitutes the word. It is a sequence of morphological objects which are phonologically realized. It will include both lexical formatives and affixes. For example, for Arabic word كَاتِبًا (*kātiban* ), value of FORM will be $< kaatib+an >$. For كَاتِبٌ (*kātibun* ), value of FORM will be $< kaatib + un >$. This feature is very much similar to the feature SKELETON used in [42].

3. DEC feature is a significant addition in MORPH of Arabic AVM. This feature indicates the declension type as discussed in Section 2.2.4. Declension type is a morphological feature. Hence DEC is placed under MORPH. It determines how the end vowel of a noun lexeme changes to reflect its case. The change of end

vowel changes the form of a lexicon. As discussed in Section 2.2.4, there exists nine possible ways in which grammatical cases can be represented on an Arabic noun. So for a declinable noun, the value of the DEC feature will be one of $T_n1$, $T_n2$, $T_n3$, ..., $T_n9$, corresponding to the nine declension types presented in Table 2.9. The value of this DEC feature can be determined from the type hierarchy mentioned in Figure 3.3. For indeclinable nouns, the value of the DEC feature will be *none*.

ARG-ST feature contains syntactico-semantic arguments. In our case, a noun lexeme doesn't have any arguments as it is detached. Though accusative lexeme has a governor requirement but it will be captured by the AVM of the governor. That is governor lexeme will contain governed as an argument in ARG-ST. For example, كِتَابًا (*kitāban* - *book*) is in accusative case and it's ARG-ST will be empty. But كَتَبَ (*kataba* - *write*) is the governor of كِتَابًا (*kitāban* ). So ARG-ST of كَتَبَ (*kataba* ) will contain كِتَابًا (*kitāban* ).

SYN feature contains CAT, VAL and MRKG features. We modify the CAT feature of SBCG to adopt it for Arabic language. Note that, for all kinds of verbal nouns, the sort description of the CAT feature is *noun*. In Arabic there are only three parts of speech (POS) for lexemes or words: noun (in Arabic, pronoun is also considered as noun), verb and particle. In the case of the Arabic noun, the CAT feature consists of CASE, DEF, SELECT, XARG and LID features. As Arabic has three cases for noun, the value of CASE will be either *nominative*, *accusative* or *genitive*.

Among these features, we introduce the DEF feature, which is used for syntactic agreement in phrasal construction. This feature also strengthens our design. The DEF feature denotes the value of definiteness of an Arabic noun. There are eight ways for a noun word or lexeme to become definite [25]:

1. A word made definite by means of the definite article: أَل (*al* )

2. A sentence can be made definite by means of a relative pronoun: "the car that was driven"

3. Demonstrative pronouns: "This", "That"

4. Proper nouns are also definite.

5. Personal pronouns such as "he", "I" and "you" are inherently definite.

6. Objects of vocation: "O car!"

7. A noun which is possessive to any of the above: "Zahid's car"

8. Special category: أَللّٰهُ (*ʾal-lāhu* ) is another instance of definite lexeme.

The last category confirms that definiteness must be specifiable at the lexeme level. Thus if the state of a noun is definite, the noun lexeme contains *yes* as the value of DEF, otherwise its value will be *no*.

In Arabic, there is a significant role of the definiteness (DEF) feature for syntactic agreement. A noun and its modifier must agree on the value of the DEF feature. For example, اَلكِتَابُ الأَحمَرُ (*alkitābu 'l-ʾaḥmaru* ) means "the red book". Here, اَلكِتَابُ (*alkitābu* ) means "the book" and أَحمَرُ (*ʾaḥmaru* ) means "red". As "red" is used as a modifier for "the book", the definiteness prefix 'al' has been added to أَحمَرُ (*ʾaḥmaru* ) yielding اَلأَحمَرُ (*al-ʾaḥmaru* ).

Like SBCG in English, SEM feature in Arabic contains two function features - INDEX and FRAMES. The INDEX is used for index based semantic agreement and FRAMES contains the list of frames which contain semantic information.

Throughout this whole formalism, we use the event frame for verb and verbal nouns to capture their semantic content efficiently. This event frame takes an event or situational index variable (SIT) and index-valued features such as actor, undergoer, instrument, location. In case of *write-fr*, this event frame contains three indices: action or event (SIT), actor (ACTOR) and undergoer of the action (UNDGR) i.e. the object of the verb.

We use this index based agreement [40] as opposed to putting the agreements under AGR feature [26]. This is because index based agreement is more customary in HPSG and most of the scholars use index based agreement.

Depending on languages, agreement may have gender, human/non-human, animate/inanimate or shape features [40]. In Arabic, person, number, gender and human/nonhuman - these information must be kept for semantic agreement. So, INDEX feature is composed of PERSON, NUMBER, GENDER and HUM and it is contained under SEM. Notably, the first three features (PERSON, NUMBER and GENDER) are also used for semantic agreement in English [40].

We introduce HUM feature for Arabic which denotes humanness. In Arabic, Humanness is a crucial grammatical factor for predicting certain kinds of plural formation and for the purpose of agreement with other components of a phrase or clause within a sentence. The grammatical criterion of humanness only applies to nouns in the plural forms. As an example, consider the sentences, "these boys are intelligent" (هَؤُلَاْء الَاولَادُ أَذكِيَاْء - *haʾulāʾ alāwlādu ʾaḏkiyāʾ* ) and "these birds are intelligent" (هَذِهِ الطُيورُ ذَكِيَّةٌ - *haḏihi ʾlṭuywru ḏakiyyatun* ). Both of these sentences are plural. But the former refers to human beings whereas the latter refers to non-humans. So the same word "intelligent" (*ḏakiyyun* ) has taken two different plural forms in two sentences, namely, أَذكِيَاْء (*ʾaḏkiyāʾ* ) and ذَكِيَّةٌ (*ḏakiyyatun* ), respectively. In case of boys, it is in the third person masculine plural form

( أَذْكِيَاء - *ʾaḏkiyāʾ* ) whereas in case of birds, it is in the third person feminine singular form ( ذَكِيَّةٌ - *ḏakiyyatun* ). Also, note that from the third person feminine singular form ( ذَكِيَّةٌ - *ḏakiyyatun* ), we cannot readily say that it refers to feminine. In fact, it may refer plural of nonhuman beings too. This is why, along with PERSON, NUMBER and GENDER, we keep HUM as a semantic agreement feature.

If the noun refers to a human being then the value of HUM is *yes*, otherwise it is *no*. The value of PERSON for Arabic nouns can be $1st$, $2nd$ or $3rd$. There are three number values in Arabic. So, the value of NUMBER can be *sg*, *dual* or *pl* denoting singular, dual or plural, respectively. The GENDER feature contains either *masc* or $fem$ denoting masculine and feminine, respectively. It should be noted that there is no neutral gender in Arabic.

A complete example of noun نَاصِرٌ (*nāṣirun* ) is shown in Figure 3.8. Type of this AVM is $triptote - sound - sg - noun - lex$.

MORPH contains ROOT, STEM and DEC. ROOT contains root letters i.e. $n, s, r$. FORM contains two morphemes: *naasir* and $-un$. DEC contains $T_n1$ denoting declension type 1. It's CASE is *nom* denoting nominative and DEF is *no* that is indefinite. Values of SELECT, XARG, LID are *none*. As it is detached noun, it's ARG-ST (argument structure) and VAL (valance) are empty. FRAMES contains a frame of type $help - fr$.

### 3.4.2 AVM for Arabic Verb

We modify the verb AVM proposed by Bhuyan et al. [2], particularly the INDEX feature. We try to align the design of the verb AVM with that of the noun AVM. Figure 3.9 shows the SBCG AVM of an Arabic verb.

The MORPH feature in the verb AVM is similar to that in the noun AVM except for the VDEC feature. It captures the declension type of verbs and it replaces the DEC feature of the noun AVM which captures the declension type of nouns. Like DEC, it

$$
\begin{bmatrix}
\textit{triptote-sound-sg-naasirun-lex} & \\
\textsc{morph} & \begin{bmatrix} \textsc{root} & \left\langle n,s,r \right\rangle \\ \textsc{skeleton} & \left\langle \textit{naasir} + \textit{un} \right\rangle \\ \textsc{dec} & T_n 1 \end{bmatrix} \\
\textsc{arg-st} & [\,] \\
\textsc{syn} & \begin{bmatrix} \textsc{cat} & \begin{bmatrix} \textit{noun} & \\ \textsc{case} & \textit{nom} \\ \textsc{def} & \textit{no} \\ \textsc{select} & \textit{none} \\ \textsc{xarg} & \textit{none} \\ \textsc{lid} & \textit{none} \end{bmatrix} \\ \textsc{val} & [\,] \\ \textsc{mrkg} & \textit{unmk} \end{bmatrix} \\
\textsc{sem} & \begin{bmatrix} \textsc{index} & i \\ \textsc{frames} & \left\langle \begin{bmatrix} \textit{help-fr} & \\ \textsc{sit} & s \\ \textsc{actor} & i \\ \textsc{undgr} & j \end{bmatrix} \right\rangle \end{bmatrix}
\end{bmatrix}
$$

Figure 3.8: SBCG Arabic noun example: naasirun

determines how the end vowel of a verb lexeme changes to reflect the mood. As we can see from Table 2.10, there are 4 declension types for verbs. So possible values for VDEC feature are $T_v1, T_v2 \ldots T_v4$ denoting four declension types of verbs. As it is for verb, the AVM doesn't contain CASE. Rather, it contains VFORM (verbform), VOICE, MOOD. Value of VFORM can be *perf* or *imperf* to denote perfect and imperfect verb. Value of VOICE can be *active* or *passive*. Value of MOOD can be any of three moods in Arabic i.e. *subjunctive*, *indicative* or *jussive*.

The SEM feature in this AVM is the same as it is in the SBCG English verb AVM. SIT-INDEX, i.e., situation index is used for index based semantic agreement. SBCG does not show any distinction between INDEX and SIT-INDEX. Also, it does not show the feature description of SIT-INDEX. We put it as a function feature but currently it has only one atomic attribute. This attribute is SITUATION. It contains the name of the verb. This SIT-INDEX is used in event-frames of the verb and the verbal noun lexemes. Thus, ultimately it is very similar to Davidsonian event variable [17]. Like AVM for noun, FRAMES contains the list of frames which contain semantic information in Minimal Recursion Semantics (MRS). These frames contain indices of both INDEX and SIT-INDEX. A sample AVM of verb كَتَبَ (*kataba* ) is shown in Figure 3.10.

## 3.5 Construction Rules to Capture Condition of Declension

In Table 2.7 we have seen different types of declension follows different case markings. Section 3.2 shows different types of lexemes follows different declension types. Based on the mapping, for a particular type of lexeme, we develop construction rules to construct accusative and genitive from nominative lexemes. For simplicity, in this thesis we have analyzed singular noun classes. So our construction rules will not cover any plural noun classes i.e. class 4 which follows declension type 2. In this section, we show sample con-

$$
\begin{bmatrix}
\textit{verb-lex} & \\
\textsc{phon} & [\,] \\
\textsc{morph} & \begin{bmatrix} \textsc{root} & list(letter) \\ \textsc{skeleton} & list(letter) \\ \textsc{vdec} & list(letter) \end{bmatrix} \\
\textsc{arg-st} & list(sign) \\
\textsc{syn} & \begin{bmatrix} \textsc{cat} & \begin{bmatrix} \textit{verb} & \\ & \\ \textsc{vform} & \ldots \\ \textsc{voice} & \ldots \\ \textsc{mood} & \ldots \\ \textsc{select} & \ldots \\ \textsc{xarg} & \ldots \\ \textsc{lid} & \ldots \end{bmatrix} \\ \textsc{val} & list(sign) \\ \textsc{mrkg} & mrk \end{bmatrix} \\
\textsc{sem} & \begin{bmatrix} \textsc{sit-index} & \begin{bmatrix} \textsc{situation} & \ldots \end{bmatrix} \\ \textsc{frames} & list(frame) \end{bmatrix}
\end{bmatrix}
$$

Figure 3.9: AVM for Arabic verb

```
[ kataba-form-IA-triliteral-sound-active-perfect-3rd-sg-masc-verb-lex
  PHON      []
  MORPH     [ ROOT      ⟨ k, t, b ⟩
              SKELETON  ⟨ k, a, t, a, b, a ⟩
              VDEC      T_v1 ]
  ARG-ST    ⟨ [1] [ SYN [ CAT [ noun
                                CASE  accusative
                                OPT   − ] ]
                    SEM [ INDEX j ] ] ⟩
  SYN       [ CAT   [ verb
                      VFORM   perfect
                      VOICE   active
                      MOOD    indicative
                      SELECT  none
                      XARG    none
                      LID     none ]
              VAL   ⟨ [1] ⟩
              MRKG  none ]
  SEM       [ SIT-INDEX  [s] [ SITUATION writing ]
              FRAMES     ⟨ [ write-fr
                             SIT         [s]
                             ACTOR       [i] [ PERSON  3rd
                                               NUMBER  sg
                                               GENDER  masc
                                               HUM     yes ]
                             UNDGR       j
                             LOCATION    k
                             INSTRUMENT  l ] ⟩ ] ]
```

Figure 3.10: AVM for a sample root verb - kaataba

struction rules for nouns and verbs. For nouns, we show construction rules for declension type 1 and 3 and for verbs we show construction rules for declension type 1.

### 3.5.1 Construction by $T_n1$ Declension

In this section, we show construction rules to construct accusative and genitive lexemes from nominatives by $T_n1$ declension.

**Accusative Construction by $T_n1$**

A construction rule to construct accusatives from nominative lexemes is shown in Figure 3.11 following $T_n1$ declension. Here MTR (mother) is in accusative case and DTRS (daughter) contains only one lexeme which is in nominative case. MTR (accusative) is formed from DTRS (nominative) using $T_n1$ declension.

As in Table 2.7, case marking of nominative is ٌ (*un* ). But for accusative, case marking of is ً (*an* ). Our construction rule captures this change by changing SKELETON feature under MORPH (morphology) from DTRS to MTR. SKELETON of MTR is ended with $-an$ but SKELETON of DTRS lexeme is ended with $-un$. Another change we can identify in CASE under CAT (category) to denote the case of the lexemes. All other features are same for MTR and DTRS.

An example of this construction rule is shown in is shown in Figure 3.12 where كَاتِبًا (*kātiban* ) is constructed from كَاتِبٌ (*kātibun* ) . كَاتِبٌ (*kātibun* ) is in nominative case and ended with ٌ (*un* ). But in accusative case, it is modified to كَاتِبًا (*kātiban* ) which is ended with ً (*an* ).

$$
\begin{bmatrix}
\textit{triptote} - \textit{sound} - \textit{sg} - \textit{acc} - T_n1 - \textit{noun} - \textit{cxt} & \\
\text{MTR} & \begin{bmatrix}
\textit{triptote} - \textit{sound} - \textit{sg} - \textit{noun} - \textit{lex} & \\
\text{MORPH} & \begin{bmatrix} \text{SKELETON} & \left\langle \boxed{7} + \textit{an} \right\rangle \\ \text{ROOT} & \boxed{6} \\ \text{DEC} & T_n1 \end{bmatrix} \\
\text{ARG-ST} & \boxed{5} \\
\text{SYN} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \textit{noun} & \\ \text{CASE} & \textit{acc} \\ \text{DEF} & \boxed{4} \end{bmatrix} \\ \text{VAL} & \boxed{3} \\ \text{MRKG} & \boxed{2} \end{bmatrix} \\
\text{SEM} & \boxed{1}
\end{bmatrix} \\
\text{DTRS} & \left\langle \begin{bmatrix}
\textit{triptote} - \textit{sound} - \textit{sg} - \textit{noun} - \textit{lex} & \\
\text{MORPH} & \begin{bmatrix} \text{SKELETON} & \left\langle \boxed{7} + \textit{un} \right\rangle \\ \text{ROOT} & \boxed{6} \\ \text{DEC} & T_n1 \end{bmatrix} \\
\text{ARG-ST} & \boxed{5} \\
\text{SYN} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \textit{noun} & \\ \text{CASE} & \textit{nom} \\ \text{DEF} & \boxed{4} \end{bmatrix} \\ \text{VAL} & \boxed{3} \\ \text{MRKG} & \boxed{2} \end{bmatrix} \\
\text{SEM} & \boxed{1}
\end{bmatrix} \right\rangle
\end{bmatrix}
$$

Figure 3.11: Lexical rule for accusative construction using $T_n1$ construct

$$
\begin{bmatrix}
triptote - sound - sg - acc - T_n1 - noun - cxt & \\
\text{MTR} & \begin{bmatrix}
triptote - sound - sg - noun - lex & \\
\text{MORPH} & \begin{bmatrix}
\text{SKELETON} & \left\langle \boxed{7} + an \right\rangle \\
\text{ROOT} & \boxed{6} \\
\text{DEC} & T_n1
\end{bmatrix} \\
\text{ARG-ST} & \boxed{5}\,\langle\rangle \\
\text{SYN} & \begin{bmatrix}
\text{CAT} & \begin{bmatrix}
noun & \\
\text{CASE} & acc \\
\text{DEF} & \boxed{4}
\end{bmatrix} \\
\text{VAL} & \boxed{3}\,\langle\rangle \\
\text{MRKG} & \boxed{2}
\end{bmatrix} \\
\text{SEM} & \boxed{1}
\end{bmatrix} \\
\text{DTRS} & \left\langle \begin{bmatrix}
triptote - sound - sg - noun - lex & \\
\text{MORPH} & \begin{bmatrix}
\text{SKELETON} & \left\langle \boxed{7}\ kaatib + un \right\rangle \\
\text{ROOT} & \boxed{6} \left\langle k,t,b \right\rangle \\
\text{DEC} & T_n1
\end{bmatrix} \\
\text{ARG-ST} & \boxed{5}\,\langle\rangle \\
\text{SYN} & \begin{bmatrix}
\text{CAT} & \begin{bmatrix}
noun & \\
\text{CASE} & nom \\
\text{DEF} & \boxed{4}\ no
\end{bmatrix} \\
\text{VAL} & \boxed{3}\,\langle\rangle \\
\text{MRKG} & \boxed{2}\ unmk
\end{bmatrix} \\
\text{SEM} & \boxed{1} \begin{bmatrix}
\text{INDEX} & i \\
\text{FRAMES} & \left\langle \begin{bmatrix}
help\text{-}fr & \\
\text{SIT} & s \\
\text{ACTOR} & i \\
\text{UNDGR} & j
\end{bmatrix} \right\rangle
\end{bmatrix}
\end{bmatrix} \right\rangle
\end{bmatrix}
$$

Figure 3.12: Example of lexical rule for accusative construction by $T_n1$ declension

#### Genitive Construction by $T_n1$

Genitive lexeme construction is same as accusative construction. The only difference is in SKELETON under MORPH and CASE under CAT. As in Table 2.7, for $T_n1$ declension nominative case marking is *-un* whereas genitive case marking is *-in*. To capture this, FORM of DTRS is ended with $-un$ and FORM of MTR is ended with $-an$. The construction rule is shown in Figure 3.13.

### 3.5.2 Construction by $T_n3$ Declension

In this section we show construction of accusative and genitive lexemes from nominative lexemes by $T_n3$ declension.

#### Accusative Construction by $T_n3$

Table 2.7 shows, for $T_n3$ declension nominative case marking is ـُ ($u$ ) whereas accusative case marking is ـَ ($a$ ). To capture this, SKELETON of DTRS is ended with $-u$ and SKELETON of MTR is ended with $-a$. Figure 3.14 shows the construct rule. Here we can see significance change in SKELETON and also CASE under CAT.

#### Genitive Construction by $T_n3$

Table 2.7 shows, for $T_n3$ declension, the case marking for nominative is ـُ ($u$ ) but genitive and accusative case marking is same which is ـَ ($a$ ). So construction rule for genitives is same as accusatives for $T_n3$ declension. Figure 3.15 shows construction rule for geni-

$$
\begin{bmatrix}
\textit{triptote}-\textit{sound}-\textit{sg}-\textit{gen}-T_n1-\textit{noun}-\textit{cxt} \\
\text{MTR} & \begin{bmatrix}
\textit{triptote}-\textit{sound}-\textit{sg}-\textit{noun}-\textit{lex} \\
\text{MORPH} & \begin{bmatrix} \text{SKELETON} & \left\langle \boxed{7} + \textit{in} \right\rangle \\ \text{ROOT} & \boxed{6} \\ \text{DEC} & T_n1 \end{bmatrix} \\
\text{ARG-ST} & \boxed{5} \\
\text{SYN} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \textit{noun} \\ \text{CASE} & \textit{gen} \\ \text{DEF} & \boxed{4} \end{bmatrix} \\ \text{VAL} & \boxed{3} \\ \text{MRKG} & \boxed{2} \end{bmatrix} \\
\text{SEM} & \boxed{1}
\end{bmatrix} \\
\text{DTRS} & \left\langle \begin{bmatrix}
\textit{triptote}-\textit{sound}-\textit{sg}-\textit{noun}-\textit{lex} \\
\text{MORPH} & \begin{bmatrix} \text{SKELETON} & \left\langle \boxed{7} + \textit{un} \right\rangle \\ \text{ROOT} & \boxed{6} \\ \text{DEC} & T_n1 \end{bmatrix} \\
\text{ARG-ST} & \boxed{5} \\
\text{SYN} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \textit{noun} \\ \text{CASE} & \textit{nom} \\ \text{DEF} & \boxed{4} \end{bmatrix} \\ \text{VAL} & \boxed{3} \\ \text{MRKG} & \boxed{2} \end{bmatrix} \\
\text{SEM} & \boxed{1}
\end{bmatrix} \right\rangle
\end{bmatrix}
$$

Figure 3.13: Lexical rule for genitive construction by $T_n1$ construction

$$
\begin{bmatrix}
\textit{diptote}-\textit{acc}-T_n3-\textit{noun}-\textit{cxt} & \\
\textsc{mtr} & \begin{bmatrix}
\textit{diptote}-\textit{noun}-\textit{lex} & \\
\textsc{morph} & \begin{bmatrix} \text{SKELETON} & \left\langle \boxed{7} + a \right\rangle \\ \textsc{root} & \boxed{6} \\ \textsc{dec} & T_n3 \end{bmatrix} \\
\textsc{arg-st} & \boxed{5} \\
\textsc{syn} & \begin{bmatrix} \textsc{cat} & \begin{bmatrix} \textit{noun} & \\ \textsc{case} & \textit{acc} \\ \textsc{def} & \boxed{4} \end{bmatrix} \\ \textsc{val} & \boxed{3} \\ \textsc{mrkg} & \boxed{2} \end{bmatrix} \\
\textsc{sem} & \boxed{1}
\end{bmatrix} \\
\textsc{dtrs} & \left\langle \begin{bmatrix}
\textit{diptote}-\textit{noun}-\textit{lex} & \\
\textsc{morph} & \begin{bmatrix} \text{SKELETON} & \left\langle \boxed{7} + u \right\rangle \\ \textsc{root} & \boxed{6} \\ \textsc{dec} & T_n3 \end{bmatrix} \\
\textsc{arg-st} & \boxed{5} \\
\textsc{syn} & \begin{bmatrix} \textsc{cat} & \begin{bmatrix} \textit{noun} & \\ \textsc{case} & \textit{nom} \\ \textsc{def} & \boxed{4} \end{bmatrix} \\ \textsc{val} & \boxed{3} \\ \textsc{mrkg} & \boxed{2} \end{bmatrix} \\
\textsc{sem} & \boxed{1}
\end{bmatrix} \right\rangle
\end{bmatrix}
$$

Figure 3.14: Lexical rule for accusative construction by $T_n3$ declension

tive construction by $T_n3$. Difference between Figure 3.14 and Figure 3.15 is only in CASE.

$$
\begin{bmatrix}
\textit{diptote} - \textit{gen} - T_n3 - \textit{noun} - \textit{cxt} & \\
\text{MTR} & \begin{bmatrix}
\textit{diptote} - \textit{noun} - \textit{lex} & \\
\text{MORPH} & \begin{bmatrix} \text{SKELETON} & \left\langle \boxed{7} + a \right\rangle \\ \text{ROOT} & \boxed{6} \\ \text{DEC} & T_n3 \end{bmatrix} \\
\text{ARG-ST} & \boxed{5} \\
\text{SYN} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \textit{noun} & \\ \text{CASE} & \textit{gen} \\ \text{DEF} & \boxed{4} \end{bmatrix} \\ \text{VAL} & \boxed{3} \\ \text{MRKG} & \boxed{2} \end{bmatrix} \\
\text{SEM} & \boxed{1}
\end{bmatrix} \\
\text{DTRS} & \left\langle \begin{bmatrix}
\textit{diptote} - \textit{noun} - \textit{lex} & \\
\text{MORPH} & \begin{bmatrix} \text{SKELETON} & \left\langle \boxed{7} + u \right\rangle \\ \text{ROOT} & \boxed{6} \\ \text{DEC} & T_n3 \end{bmatrix} \\
\text{ARG-ST} & \boxed{5} \\
\text{SYN} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \textit{noun} & \\ \text{CASE} & \textit{nom} \\ \text{DEF} & \boxed{4} \end{bmatrix} \\ \text{VAL} & \boxed{3} \\ \text{MRKG} & \boxed{2} \end{bmatrix} \\
\text{SEM} & \boxed{1}
\end{bmatrix} \right\rangle
\end{bmatrix}
$$

Figure 3.15: Lexical rule for genitive construction by $T_n3$ declension

### 3.5.3 Construction by $T_v1$ Declension

#### Subjunctive Construction by $T_v1$

Figure 3.16 shows construction rule to construct subjunctives from indicatives. Major difference between MTR and DTRS is in SKELETON and MOOD. From Table 2.10, we can see for declension type 1, the case marking of indicatives is

Here we show construction rules for verbal declension using $T_v1$ declension. ـُ ($u$ ) but the case marking of subjunctives is ـَ ($a$ ). So SKELETON of DTRS ended with $-u$ whereas SKELETON of MTR is ended with $-a$.

**Jussive Construction by $T_v1$**

Figure 3.17 shows jussive construction from indicative. From Table 2.10, for declension type 1, the case marking of indicatives is ـُ ($u$ ) but the case marking of subjunctives is *jawazim*. So SKELETON of DTRS ended with $-u$ whereas affix $-u$ is removed from the SKELETON of MTR.

## 3.6 Construction Rules for Definiteness

We now show construction rules to capture definiteness. In section 3.4, we have seen, an Arabic lexeme is made definite by adding أَل ($al$ ) affix. Also nunation is removed from definite lexeme construction. That is, if an indefinite lexeme ends with ـٌ ($un$ ) then a definite lexeme ends with ـُ ($u$ ). Figure 3.18 shows definite construction rule for nominative case. DTRS lexeme is indefinite and MTR is definite. Here we can see significant change in SKELETON. In DTRS, SKELETON is ended with $-un$ where is in MTR is prefixed with $-al$ and nunation is removed i.e. ended with $-u$.

Construction rule to capture definiteness of accusative and genitive cases is same as nominatives with changes in SKELETON.

## 3.7 Summary

This chapter has depicted our contributions for HPSG formalism of our research of Arabic declension. We have presented HPSG type hierarchy for both nouns and verbs after analyzing their dimensions. We have mapped lexical types to declension types for both

$$
\begin{bmatrix}
\textit{sound}-\textit{sg}-\textit{subjunctive}-T_v1-\textit{verb}-\textit{cxt} & \\
\text{MTR} & \begin{bmatrix}
\textit{sound}-\textit{sg}-\textit{verb}-\textit{lex} & \\
\text{MORPH} & \begin{bmatrix} \text{SKELETON} & \left\langle \boxed{11} + a \right\rangle \\ \text{ROOT} & \boxed{10} \\ \text{VDEC} & T_v1 \end{bmatrix} \\
\text{ARG-ST} & \boxed{9} \\
\text{SYN} & \begin{bmatrix}
\text{CAT} & \begin{bmatrix} \textit{verb} & \\ \text{VFORM} & \boxed{8} \\ \text{VOICE} & \boxed{7} \\ \text{MOOD} & \textit{subjunctive} \\ \text{SELECT} & \boxed{6} \\ \text{XARG} & \boxed{5} \\ \text{LID} & \boxed{4} \end{bmatrix} \\
\text{VAL} & \boxed{3} \\
\text{MRKG} & \boxed{2}
\end{bmatrix} \\
\text{SEM} & \boxed{1}
\end{bmatrix} \\
\text{DTRS} & \left\langle \begin{bmatrix}
\textit{sound}-\textit{sg}-\textit{verb}-\textit{lex} & \\
\text{MORPH} & \begin{bmatrix} \text{SKELETON} & \left\langle \boxed{11} + u \right\rangle \\ \text{ROOT} & \boxed{10} \\ \text{VDEC} & T_v1 \end{bmatrix} \\
\text{ARG-ST} & \boxed{9} \\
\text{SYN} & \begin{bmatrix}
\text{CAT} & \begin{bmatrix} \textit{verb} & \\ \text{VFORM} & \boxed{8} \\ \text{VOICE} & \boxed{7} \\ \text{MOOD} & \textit{indicative} \\ \text{SELECT} & \boxed{6} \\ \text{XARG} & \boxed{5} \\ \text{LID} & \boxed{4} \end{bmatrix} \\
\text{VAL} & \boxed{3} \\
\text{MRKG} & \boxed{2}
\end{bmatrix} \\
\text{SEM} & \boxed{1}
\end{bmatrix} \right\rangle
\end{bmatrix}
$$

Figure 3.16: Lexical rule for subjunctive construction by $T_v1$ declension

$$
\begin{bmatrix}
\textit{sound}-\textit{sg}-\textit{jussive}-T_v1-\textit{verb}-\textit{cxt} & \\
\text{MTR} & \begin{bmatrix}
\textit{sound}-\textit{sg}-\textit{verb}-\textit{lex} & \\
\text{MORPH} & \begin{bmatrix} \text{SKELETON} & \left\langle \boxed{11} \right\rangle \\ \text{ROOT} & \boxed{10} \\ \text{VDEC} & T_v1 \end{bmatrix} \\
\text{ARG-ST} & \boxed{9} \\
\text{SYN} & \begin{bmatrix}
\text{CAT} & \begin{bmatrix} \textit{verb} & \\ \text{VFORM} & \boxed{8} \\ \text{VOICE} & \boxed{7} \\ \text{MOOD} & \textit{jussive} \\ \text{SELECT} & \boxed{6} \\ \text{XARG} & \boxed{5} \\ \text{LID} & \boxed{4} \end{bmatrix} \\
\text{VAL} & \boxed{3} \\
\text{MRKG} & \boxed{2}
\end{bmatrix} \\
\text{SEM} & \boxed{1}
\end{bmatrix} \\
\text{DTRS} & \left\langle \begin{bmatrix}
\textit{sound}-\textit{sg}-\textit{verb}-\textit{lex} & \\
\text{MORPH} & \begin{bmatrix} \text{SKELETON} & \left\langle \boxed{11} + u \right\rangle \\ \text{ROOT} & \boxed{10} \\ \text{VDEC} & T_v1 \end{bmatrix} \\
\text{ARG-ST} & \boxed{9} \\
\text{SYN} & \begin{bmatrix}
\text{CAT} & \begin{bmatrix} \textit{verb} & \\ \text{VFORM} & \boxed{8} \\ \text{VOICE} & \boxed{7} \\ \text{MOOD} & \textit{indicative} \\ \text{SELECT} & \boxed{6} \\ \text{XARG} & \boxed{5} \\ \text{LID} & \boxed{4} \end{bmatrix} \\
\text{VAL} & \boxed{3} \\
\text{MRKG} & \boxed{2}
\end{bmatrix} \\
\text{SEM} & \boxed{1}
\end{bmatrix} \right\rangle
\end{bmatrix}
$$

Figure 3.17: Lexical rule for jussive construction by $T_v1$ declension

$$
\begin{bmatrix}
triptote - sound - sg - nom - def - noun - cxt \\
\textsc{mtr} & \begin{bmatrix}
triptote - sound - sg - noun - lex \\
\textsc{morph} & \begin{bmatrix} \textsc{skeleton} & \left\langle al + \boxed{7} + u \right\rangle \\ \textsc{root} & \boxed{6} \end{bmatrix} \\
\textsc{arg-st} & \boxed{5} \\
\textsc{syn} & \begin{bmatrix} \textsc{cat} & \begin{bmatrix} noun \\ \textsc{case} & \boxed{4} \\ \textsc{def} & yes \end{bmatrix} \\ \textsc{val} & \boxed{3} \\ \textsc{mrkg} & \boxed{2} \end{bmatrix} \\
\textsc{sem} & \boxed{1}
\end{bmatrix} \\
\textsc{dtrs} & \left\langle \begin{bmatrix}
triptote - sound - sg - noun - lex \\
\textsc{morph} & \begin{bmatrix} \textsc{skeleton} & \left\langle \boxed{7} + un \right\rangle \\ \textsc{root} & \boxed{6} \end{bmatrix} \\
\textsc{arg-st} & \boxed{5} \\
\textsc{syn} & \begin{bmatrix} \textsc{cat} & \begin{bmatrix} noun \\ \textsc{case} & \boxed{4}\ nom \\ \textsc{def} & no \end{bmatrix} \\ \textsc{val} & \boxed{3} \\ \textsc{mrkg} & \boxed{2} \end{bmatrix} \\
\textsc{sem} & \boxed{1}
\end{bmatrix} \right\rangle
\end{bmatrix}
$$

Figure 3.18: Lexical rule for nominative definite construction

nouns and verbs. This mapping is necessary for capturing declension phenomenon in HPSG. We have also proposed algorithm to find out declension types of nouns easily. We have proposed HPSG AVM for nouns and verbs. We have proposed construction rules to construct accusative and genitive lexemes from nominatives and subjunctive and jussive lexemes from indicatives. Lastly, we have proposed construction rule for construction of definite from indefinite lexemes. In next chapter, we will show implementation methodologies for implementing our HPSG formalism i.e. type hierarchy, signs and construction rules.

# Chapter 4

# Implementation

In this chapter, we show implementation methodologies of HPSG formalism which is proposed in previous chapter. Section 4.1 describes possible choices of implementation platforms followed by TRALE introduction. Section 4.2 discusses TRALE basics by describing its basic components. Section 4.3 provides methodologies that we follow to implement proposed HPSG formalism in TRALE. We give some sample input and output. Detail input files are given in Appendix B.

## 4.1 Introduction to TRALE

HPSG can be implemented in several grammar development systems i.e. TFS, CUF, ALE, ALEP, PAGE, ProFIT, CL-ONE and ConTroll. Among these, ConTroll ( [19]) and ALE ( [10, 37]) show better behaviour than the others considering type system, syntax and feature terms and other computational aspects ( [7]). Both of these compilers are designed based on the formalism of HPSG'87 ( [39]).

ALE is an integrated phrase structure parsing and definite clause logic programming system in which the terms are typed feature structures. Typed feature structures combine type inheritance and appropriateness specifications for features and their values.

At the term level, ALE has variables, types, feature value restrictions, path equations, inequations, general constraints, and disjunction. The definite clause programs allow disjunction, negation and cut, specified with Prolog syntax. For Parsing, ALE compiles from the grammar specification a Prolog-optimized bottom-up, dynamic chart parser. Definite clauses are also compiled into Prolog.

TRALE is an extension of ALE that supports extra functionality i.e. complex-antecedent constraints which adds extra constraints to grammars in order to enforce the view of subtyping standardly assumed in HPSG. TRALE is implemented as a preprocessor for ALE that intercepts certain grammar clauses at compile-time to generate extra code for the ALE compiler ( [11]).

We use TRALE on Grammix operation system version of June, 2007 [35]. There is another new version of TRALE which is not complete but can be run stand alone on Linux platform. This new version was published on 2008 [41]. Grammix is developed for grammar development and contains two complete grammar development systems - TRALE and LKB.

In [33], author has explored to two leading implementation platforms for implementing HPSG grammars. One is Linguistic Knowledge Building (LKB) system ( [14]) is developed not particularly for implementing HPSG grammars, but rather, as a framework independent environment for typed feature structures grammar. On the other hand, TRALE ( [41]), an extension of the Attribute Logic Engine (ALE) system, is a grammar implementation platform that was developed as part of the MiLCA project ( [34]), specifically for the implementation of theoretical HPSG grammars. Considering these, we have decided to use TRALE for implementation of our grammar.

## 4.2 TRALE Basics

In this section, we discuss about TRALE preliminaries, particularly TRALE components and implementation procedure. This is helpful to understand our implementation methodologies.

### 4.2.1 Signature File

To capture type hierarchy of a grammar, TRALE reads a special signature file, which follows a specific format. Signature file should be a separate text file where subtyping indicated by indentation.

Signature file starts with *type_hierarchy* which indicates type hierarchy. Type hierarchy begins with most general element *bot*. Sub-types of a type are shown by adding indentation with the indentation of parent type.

For each type there may be zero or more features. Feature and value is separated by colon (:) that is, *feature:value*. All feature value pairs are separated by white-space. An example is shown in Figure 4.2.1. The example shows a type hierarchy with a most general element *bot*, which immediately subsumes types $a$ and *bool*. Type $a$ introduces two features $F$ and $G$ whose values must be of type *bool*. Type $a$ also subsumes two other types $b$ and $c$. The value of $F$ of the former is always *plus* and value of $G$ is always *minus*. The feature values of type $c$ are the inverse of type $b$. Finally, *bool* has two subtypes *plus* and *minus*.

A type can occur as a subtype of two or more different super types which is called multiple inheritance. In this case, one prefixes the subtype with an ampersand (&) to indicate that the multiple inheritance is intended. Lastly a single period (.) in an otherwise blank line signals the end of a signature declaration.

```
type_hierarchy
bot
 a f:bool g:bool
  b f:plus g:minus
  c f:minus g:plus
bool
 plus
 minus
```

Figure 4.1: Example of signature file

### 4.2.2 Lexical Rule Compiler

In TRALE, *theory.pl* file works as a mother file which is used to load all other files. For lexical entries and rules separate file can be used or can be written in *theory.pl*. If we use more than one file for declaring our grammar (i.e. *signature* file, *theory.pl*) then *multifile* declaration needs to be used at top of *theory.pl*.

#### Lexical Rule Depth Bound

ALE lexical rules are productive because here lexical rules are applied sequentially to their own output or the output of other lexical rules. Thus, it is possible to derive the nominal *writer* from the verb *write*, and then to derive the plural nominal *writers* from *writer*, and so on. At the same time, the lexical system is leashed to a fixed depth-bound, which may be specified by the user. This bound limits the number of rules that can be applied to any given category. The bound on application of rules is defined by *lex_rule_depth*, that is, in *theory.pl* we need to write like this:

```
:-lex_rule_depth(2).
```

### Loading Signature File

Signature file is loaded by calling *signature* like following:

```
signature(signature).
```

### Chart Display

TRALE can show AVMs of signs or construction rules in standard output which is called *Grisu*. By Grisu an AVM and its structure sharing is clearly visible.

### Feature Ordering

To specify order in which features are displayed by the pretty printer, we need to include statement like this:

- $f <<< g$: This means $f$ will be ordered before $g$.
- $<<< h$: This means $h$ has lowest precedence and will be ordered last.
- $>>> k$: This means $k$ has highest precedence and will be ordered first.

### Lexical Entries

In TRALE lexical entries are listed by following format:

```
<word> ~~> <desc>.
```

As an example, a lexical entry for *john* is shown below. Here for *john*, SYN and SEM feature are described. SYN includes CAT and VAL feature and their values. A lexical entry is ended with a dot (.).

```
john ~~>
  (syn:
    (cat:noun,
    val:VAL),
  sem:j).
```

**Lexical Rules**

In TRALE lexical rules is specified by following format:

```
<lex_rule_name>##<lex_rewrite>
morphs <morphs>
```

Here *lex_rewrite* denotes conversion from one type to another type denoting DTRS and MTR respectively and separated by $**$ .

```
<lex_rewrite> ::= <desc> **> <desc>
```

On the other hand, *morphs* denotes morphological changes for DTRS to MTR. It is described by the following pattern:

```
<morph> ::= (<string_pattern>) becomes (<string_pattern>)
```

*string_pattern* can be atomic pattern like atom, variable or list. Here is an example of lexical rule written in TRALE.

```
plural_n##
        (n,
        num:sg)
        **>
        (n,
        num:pl)
```

```
morphs
goose becomes geese,
(X) becomes (X,s),
(X,man) becomes (X,men),
(X,ey) becomes (X,[i,e,s]).
```

In the example, singular to plural conversion is shown. Four cases has been mentioned for morphological change i.e. plural can be formed from singular by intermediate character change or adding *-s* or adding *-ies* instead of -y.

### Commands

After developing signaute and theory.pl a grammar can be compiled using following command in TRALE:

```
| ?- compile_gram(GramFile).
```

Here *GramFile* denotes the grammar file which is actually theory.pl.

A word or lexeme or phrase can be parsed using command *rec* like following:

```
| ?- rec([big,kid]).
```

This will also start GRISU output of the AVM.

## 4.3 Implementation Methodologies

In this section, we discuss about the input files that we have implemented. Later, we show verification of our AVM and construction rules.

### 4.3.1 Signature File

For implementation we have to develop signature file according to type hierarchy. Complete signature file of our research is shown in Appendix B. To match an AVM it starts with sign. Type sign has features like morph, arg_st, syn and sem. Again syn contains cat (category), val (valence) and mrkg (marking). cat contains case, def (definiteness), mood, vform (verb form) and voice. char type contains only English alphabets which are required for our lexeme formation. Possible values of case, person, number, gender, def (definiteness), hum (humanness), vform (verb form), mood, mrkg are shown. Feature sem contains index and frames.

TRALE output for type hierarchy of noun lexeme is shown in 4.2 and type hierarchy of verb lexeme is shown in 4.3.

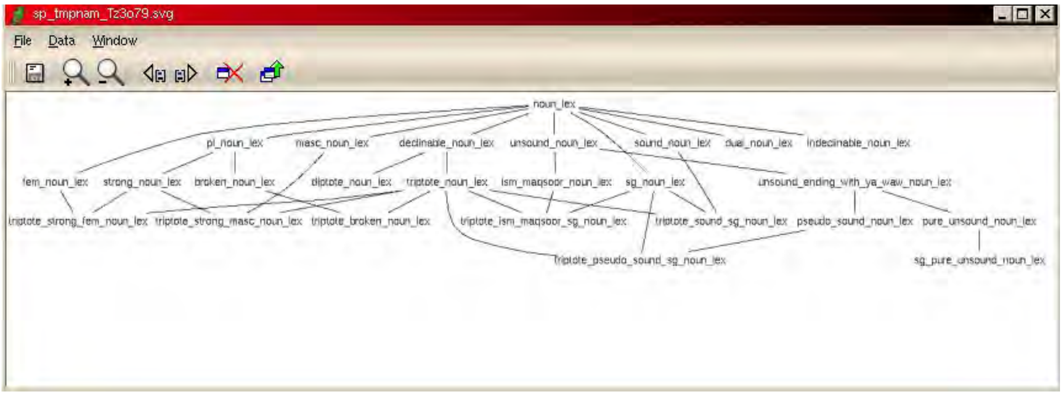


Figure 4.2: TRALE output for type hierarchy of noun-lexemes

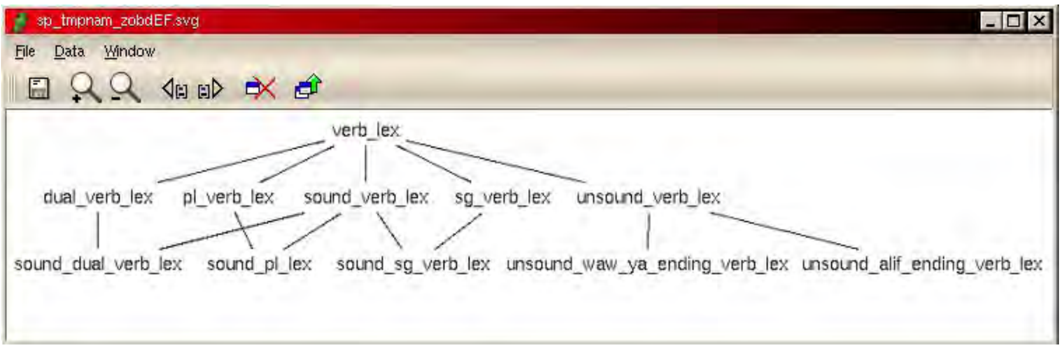


Figure 4.3: TRALE output for type hierarchy of verb-lexemes

### 4.3.2 Theory File

In this section we show how we have implemented lexical entries and corresponding lexical rules in theory.pl.

#### Preliminaries

For the implementation purpose we are using two files: signature which contains type hierarchy and theory.pl which loads all other files, lexical entries and lexical rules. As we are using two files so we have used *multifile* declaration in theory.pl.

```
%:- multifile '##'/2.
%:- multifile '~~>'/2.
```

After that we have loaded TRALE extension using following command:

```
:- [trale_home(tree_extensions)].
```

We have restrict depth of lexical rules to 4.

```
:-lex_rule_depth(4).
```

After that we have loaded our signature file.

```
signature(signature).
```

We want show MORPH at the beginning of AVM of sign. After MORPH we want to show ARG_ST, SYN, SEM. In MORPH we want to show ROOT first, then FORM and then DEC feature. In CAT (category) of verb, ordering will be VFORM, VOICE and then MOOD. In SYN, first CAT will be shown and after that VAL and lastly MRKG. In SEM, INDEX will come before FRAMES. So we have set precedence order as follows:

```
>>>morph.
morph <<< arg_st.
arg_st <<< syn.
syn <<< sem.

root <<< form.
form <<< dec.

vform <<< voice.
voice <<< mood.

cat <<< val.
val <<< mrkg.

index <<< frames.
```

### Lexical Entries

In our experiment we have given several lexical entries of verb and noun.

We have given lexical entries for several nouns. 3 examples entries have been shown in Appendix B. These entries are for كَاتِبٌ (*kātibun* ), نَاصِرٌ (*nāṣirun* ) and سَاجِدٌ (*sāǧidun* ) respectively. Each of these has 3 root letters i.e. $k, t, b$ or $n, s, r$ or $s, j, d$ respectively. As type of these AVM is *triptote_sound_sg_noun_lex*, so these will follow declension type $T_n1$. Argument structure and valance will be empty. These are in *nominative* case and indefinite. Semantic index of these is same as index of actor. These entries follow AVM of Figure 3.7.

For verb we have also given several entries. 3 examples lexical entries are shown in Appendix B. These entries are for أَكْتُبُ (*aktubu* ), أَنْسُرُ (*ansuru* ) and أَسْجُدُ (*asğudu* ) respectively. All these are *sound_sg_verb_lex* type verb (sound singular verb lexeme). So all these follow $T_v1$ declension type as discussed in Section 3.2.2. Each of these has separate root letters: $k, t, b$, $n, s, r$, $s, j, d$, respectively. All these are in *indicative* mood, verb form is $imperfect$ and in *active* voice. As all these are verbs, so semantic index will be same as SIT of FRAMES. Verb entries follow AVM of Figure 3.9.

### Lexical Construction Rules

In our implementation, we have shown several construction rules to justify our research. There are three types of rules in theory.pl file:

1. Noun construction rules: In Appendix B we have shown 2 example construction rules to implement rules of Figure 3.11 and 3.13, respectively. These rules are named as $triptote - sound - sg - acc - tn1 - noun - cxt$ and $triptote - sound - sg - gen - tn1 - noun - cxt$, respectively.

2. Verb construction rules: We have used several construction rules for verb based on moods. Verb of subjunctive or jussive moods can be constructed from indicative verbs. In Appendix B, we show two sample rules written in TRALE. Implementation of Figure 3.16 is shown first which is construction of subjunctive mood from indicative. It's type is $sound - sg - subjunctive - tv1 - verb - cxt$ Then implemen-

tation of Figure 3.17 is shown which is construction of jussive from indicative.

3. Definiteness construction rule: In Appendix B, we have shown implementation of our rule of Figure 3.18 and named as $triptote - sound - sg - def - noun - cxt$.

### Verification of Construction Rules of Nouns

We have done verification for AVM and construction rules of nouns. In our *signature* file, there is an entry for *kaatibun* which is in nominative case. We have construction rule $triptote - sound - sg - acc - tn1 - noun - cxt$ to construct accusative lexeme from nominative. Resulting accusative of *kaatibun* is *kaatiban*. We run command *rec(['kaatiban'])* then we got successful parsing result from TRALE. The screenshot for this inflected lexeme is shown in Figure 4.4. Similarly, we have tested construction *kaatibin* which is in genetive case and constructed from *kaatibun* by construction rule $triptote - sound - sg - gen - tn1 - noun - cxt$. We run command *rec(['kaatibin'])* then we got successful parsing result from TRALE as in Figure 4.5.

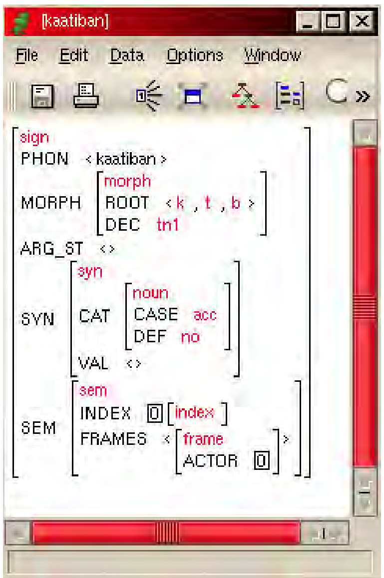


Figure 4.4: TRALE output for parsing kaatiban

As we have lexical entry for *naasirun* so TRALE can parse *naasiran* and *naasirin*. Resulting AVM from TRALE is given in Figure 4.6 and 4.7 respectively.

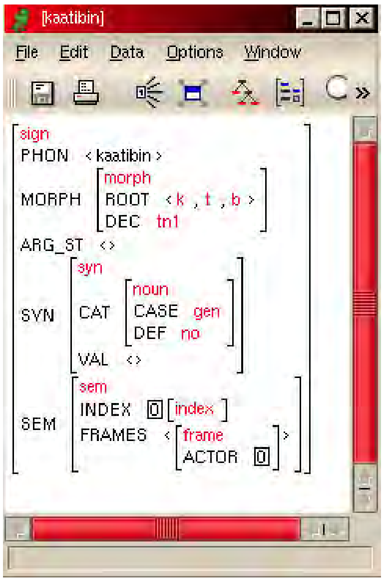


Figure 4.5: TRALE output for parsing kaatibin

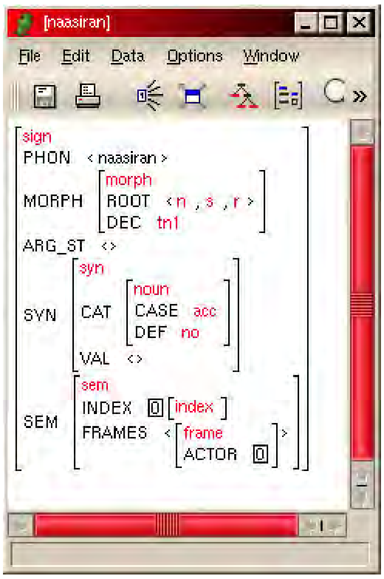


Figure 4.6: TRALE output for parsing naasiran

We have also tested construction of *saajidan* and *saajidin* from *saajidin*.

### Verification of Construction Rules of Verbs

We have tested verb construction rules. We have lexical entry for *aktubu* which is in indicative mood. By our construction rule $sound - sg - subjunctive - tv1 - verb - cxt$

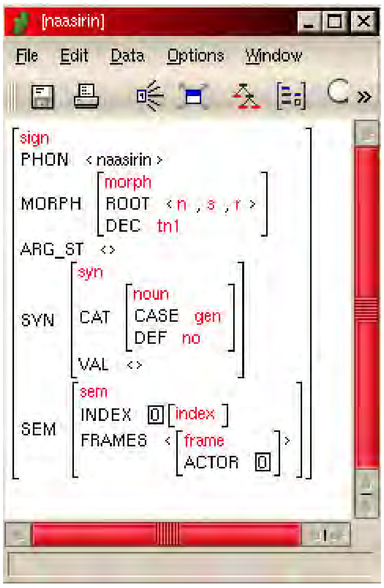


Figure 4.7: TRALE output for parsing naasirin

and $sound-sg-jussive-tv1-verb-cxt$, we can license *aktuba* and *aktub* which are in subjunctive and jussive mood respectively. So if we run *rec(['aktuba'])* and *rec(['aktub'])*, then we will get result as in Figure 4.8 and 4.9, respectively.

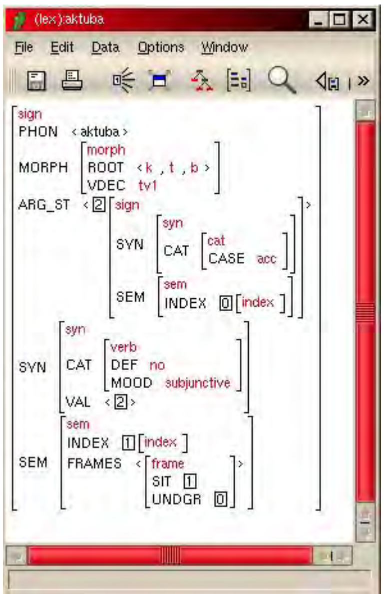


Figure 4.8: TRALE output for parsing aktuba

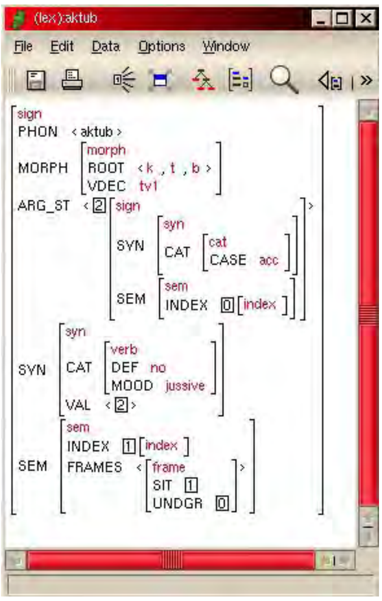


Figure 4.9: TRALE output for parsing aktub

### Verification of Construction Rules of Definiteness

We have lexical entry for *kaatibun* which is nominative indefinite lexeme. We generate *alkaatibu* from it which is nominative definite lexeme by construction rule *triptote_sound_sg_def_noun_c*
Resulting AVM is shown in Figure 4.10.

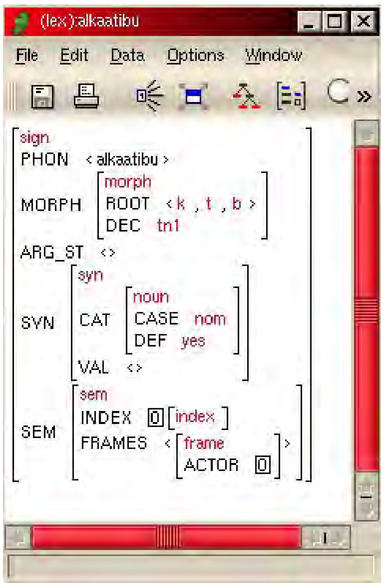


Figure 4.10: TRALE output for parsing alkaatibu

We have also tested *alkaatiba* which is accusative definite lexeme. Here 2 rules are applied successively for this generation. First, $triptote-sound-sg-acc-tn1-noun-cxt$ is used to construct *kaatiban* from *kaatibun*. Then definiteness rule $triptote - sound - sg - def - noun - cxt$ is used to construct *alkaatiba* from *kaatiban*. Resulting AVM is shown in Figure 4.11.

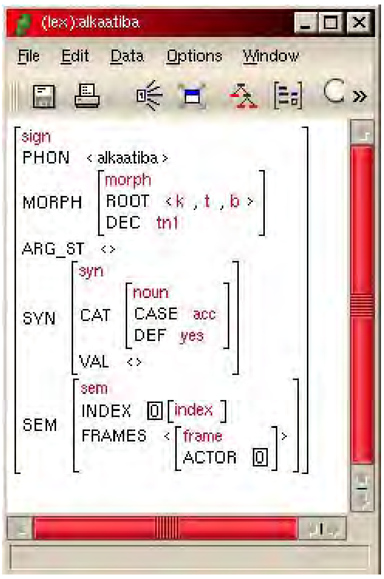


Figure 4.11: TRALE output for parsing alkaatiba

# Chapter 5

# Conclusion

This thesis is a part of a big ongoing research of HPSG modeling of Arabic nominal and verbal declension. It is mainly focused on type hierarchy to declension type mapping and construction rules to eliminate lexical entries. In this last chapter, we draw conclusion of our thesis by describing major contributions followed by some direction for future research.

## 5.1 Contributions

Our major contributions are enumerated as follows:

- We have discussed dimensions of Arabic nouns and verbs. Based on the dimensions we have proposed a detail type hierarchy for nouns and verbs.
- We have analyzed declension types of nouns and verbs and based on that, we have shown mapping of type hierarchy to declension types.
- We have devised an algorithm identify the declension type of a noun lexeme. This algorithm also proofs completeness of nine declension types and hence, completeness of classical 16 categories.

- We have shown AVM for English nouns and extend it for Arabic noun. In this AVM, we capture morphology, syntactic and semantic effect for an Arabic noun. Later, we have shown AVM for Arabic verbs.

- We have captured Arabic declension phenomenon and proposed construction rules for construction of genitive and accusative lexemes from accusative lexemes and for construction of jussive and subjunctive verb lexemes from indicative lexemes. These construction rules will eliminate numerous lexical entries and hence database of lexical entries will be much optimized.

- We have proposed construction rules to construct definite lexemes from indefinites.

- We have implemented our type hierarchy, AVM and construction rules in TRALE. We have verified construction rules for declension and definiteness in TRALE. Here we have also verified two level construction. For example, construction of *alkaatiba* from *kaatibun*. At top level, *alkaatiba* is verified from *kaatiban* using a definiteness rule. Then, *kaatiban* is verified from *kaatibun* using a declension rule.

## 5.2 Future Directions

Though linguistic modeling of Arabic language is a massive task, we believe this work will pave the way of it. Much scope is still open for research following this thesis. Following directions can be considered as future directions for further research:

- In this thesis, we have omitted analysis on plural number to preserve simplicity. There are five plural noun classes among 16 classes. These are $3, 4, 10, 12, 16$. So scope to explore declension of these classes is still open and this can be a future extension of this thesis.

- The thesis has considered only lexical construction that is, a lexeme is generated from another lexeme. For example, we have analyzed construction of كَاتِبًا (*kātiban*

) from كَاتِبٌ (*kātibun* ). We have not considered any phrasal construction. For example, construction of كَتَبَ كِتَابًا (*kataba kitāban* ) from two lexemes: كَتَبَ (*kataba* ) and كِتَابًا (*kitāban* ). So, research on phrasal construction can also be a future extension.

# Bibliography

[1] Kenneth R. Beesley. Finite-State Morphological Analysis and Generation of Arabic at Xerox Research: Status and Plans in 2001. In *Proceedings of the Workshop on Arabic Language Processing: Status and Prospects, Association for Computational Linguistics*, 2001.

[2] Md. Shariful Islam Bhuyan and Reaz Ahmed. An HPSG Analysis of Arabic Passive. In *Proceedings of the 11th International Conference on Computer and Information Technology*, 2008.

[3] Md. Shariful Islam Bhuyan and Reaz Ahmed. An HPSG analysis of arabic passive. In *Proceedings of the 11th International Conference on Computer and Information Technology*, 2008.

[4] Md. Shariful Islam Bhuyan and Reaz Ahmed. An HPSG analysis of arabic verb. In *The International Arab Conference on Information Technology*, 2008.

[5] Md. Shariful Islam Bhuyan and Reaz Ahmed. Nonconcatenative morphology: An HPSG analysis. In *the 5th International Conference on Electrical and Computer Engineering*, 2008.

[6] Steven Bird and Ewan Klein. Phonological Analysis in Typed Feature Systems. *Computational Linguistics*, 20:455–491, 1994.

[7] Leonard Bolc, Krzysztof Czuba, Anna Kupsc, Malgorzata Marciniak, Agnieszka Mykowiecka, and Adam Przepirkowski. A survey of systems for implementing

HPSG grammars. Technical report, IPI PAN (Institute of Computer Science, Polish Academy of Sciences, 1996.

[8] Joan Bresnan. *The Mental Representation of Grammatical Relations*. Cambridge, MA, USA: MIT Press, 1982.

[9] Tim Buckwalter. Buckwalter Arabic Morphological Analyzer Version 2.0. In *Linguistic Data Consortium*, Philadelphia, PA, USA, 2004.

[10] Bob Carpenter and Gerald Penn. The logic of typed feature structures. *Cambridge Tracts in Theoretical Computer Science*, 1993.

[11] Bob Carpenter and Gerald Penn. *The Attribute Logic Engine (Version 4.0). User's Guide*. Carnegie Mellon University, Pittsburgh, December, 2003.

[12] Noam Chomsky. Lectures on Government and Binding, 1981.

[13] Domenic Cipollone. Morphologically complex predicates in Japanese and what they tell us about grammar architecture. In *OSU Working Papers in Lingusitics 56*, pages 1–52. Ohio State University, 2001.

[14] Ann Copestake. *Implementing Typed Feature Structure*. Stanford: CSLI Publications, 2002.

[15] Ann Copestake, Dan Flickinger, Rob Malouf, Susanne Riehemann, and Ivan Sag. Translation using minimal recursion semantics. In *Proceedings of the 6th International Conference on Theoretical and Methodological Issues in Machine Translation, Leuven*, 1995.

[16] Ann Copestake, Dan Flickinger, Susanne Riehemann, and Ivan Sag. Minimal recursion semantics: An introduction. *Research on Language and Computation*, 3(4):281–332, 2006.

[17] Anthony R. Davis. *Linking and the Hierarchical Lexicon*. PhD thesis, Stanford University, 1996.

[18] Gerald Gazdar, Ewan Klein, Geoffrey Pullum, and Ivan A. Sag. *Generalized Phrase Structure Grammar*. Chicago: University of Chicago Press, 1985.

[19] Thilo Goetz. *The ConTroll User's Guide and Manual: First Draft*. Seminar fü r Sprachwissenschaft, Universitä t Tü bingen, 1995.

[20] Georgia M. Green. Elementary principles of HPSG. In *FIPS PUB*, pages 140–1, 1999.

[21] Kais Haddar and Ines Zalila. An HPSG parser generation with the lkb for arabic relatives. In *3rd International Conference on Arabic Language Processing (CITALA09)*, Rabat, Morocco, 2009.

[22] Md. Sadiqul Islam, Mahmudul Hasan Masum, Reaz Ahmed, and Shariful Islam Bhuyan. Arabic nominals in HPSG: A verbal noun perspective. In *Proceedings of the 17th International Conference on Head-Driven Phrase Structure Grammar*, Paris, France, 2010.

[23] Md. Sadiqul Islam, Mahmudul Hasan Masum, Md. Shariful Islam Bhuyan, and Reaz Ahmed. An HPSG analysis of declension in arabic grammar. In *Proceedings of the 9th International Arab Conference on Information Technology*, 2009.

[24] Mohtanick Jamil. Declension. Website, 2003-2011. `http://www.learnarabiconline.com/reflection.shtml`.

[25] Mohtanick Jamil. Definiteness. Website, 2003-2011. `http://www.learnarabiconline.com/definiteness.shtml`.

[26] Andreas Kathol. Agreement and the syntax-morphology interface in HPSG. In *Studies in contemporary phrase structure grammar*, pages 223–274. UC Berkeley, 1999.

[27] Alain Kihm. Nonsegmental Concatenation: A Study of Classical Arabic Broken Plurals and Verbal Nouns . *Morphology*, 16:69–105, 2006.

[28] Hans-Ulrich Krieger and Ulrich Schǎfer. Tdl: A type description language for HPSG (part 1 and part 2). *Research Report, RR-94-37*, 1994.

[29] M. Paul Lewis. Ethnologue: Languages of the world. Website, 2009. $http://www.ethnologue.org/ethno_docs/distribution.asp$.

[30] Eugene E. Loos, Susan Anderson, Jr. Dwight H., Day, Paul C. Jordan, and J. Douglas Wingate. Glossary of linguistic terms. Website, 2011. `http://www.sil.org/linguistics/GlossaryOfLinguisticTerms/`.

[31] Mahmudul Hasan Masum, Muhammad Sadiqul Islam, and Reaz Ahmed M. Sohel Rahman. HPSG analysis of type-based arabic nominal declension. In *Proceedings of the 12th International Arab Conference on Information Technology*, 2012.

[32] Mahmudul Hasan Masum, Muhammad Sadiqul Islam, M Sohel Rahman, and Reaz Ahmed. Type-based HPSG analysis of arabic verbal declension. In *the 7th International Conference on Electrical and Computer Engineering*, 2012.

[33] Nurit Melnik. From "hand-written" to computationally implemented HPSG theories. In *Proceedings of the 12th International HPSG Conference, University of Lisbon*, pages 311–321. On-line: CSLI Publications, 2005.

[34] W. Detmar Meurers, Gerald Penn, and Frank Richter. A web-based instructional platform for constraint-based grammar formalisms and parsing. In *InProc. of the Effective Tools and Methodologies for Teaching NLP and CL. ACL*, pages 18–25, New Brunswick, NJ, 2002.

[35] Stefan Muller. Grammix. Website, 2007. `http://hpsg.fu-berlin.de/Software/Grammix/`.

[36] A.M. Mutawa, Salah Alnajem, and Fadi Alzhouri. An HPSG approach to arabic nominal sentences. *Journal of the American Society for Information Science and Technology*, 59(3):422–434, 2008.

[37] Gerald Penn and Mohammad Haji-Abdolhosseini. ALE Documentation. Website, 2003. `http://www.ale.cs.toronto.edu/docs/`.

[38] Carl J. Pollard. *Lectures on the foundations of HPSG*. Chicago: University of Chicago Press, 1997.

[39] Carl J. Pollard and Ivan A. Sag. Information-based syntax and semantics. *Stanford: Center for the Study of Language and Information (CSLI)*, 1:262–267, 1987.

[40] Carl J. Pollard and Ivan A. Sag. *Head-Driven Phrase Structure Grammar*. Chicago: University of Chicago Press, 1994.

[41] Frank Richter. Priliminary TRALE Page. Website, 2008. `http://milca.sfs.uni-tuebingen.de/A4/Course/trale/`.

[42] Susanne Z. Riehemann. Type-Based Derivational Morphology. *Journal of Comparative Germanic Linguistics*, 2:49–77, 1998.

[43] Susanne Z. Riehemann. *A Constructional Approach to Idioms and Word Formation*. PhD thesis, Stanford University, 2001.

[44] Jr. Robert D. Van Valin. *Aspects of Lakhota syntax*. PhD dissertation, University of California, Berkeley, 1977.

[45] Jr. Robert D. Van Valin. Semantic macroroles in role and reference grammar. In *University at Buffalo, New York*, 2001.

[46] Karin C. Ryding. *Modern Standard Arabic*. Cambridge University Press, UK, 2005.

[47] Ivan A. Sag. *Sign-Based Construction Grammar*. Stanford University, August 2010.

[48] Ivan A. Sag and Thomas Wasow. *Syntactic Theory: A Formal Introduction*. Stanford University Center for the Study, 1999.

[49] Stuart M. Shieber. Natural-language processing: Grammar formalisms. In William Bright, editor, *The Oxford International Encyclopedia of Linguistics*, pages 61–64. Oxford University Press, New York, 1991.

[50] Otakar Smrž. *Functional Arabic Morphology. Formal System and Implementation.* PhD thesis, Charles University in Prague, 2007.

[51] Nathan Vaillette. Hebrew relative clauses in HPSG. In *Proceedings of the 7th International HPSG Conference, UC Berkeley (2223)*, pages 305–324. On-line: CSLI Publications, 2000.

# Appendix A

### Arabic Alphabet Transliteration

**Romanized transliteration of Arabic alphabet is given below.**

Table 5.1: Transliteration Table of Arabic Alphabet

| Arabic Letter | Transliteration | Arabic Letter | Transliteration |
|---|---|---|---|
| ا | *ā* | ط | *ṭ* |
| ب | *b* | ظ | *ẓ* |
| ت | *t* | ع | ʿ |
| ث | *ṯ* | غ | *ġ* |
| ج | *ğ* | ف | *f* |
| ح | *ḥ* | ق | *q* |
| خ | *ḫ* | ك | *k* |
| د | *d* | ل | *l* |
| ذ | *ḏ* | م | *m* |
| ر | *r* | ن | *n* |
| ز | *z* | و | *w* |
| س | *s* | عـ | ʾ |
| ش | *š* | ه | *h* |
| ص | *ṣ* | ي | *y* |
| ض | *ḍ* | ى | *ā* |

# Appendix B

### Signature file

```
type_hierarchy
bot
   list
     ne_list hd:bot tl:list
     e_list
   char
      k
      t
      b
      a
      i
      u
      n
      r
      s
      j
      d
   sign morph:morph arg_st:list  syn:syn sem:sem
      word dtrs:list dtr:sign
      lexeme
         noun_lex
            masc_noun_lex
              triptote_strong_masc_noun_lex
            fem_noun_lex
              triptote_strong_fem_noun_lex
            dual_noun_lex
            sg_noun_lex
              triptote_sound_sg_noun_lex
              triptote_pseudo_sg_noun_lex
              pure_sg_noun_lex
              triptote_ism_maqsoor_sg_noun_lex
            pl_noun_lex
              strong_noun_lex
              broken_noun_lex
                triptote_broken_noun_lex
            indeclinable_noun_lex
            declinable_noun_lex
              triptote_noun_lex
```

```
          diptote_noun_lex
          sound_noun_lex
            &triptote_sound_sg_noun_lex
          unsound_noun_lex
            ism_maqsoor_noun_lex
            unsound_ending_with_ya_waw_noun_lex
       verb_lex
          sg_verb_lex
            sound_sg_verb_lex
          dual_verb_lex
            sound_dual_verb_lex
          pl_verb_lex
            sound_pl_lex
          sound_verb_lex
            &sound_sg_verb_lex
            &sound_dual_verb_lex
            &sound_pl_lex
          unsound_verb_lex
            unsound_alif_ending_verb_lex
            unsound_waw_ya_ending_verb_lex
morph root:list form:list dec:dec
syn cat:cat val:list mrkg:mrkg
cat case:case def:def mood:mood vform:vform voice:voice
   noun
   verb
case
   nom
   acc
   gen
dec
  tn1
  tn2
  tn3
  tn4
  tn5
  tn6
  tn7
  tn8
  tn9
  tv1
  tv2
  tv3
  tv4
person
   first
   second
   third
number
   sg
   dual
   plural
gender
   male
   female
```

```
   def
      yes
      no
   hum
      y
      n
   vform
      perf
      imperf
   voice
      active
      passive
   mood
      subjunctive
      indicative
      jussive
   mrkg
      none
      that
   lid
   select
   sem index:index frames:list
   index pers:person num:number gen:gender hum:hum
   frame sit:index actor:index undgr:index
   ref_fr ref_index:cat
.
```

### Lexical entries for noun:

```
kaatibun ~~> (triptote_sound_sg_noun_lex,
                (morph:
                        (
                        root:[k,t,b],
                        form:[k,a,a,t,i,b,u,n],
                        dec:tn1
                        ),
                arg_st:[],
                syn:
                        (
                        cat:
                                (noun,
                                case:nom,
                                def:no
                                )
                        ,
                        val:[]
                        ),
                sem:(
                        index:SUB_INDEX,
                        frames:[
                                (sit:VERB_INDEX,actor:SUB_INDEX, undgr:OBJ_INDEX)
                                ]
                        )
```

```
                )
                ).

naasirun ~~> (triptote_sound_sg_noun_lex,
                (morph:
                        (
                        root:[n,s,r],
                        form:[n,a,a,s,i,r,u,n],
                        dec:tn1
                        ),
                arg_st:[],
                syn:
                        (
                        cat:
                                (noun,
                                case:nom,
                                def:no
                                )
                        ,
                        val:[]
                        ),
                sem:(
                        index:SUB_INDEX,
                        frames:[
                                (sit:VERB_INDEX,actor:SUB_INDEX, undgr:OBJ_INDEX)
                                ]
                        )
                )
                ).


saajidun ~~> (triptote_sound_sg_noun_lex,
                (morph:
                        (
                        root:[s,j,d],
                        form:[s,a,a,j,i,d,u,n],
                        dec:tn1
                        ),
                arg_st:[],
                syn:
                        (
                        cat:
                                (noun,
                                case:nom,
                                def:no
                                )
                        ,
                        val:[]
                        ),
                sem:(
                        index:SUB_INDEX,
                        frames:[
                                (sit:VERB_INDEX,actor:SUB_INDEX, undgr:OBJ_INDEX)
                                ]
```

```
                )
        )
        ).
```

**Lexical entries for verb:**

```
aktubu ~~>     (sound_sg_verb_lex,
                (morph:
                        (
                        root:[k,t,b],
                        form:[a,k,t,u,b,u],
                        dec:tv1
                        ),
                arg_st:[(OBJ_SIGN,(syn:(cat:(case:acc)), sem:(index:OBJ_INDEX)))],
                syn:
                        (
                        cat:
                                (verb,
                                vform:imperf,
                                voice:active,
                                mood:indicative
                                )
                        ,
                        val:[OBJ_SIGN]
                        ),
                sem:(
                        index:VERB_INDEX,
                        frames:[
                                (sit:VERB_INDEX,actor:SUB_INDEX, undgr:OBJ_INDEX)
                                ]
                        )
                )
                ).


ansuru ~~>     (sound_sg_verb_lex,
                (morph:
                        (
                        root:[n,s,r],
                        form:[a,n,s,u,r,u],
                        dec:tv1
                        ),
                arg_st:[(OBJ_SIGN,(syn:(cat:(case:acc)), sem:(index:OBJ_INDEX)))],
                syn:
                        (
                        cat:
                                (verb,
                                vform:imperf,
                                voice:active,
                                mood:indicative
                                )
                        ,
                        val:[OBJ_SIGN]
```

```
                ),
        sem:(
                index:VERB_INDEX,
                frames:[
                        (sit:VERB_INDEX,actor:SUB_INDEX, undgr:OBJ_INDEX)
                        ]
                )
        )
        ).

asjudu ~~>     (sound_sg_verb_lex,
        (morph:
                (
                root:[s,j,d],
                form:[a,s,j,u,d,u],
                dec:tv1
                ),
        arg_st:[(OBJ_SIGN,(syn:(cat:(case:acc)), sem:(index:OBJ_INDEX)))],
        syn:
                (
                cat:
                        (verb,
                        vform:imperf,
                        voice:active,
                        mood:indicative
                        )
                ,
                val:[OBJ_SIGN]
                ),
        sem:(
                index:VERB_INDEX,
                frames:[
                        (sit:VERB_INDEX,actor:SUB_INDEX, undgr:OBJ_INDEX)
                        ]
                )
        )
        ).
```

### Noun construction rules:

```
triptote-sound-sg-acc-tn1-noun-cxt##
(
         morph:
        (
                root:ROOTS,
                dec:tn1
        ),
        arg_st:ARGST,
        syn:
                (
                cat:
                        (noun,
                        case:nom,
```

```
                        def:no
                        )
                ,
                val:VAL,
                mrkg:MRKG
                ),
        sem:SEM
)
**>
(
         morph:
        (
                root:ROOTS,
                dec:tn1
        ),
        arg_st:ARGST,
        syn:
                (
                cat:
                        (noun,
                        case:acc,
                        def:no
                        )
                ,
                val:VAL,
                mrkg:MRKG
                ),
        sem:SEM
)
morphs
(X,u,n) becomes (X,a,n)
.

triptote-sound-sg-gen-tn1-noun-cxt##
(
         morph:
        (
                root:ROOTS,
                dec:tn1
        ),
        arg_st:ARGST,
        syn:
                (
                cat:
                        (noun,
                        case:nom,
                        def:no
                        )
                ,
                val:VAL,
                mrkg:MRKG
                ),
        sem:SEM
)
```

```
**>
(
         morph:
        (
                root:ROOTS,
                dec:tn1
        ),
        arg_st:ARGST,
        syn:
                (
                cat:
                        (noun,
                        case:gen,
                        def:no
                        )
                ,
                val:VAL,
                mrkg:MRKG
                ),
        sem:SEM
)
morphs
(X,u,n) becomes (X,i,n)
.
```

### Verb construction rules:

```
sound-sg-subjunctive-tv1-verb-cxt##
(
         morph:
        (
                root:ROOTS,
                dec:tv1
        ),
        arg_st:ARGST,
        syn:
                (
                cat:
                        (verb,
                        mood:indicative,
                        def:no
                        )
                ,
                val:VAL,
                mrkg:MRKG
                ),
        sem:SEM
)
**>
(
         morph:
        (
```

```
                root:ROOTS,
                dec:tv1
        ),
        arg_st:ARGST,
        syn:
                (
                cat:
                        (verb,
                        mood:subjunctive,
                        def:no
                        )
                ,
                val:VAL,
                mrkg:MRKG
                ),
        sem:SEM
)
morphs
(X,u) becomes (X,a)
.

sound-sg-jussive-tv1-verb-cxt##
(
         morph:
        (
                root:ROOTS,
                dec:tv1
        ),
        arg_st:ARGST,
        syn:
                (
                cat:
                        (verb,
                        mood:indicative,
                        def:no
                        )
                ,
                val:VAL,
                mrkg:MRKG
                ),
        sem:SEM
)
**>
(
         morph:
        (
                root:ROOTS,
                dec:tv1
        ),
        arg_st:ARGST,
        syn:
                (
                cat:
                        (verb,
```

```
                        mood:jussive,
                        def:no
                        )
                ,
                val:VAL,
                mrkg:MRKG
                ),
        sem:SEM
)
morphs
(X,u) becomes (X)
.
```

**Definiteness construction rule:**

```
triptote-sound-sg-def-noun-cxt##
(
        morph:
        (
                root:ROOTS,
                dec:tn1
        ),
        arg_st:ARGST,
        syn:
                (
                cat:
                        (noun,
                        case:CASE,
                        def:no
                        )
                ,
                val:VAL,
                mrkg:MRKG
                ),
        sem:SEM
)
**>
(
        morph:
        (
                root:ROOTS,
                dec:tn1
        ),
        arg_st:ARGST,
        syn:
                (
                cat:
                        (noun,
                        case:CASE,
                        def:yes
                        )
                ,
                val:VAL,
```

```
                mrkg:MRKG
                ),
        sem:SEM
)
morphs
(X,a,n) becomes (a,l,X,a),
(X,u,n) becomes (a,l,X,u),
(X,i,n) becomes (a,l,X,i)
.
```