

Arabic weak verb formulation and computation

Zahed Ahmed

Pembroke College and Computer Laboratory
University of Cambridge

za209@cam.ac.uk

Abstract

This paper presents a computational analysis of Arabic weak verbs implemented under a multi-tape two level model in Prolog. We describe weak rewrite rule derivations for the weak environment and illustrate their use using a finite state transducer implementation in Prolog.

1 Introduction

Two level theory was introduced by Koskeniemi (1983) and has adopted the position as the standard model in computational linguistics. This was created through the highly agglutinative, yet linear morphology of Finnish. This model has been successfully applied to many other languages.

Traditional two level theory assumes two levels of illustration, lexical and surface, with a mapping between them realised as finite state transducers (FSTs). It also assumes that surface forms are created by concatenation, which for non-linear systems like Arabic can create incorrectly formed words. This paper will adopt the convention of morphemes in braces { } and surface forms in solidi / /, and kasra, fatha and dhamma take their usual Arabic meaning. Kiraz uses the example of /kutib/ ‘to write – perfect passive’, which consists of the morphemes {ktb} and {ui} ‘perfect passive’. The concatenation of these morphemes produces /ktbui/, which is clearly not well formed. As an effort to

overcome this downfall, Kay (1987) split the lexical morphemes so that they would inhabit multiple tapes. This leads us to the current system of a multi tape two level system (Kiraz 1994), motivated by the above mentioned and a two level formalism by Pulman and Hepple (1993).

2 Problem Description

There has been a fair amount of computational research on Arabic morphology; however the less-studied weak verb paradigm provides rather complex problems. The presence of any of the 2 weak letters, ‘waw’ and ‘yaa’ make a given verb weak. However, we also tackle another form of weakness whereby any given verb containing the ‘hamza’ also can make the verb weak. Weakness can be divided into 4 groups:

Mahmooz: hamzated verb

Misaal: weak in the 1st radical

Ajwaf: weak in the 2nd radical

Naaqis: weak in the 3rd radical

The presence of weak letters in verbs causes complex morphology, differing from the classical morphology of strong verbs (defined conversely to weakness). In an attempt to formalise weakness and its effects, we extend the original multi-tape grammar by Kiraz (1995) to accommodate the complexity. Figure 1a illustrates 3 lexical tapes and one surface tape. The root tape provides the roots of the verb, the vowel tape, the vowels of the roots and the pattern tape defines the verbal pattern

within which the verb will operate. This system is based around the CV scheme (McCarthy, 1981). The surface tape shows the morphological result of processing the lexical information. Each row represents a tape and each column represents a 3-tuple lexical form and a 1-tuple surface form. Notionally, each column 3-tuple is a unit of data. We note that the morphological transductions are based on rules (a detailed description which will follow). The processing of weak verbs generally requires a larger amount of knowledge of the context in which any given weak letter resides in order to produce correct results. Hence, the weak verb problem is principally one of context flexibility, which is the ability of the rule formalism to handle the dynamism of context.

	A			U			A	vocal
Y		Q	W		L	N		root
C	V	C	C	V	C	C	V	pattern
list the rules used								
Y	A	Q	W	U	L	N	A	surface

Figure (1a) – CV description of multi-tape analysis

Figure 1a illustrates the typical analysis for strong verbs where the rules used to produce the surface form have static context, which in this case, due to the presence of weakness, produces an incorrect surface form. However, in our formalism designed to handle extended context, the surface form is correctly calculated (Figure 1b).

	A			U			A	vocal
Y		Q	W		L	N		root
C	V	C	C	V	C	C	V	pattern
list the rules used								
Y	A	Q		U	L	N	A	surface

Figure (1b)

3 Computational Analysis

Figure 2 shows the existing approach, which can be extended to allow for n-tuples such that the *i*th expression in the n-tuple refers to the *i*th tape. Note also, the rigidity in predefining only three levels of context, i.e. LLC, LEX, RLC or the surface equivalent.

a.	LSC-SURF-RSC => LLC-LEX-RLC
b.	LSC-SURF-RSC<=> LLC-LEX-RLC
where	
	LSC = left surface context
	LLC = left lexical context
	SURF = surface form
	LEX = lexical form
	RSC = right surface context
	RLC = right lexical context
	=> = optional rule
	<=> = compulsory rule

Figure 2: The Pullman Hepple/Ru-essink/Black *et al.* formalism.

Figure 3 shows the formalism implemented as the basic rules used by Kiraz (1995). The identity rules for vowels and consonants are given. The rule system is closely based on the scheme in Figure 2, where the '*' denotes a wildcard and '_' is null. Each context term is a 3-tuple (in the lexical case, 1-tuple in the surface case), where the first term corresponds to the pattern tape, the second to the root tape and the third to the vowel. That is in F1, LSC=*, SURF=X, RSC=*, LLC=(*,*,*), LEX=(C,X,_) and RLC=(*,*,*). Note also that the C in LEX corresponds to the pattern tape, the X in LEX corresponds to the root tape and the null term in LEX to the vowel tape, as the C pattern can only encode a consonant and not a vowel. So the term (C,X,A) would not be valid.

General Rules

F1: Consonants:
 $*-X-* \Rightarrow (*,*,*)-(C,X,_)-(*,*,*)$

F2: Vowels:
 $*-X-* \Rightarrow (*,*,*)-(V,_,X)-(*,*,*)$

Figure 3: basic CV grammar rules

As described above, more context is needed for weak verbs. We accommodate this by allowing each context term to consist of more than one n-tuple. We use a context function Q, which takes a context term as its parameter, to define the context within any context term.

$$Q(f) = \begin{cases} +|f| & f \in \{LEX, RLC, SURF, RSC\} \\ -|f| & f \in \{LLC, LSC\} \end{cases}$$

where f is a given context term and | | denotes magnitude. The magnitude of the function denotes the number of levels of context within any given context term and the direction is, by convention negative, if we are looking at the LLC and positive if we are looking at LEX or RLC. This reflects the notion of reading in an input list of tuples, and maintaining 2 lists. One which has been read, which includes LLC and another which is or will be read i.e. including LEX and RLC. Thus, figure 4a, presents LEX which consists of three 3-tuples, where Q(LEX) = +3.

(a) LEX = (C1V1C2,qw,a)

(b) $*-V-* \Rightarrow (C,B,_)-(V,_,A)-(C,T,_)$

(c)
 $*-V-* \Rightarrow (C,C,_)-(CVCVC,CCC,VV)-$
 $(C,*,_)$

Figure 4

Figure 4b is an example of the context system used in strong verbs, where each context term has Q(f) = 0 or Q(f) = 1. Note also that in this case LLC and RLC are used maximally, however, any weak rule requiring more the three levels of context would fail.

Figure 4c shows a rule which has Q(LEX) > 1. This increased flexibility allows us to write rules which consider enough context to produce correct results as opposed to the static null or one level context scheme of string verbs.

We produced a solution to the weak verb problem by adopting the rules in figure 3 as a basis and producing a detailed set of rules using an extended context for each of the four groups of weakness. We allowed each term in the n-tuples to be m-tuples, where m could vary such that 0 < m < MAX. We define MAX to be the smallest value of the function Q such that any weak rule will have enough context to produce a correct result. This allowed the development of rules such as:

R1: $*-V_1V_1-* \Leftrightarrow (C,*,_)-(VCV,wk,a*)-$
 $(C,*,_)$

In this example Q(LLC) = -1, Q(LEX) = +3 and Q(RLC) = +1. Note also that LEX consists of three 3-tuples, with the same tuple-tape correspondence as the rules in figure 2. Figure 5 demonstrates the use of the derived weak rules. We use the same example as mentioned in Figure 1.

	A			U			A
Y		Q	W		L	N	
C	V	C	C	V	C	C	V
F1	F2	F1	R4	F2	F1	F1	F2
Y	A	Q		U	L	N	A

Figure 5: The 1st, 2nd, 3rd, 4th and 5th rows represent the vowel, root, pattern rule and surface tapes respectively. The rule numbers between the pattern tape and surface tape represent the rules used to

sanction the transduction. Rules F1 and F2 refer to those mentioned in figure 2, however we highlight weak rule R4.

R4:

-V1- \Leftrightarrow (VC,C,V)-(CV,wk,V1)-
(C,*,_)

This rule uses 'extended context' in two of the three lexical terms and other rules use further context in all three terms. This enabled us to fully describe all the phenomena.

4 Implementation

In contrast to the rules for strong verbs, there is non-determinacy in the size of any context term (depending on the rule chosen) and thus the implementation has to allow for this. We refer to the algorithm developed for this as 'extended context unification' (ECU) implemented in Prolog. We implemented our formalism in a test suite written in Prolog. We developed both morphological generator and analyser, by implementing the grammar as a finite state transducer and modelling this in Prolog. Previous work in modelling multi-tape formalisms in Prolog include, Kiraz(1996) and Kiraz and Grimley-Evans(1998). However, we develop the principal extended context unification (ECU) algorithm, which implements the modified weak rules. We found the previous systems were designed for strong verb morphology and so cannot handle the added complexity in weakness. Figure 5a, illustrates the mechanics of the algorithm.

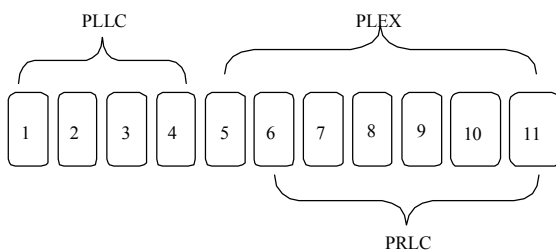


Figure 5a: ECU - Case 1

The FST has processed the 1st 4 tuples and awaits the processing of the remaining tuples. In this case each tuple is a 3-tuple. They could be n tuples as defined in the definition of an n tape FST. PLEX is the set of tuples which could potentially comprise the LEX of the next rule application. PRLC the maximum set of tuples that could potentially be used in the following rule application. Note we do not allow for the empty LEX set. This reflects a null operation of the FST, so no change occurs

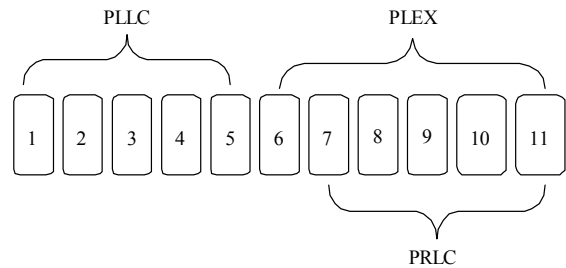


Figure 5b: ECU - Case 2a

Case 2a assumes the $\text{Magn}(\text{LEX}) = 1$, and so in essence it models the strong verb case, where no more than 1 level of context is ever required. Finally, the weak case in Figure 5c where $\text{Magn}(\text{LEX}) > 1$. Let us assume for this example that LEX consists of the tuples 5, 6, and 7 respectively. Therefore, after applying the given rule, we append the contents of LEX to the PLLC to give the new PLLC, i.e. tuples 1 to 7 inclusive. The new PLEX is tuples 8 to 11 and again PRLC is 9 to 11 for the same reason as mentioned in case 1.

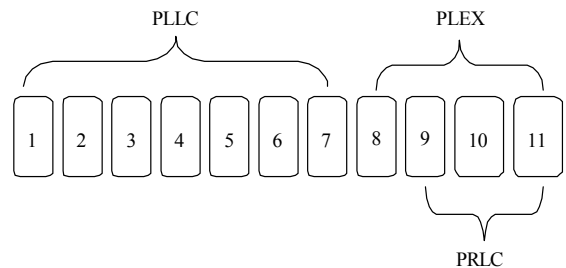


Figure 5c: ECU- Case 2b

We developed the system, through extensive use of ECU and interpreting the computational formalism in section 2, into Prolog terms and clauses. The interpretation of the rule formalism mentioned above into Prolog resulted in input queries of the form in figure 6. The generator is called with the list of tuples representing the ungenerated input and the result unifies with the 'Out' variable. The same format is used for the analyser except, the input is the generated surface in tuple form and the result is a list of analyses.

<pre> Input = [[t(c,q,[]),t(v,[],a),t(c,w,[]),t(v,[],a), t(c,l,[]),t(c,n,[]),t(v,[],a)] Generate(Input, Out). Out = result </pre>

Figure 6: Typical generator input, where [] represents null.

We tested both the generator and analyser over a set of approximately 100 test cases. The test set was chosen in order to test all the verbal paradigms for both weak and strong verbs. This ensured that nothing used to solve the weak verb case violated anything in the strong verb case. Our results indicate the correct and accurate working of the system for both analysis and generation. In addition, analysis of input generally takes a longer period of time than generation due to the fact the generation is a many to one operation and analysis is a one to many operation. Therefore due to backtracking, the system produces multiple possible results. Our results have indicated that the current system, although working correctly, does not implement an optimal search of the rule space. This is mentioned in the following section.

Conclusions and Further Work

We have presented a description of the complexity of the weak verbs and the problems that arise in processing them. In addition we provided a solution to the problem which has been evaluated on a test suite containing examples of all the weak verb types. We believe, the weak verb problem is principally one of context flexibility; hence our solution may be applied to other languages where context flexibility is necessary. We hope to extend this formalism to doubled and 4 radical verbs, as we believe these are also ultimately problems in extended context.

In addition, we are currently developing an optimised system based on a dynamic weighted FST description to implement optimal transduction priority and formulating an oracle function which dynamically updates weights on every transduction. This method would effectively empower the FST model to avoid many non-deterministic branches (which is the root cause of the inefficiency of the current system) and as a result, progressively narrow the search space on each transduction and improve efficiency. The oracle function could be described as a heuristic function based on the input and context functions.

Acknowledgements

I would like to thank Pembroke College, University of Cambridge for their generous financial provision. I am also grateful to Dr Michael Wise and Dr Omar Mahroo for their continuous encouragement and support. I would also like to thank my research supervisor, Dr Ann Copestake for her continued patience, support and excellent advice.

References

- Aho, A. and Ullman, J. (1977).
Principles of Compiler Design.
Addison-Wesley.
- Beesley, K. and Karttunen, L. 2000
Finite-State Non-concatenative
Morphotactics.
*Sigphon-2000, Proceedings of the 5th
Workshop of the ACL Special Interest
Group in Computational Phonology*, p.1-
12, Luxembourg.
- Grimley-Evans, E., Kiraz, G., and Pulman,
S. (1996).
Compiling a partition-based two-level
formalism.
COLING-96, pp. 246-60.
- Goldsmith, J. (1976).
Autosegmental Phonology.
PhD thesis, MIT. Published as
Autosegmental and Metrical Phonology,
Oxford 1990.
- Hopcroft, J.E. and Ullman, J.D 1979.
*Introduction to automata theory,
languages and computation*.
Addison-Wesley Series in Computer
Science. Addison-Wesley Publishing
Company, Reading Mass.
- Kiraz, G. 1996.
Semhe: A generalised two-level system.
In *Proceedings of the 34th Annual Meeting
of the Association of Computational
Linguistics*, Santa Cruz, CA.
- Kiraz, G. and Grimley-Evans, E. 1998.
Multi-tape automata for speech and
language systems: A prolog
implementation.
Derick Wood and Sheng Yu, editors,
Automata Implementation. Second
International Workshop on Implementing
Automata, WIA '97, pages 87-103.
Springer Lecture Notes in Computer
Science 1436, 1998.
- McCarthy, J. 1981.
A prosodic theory of nonconcatenative
morphology
Linguistic Inquiry, 12(3), pp.373-418.
- Narayanan, A. and Hashem, L. (1993).
On abstract finite-state morphology
EACL-93, pp. 297-304.
- Pulman, S. and Hepple M. 1993.
A feature based formalism for two-level
phonology: a description and
implementation.
Computer speech and Language, 7,
pp.338-58.
- Wright, W. 1988.
A Grammar of the Arabic Language
Cambridge University Press, Cambridge,
3rd edition.