

Issues at the Morphology-Syntax Interface in the Urdu ParGram Grammar

June 9, 2007

Abstract

70-150 words

1 Introduction

As part of the ParGram project (Butt et al. 1999, 2002), we are developing a grammar for the South Asian language Urdu. Very few resources exist for this language, in particular, no broad-coverage finite-state morphological analyzer exists to date. Part of the Urdu Grammar project is therefore to build a finite-state morphological analyzer for Urdu and to connect it up with the syntax via the interface (Kaplan et al. 2004) defined for Lexical-Functional Grammar (LFG; Dalrymple 2001).

Current features of the Urdu ParGram project in the context of parallel grammar development have already been discussed elsewhere (Butt and King 2007). In this paper, we focus on some issues that have arisen with respect to the morphology-syntax interface in particular. All the (larger) ParGram grammars to date include a finite-state morphological analyzer that interfaces with the syntax. These morphological analyzers are generally built with the Xerox finite-state technology tools and follow the methodology established by Beesley and Karttunen (2003). The finite-state tools and the solutions already proposed by Beesley and Karttunen (2003) prove to be more than adequate to meet the challenges posed by Urdu. However, some interesting issues do arise with respect to 1) the script and tokenization (sections 2 and 2.2); 2) reduplication (section 3) ; 3) deciding how to deal with potentially ambiguous information in terms of the morphology-syntax interface (section 4).

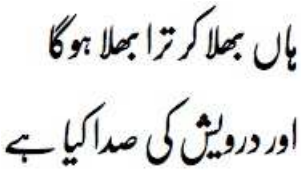
2 Two Different Scripts, One Representation

Urdu is structurally almost identical to Hindi. The major difference is that the vocabulary of Urdu bears more Persian/Arabic influences, while the vocabulary of Hindi is more Sanskrit based. Both are ultimately descended from a version of Sanskrit (i.e., are Indo-European). Urdu as a separate version of the language came into being when the Moghuls invaded the Indian subcontinent. The language of their court was Persian, which came into contact with a local language generally referred to as Hindustani (or Hindi). The very Persianized version of this language came to be known as Urdu.¹

This brief historical sketch is of relevance for morphological analysis because lexical items borrowed in from Persian tend to behave differently (i.e., have different inflectional possibilities). However, questions of lexical and morphological origin tend to be minor issues. A more major issue is that Urdu and Hindi are written in very different scripts. Urdu is written with a version of the Arabic script and it is only recently that unicode fonts for this script have been developed (e.g., see <http://www.crulp.org>; Rahman and Hussain 2003). Hindi, in contrast, is written in *Devanagari*, a phonetic-based script passed down over the millenia from Sanskrit.

2.1 A Common Transliteration System

Examples (1) and (2) show a couplet (162,9) from the poet Mirza Ghalib (1797–1869): (1) is written in Urdu, (2) is the same couplet, but written in Devanagari (Hindi). Note that Urdu is written left-to-right, whereas Hindi is written right-to-left.

(1) 
ہاں بھلا کرتا بھلا ہوگا
اور درویش کی صدا کیا ہے

¹Modern Hindi naturally also bears traces of language contact with Persian, but not as markedly as Urdu.

हां भला कर तिरा भला होगा
और दरवेश की सदा क्या है

(2)

Although the two writing systems differ markedly, the languages they are encoding are structurally almost identical. Given this fact, our general strategy in building a morphological analyzer is to produce a resource that can be used for text written in both Urdu and Hindi. This involves building a transliteration system that goes from whichever script is being processed to a common ASCII base and then being able to generate back out from the common ASCII base to either one of the scripts. That is, both the texts in (1) and (2) would be rendered as the ASCII transliteration in (3).

(3) hAN bHalA kar tirA bHalA hOgA
yes good.M.Sg do then good be.Fut.M.Sg
Or darvES kI sadA kyA he
and dervish Gen.F.Sg call.F.Sg what be.Pres.3.Sg
'Yes, do good then good will happen, what else is the call of the dervish.'

Our transliteration is based on proposals by Glassman (1977). Capitalized vowels indicate length, H marks aspiration, N nasalization, S stands for ʃ and other capitalized consonants indicate retroflexes.

A transliterator which implements our strategy has been implemented by Abbas Malik (2006). Malik's HUMTS (Hindi-Urdu Machine Transliteration System) is written as a cascade of finite-state transducers and transliterates from the Urdu and Hindi scripts to SAMPA (Wells 1997), a common underlying phonetic ASCII alphabet. While this is in principle a good idea because SAMPA has been developed to enable coverage of all the world's languages, for the purposes of Urdu, it is unwieldy and very difficult to read. In integrating Malik's work into the Urdu grammar, we will therefore move to the transliteration proposed by Glassmann. Beyond the simple conversion of letters that is necessary to do this, we anticipate no further (major) problems as HUMTS was written with the same XFST tools used in our Urdu grammar project.

2.2 Future Morphology: Illustrating Tokenization Problems

Writing a transliterator that takes one script as an input and is able to output another script is not an easy task. Many of the problems that arise are discussed in

Malik's work. In terms of the Urdu Grammar, most relevant to us are problems of tokenization. Just one of the problems is illustrated here in a representative manner with respect to the future morphology in Urdu/Hindi.

We already had an example of future usage in (1) and (2). An inspection of each example will quickly reveal one of the very general problems in dealing with the Urdu script: while in Hindi, each word is clearly demarcated and easy to identify, in Arabic-based scripts in general, word boundaries are very difficult to identify. One must basically know the language (i.e., be able to access the lexical items in the language) in order to be able to read the script.²

Beyond this very general problem that a tokenization of Urdu poses, the scripts also encode differences of opinion as to what exactly a word is. This is illustrated in (1) and (2) with respect to the future form of 'be' *hOgA*. In (1) it is expressed by the last two letter groups on line one (reading from right to left). In (2), the form is expressed by just one letter group: the last one (reading from left to right) on line 1. This difference in encoding reflects an on-going historical change in progress.

The future in Urdu/Hindi is formed as shown in the paradigm (4) for the stem *mAr* 'hit/kill'. The stem is followed by information about person and number (*UN/E/EN/O*), to which the future marker *g* is attached. This, finally, is followed about information about number and gender.

(4)

Urdu Future Paradigm				
	Singular	Plural	Respect (ap)	Familiar (tum)
	M/F	M/F	M/F	M/F
1st	mAr-UN-g-A/I	mAr-EN-g-E/I		
2nd	mAr-E-g-A/I		mAr-EN-g-E/I	mAr-O-g-E/I
3rd	mAr-E-g-A/I	mAr-EN-g-E/I		
mAr- 'hit'				

The future paradigm is thus a relatively complex assemblage of morphological pieces. The person/number morphology is identical to that used in the subjunctive paradigm, shown in (5). To these essentially subjunctive forms, a *-g-* is attached to mark the future. The consensus in the available literature is that the future *-g-* is derived from a Sanskrit participle of the verb *gā* 'go' (Kellogg 1893, Beg 1988, McGregor 1968). This analysis immediately explains the gender and number agreement morphology (*A/I/E*) exhibited by the future. Participles functioned

²The same is not true for Devanagari, which, being phonetically based, allows a sounding out of the words.

like adjectives and so generally had number and gender agreement morphology. This morphology has simply been retained in all the verb forms in Urdu/Hindi that derive from old participles (i.e., the perfect, imperfect and progressive forms), including the future.

(5)

Urdu Subjunctive Paradigm				
	Singular	Plural	Respect (ap)	Familiar (tum)
1st	mAr-UN	mAr-EN		
2nd	mAr-E		mAr-EN	mAr-O
3rd	mAr-E	mAr-EN		

mar- 'hit'

The old participle of the verb *gā* 'go' used to form its own word. Indeed, as recently as a century ago, clitics like the emphatic *hi* 'even/only' could intrude between the -g- and the stem+subjunctive morphology. This is illustrated in (6).

- (6) a. kah-ũ=hi=ga
 say-1.Sg-Emph-Fut.M.Sg
 'I will say (it), of course.' (Hindi, from Kellogg 1893:§399)
- b. man-e=hi=gi
 heed-3.Sg-Emph-Fut.F.Sg
 'She will (have to) see reason.' (Hindi, from Kellogg 1893:§399)

These examples suggest that while the old participle was no longer functioning as an independent word a century ago, it still retained some sort of prosodic independence and was probably functioning as a clitic (as the glossing in terms of '=' indicates). This is entirely consonant with well known processes of historical change whereby words are reanalyzed as clitics and then inflectional morphology as they move from being content words to functional elements (e.g., Harris and Campbell 1995, Hopper and Traugott 1993).

The examples in (6) are only marginally possible in modern Urdu, whereas speakers of Hindi tend to reject them outright. This difference in native speaker judgements may or may not be correlated with the differences encoded in the writing system. Recall that in written Hindi, the future is expressed in one word together with the subjunctive stem. In Urdu however, the stem+subjunctive and the future+number+gender are generally written as two separate words.

In both languages all the pieces of morphology involved nevertheless perform exactly the same function, so our morphological analyzer should treat them in

parallel. In the morphological analyzer, the future *-g-* is treated as an inflectional morpheme and a form like *mArEgI* would be analyzed as in (7).

- (7) *mArEgI* \Leftrightarrow
 mAr+Verb+Subjunct+2P+Sg+Fut+Fem
 mAr+Verb+Subjunct+3P+Sg+Fut+Fem

The tokenizer thus has to turn the Urdu input of *mArE gI* into *mArEgI*. This in and of itself does not present a problem, since the deletion of white space is not a problem. In principle, since forms like *marE* are also words in their own right, a serious ambiguity problem could arise. However, as *gI/gA/gE* are not words in their own right, there is no problem given our basic approach.

Other problems with tokenization do, of course, exist. The future morphology provides just one example of potentially problematic factors that must be dealt with. Another, perhaps more interesting problem is that of reduplication.

3 Reduplication

Urdu/Hindi, like most of the South Asian languages, tends to use reduplication quite frequently (Abbi 1991). All content words can generally be reduplicated and the effect of the reduplication is to either strengthen/emphasize the original word or to express something like “and those kinds of things”.

- (8) a. *kHAnA* *vAnA*
 food.M.Sg. Redup
 ‘food and those kinds of things’
 b. *tHanDA* *tHanDA*
 cold.M.Sg. Redup
 ‘ice cold (cold cold)’

There are two different kind of reduplication strategies. In the one illustrated by (8a), the onset of the content word is replaced with another consonant. This consonant could be either *v*, *t* (T) or *ʃ* (S). In another strategy ((8b)), namely the word is simply repeated. The former strategy is generally described as *echo formation* or *echo reduplication*.

In this section, we show the solutions implemented in our finite-state morphological analyzer for verbs, adjectives and nouns. While the general strategy for dealing with reduplication is similar, each of these word classes presents some specialized problem that needs to be dealt with as well.

3.1 General Strategy

Generally, reduplications are written as separate words in both Urdu and Hindi. The fundamental problem facing the tokenizer is thus the fact that a reduplicated item must be recognized. The transliteration system will yield two words, as shown in (9), for example, which are separated by white space.

- (9) calnA valnA
walk.Inf.M.Sg Redup
‘walking and such things’

Our morphological analyzer basically follows the solution for full word reduplication presented by Beesley and Karttunen (2003) for Malay. The basic lexicon built independently of reduplication for nouns, verbs, adjectives and other content verbs interacts with regular expressions that are formulated to allow for reduplication.

The morphological analysis of reduplications as in (9) is as shown in (10). That is, within the morphological analyzer, the reduplicated form is simply registered via the tag +REDUP and is passed on as such to the Urdu grammar, which can decide how to use this information (or whether to use the very subtle semantic information implied by reduplication at all).

- (10) cal+Verb+Inf+Masc+Sg+Redup

In the Malay example presented by Beesley and Karttunen, the original word and the reduplicated part are merged into a single word. Our implementation thus differs from theirs in that we need to deal with the white space. Currently, we deal with this by introducing the multiword %Hyphen into the lexc source file. When dealing with reduplication, we thus internally represent the two words involved as being connected with a hyphen.

Reduplication itself is managed via the introduction of the multicharacter brackets "^[" and "^] ", which mark the domain of reduplication. When reduplication has successfully been applied the compile-replace operator must be invoked in order to apply a bracket filter that removes the brackets. In a further step, the hyphen introduced for internal management is also eliminated and replaced with a white space.

This part of the process is illustrated with respect to just the adjective ‘cold’ in terms of simple reduplication by the code shown below. The first part reproduces a lexc file which NEED SOME TEXT SAYING EXACTLY WHAT THIS PROGRAM DOES HERE.

```

*****
* !lexc file just illustrating tHanDa
* !AdjRedup.txt
*
* Multichar_Symbols
* +Adj +Unmarked +Redup +Intensifier
*
* Lexicon Root
* 0:^[[{ Unmarked ;
*
* Lexicon Unmarked
* tHanDA Ending ;
*
* Lexicon Ending
* +Adj+Unmarked+Redup+Intensifier:}%Hyphen]^2^] # ;
* +Adj+Unmarked:}%]^1^] # ;
*
*****
* !AdjReduprules.txt
* [ ~ [ ?* "^[ " ~$["^"] ] & ~[ ~$["^[ " " ^]" ?* ] ];
* ! bracket filter
*
*****
* !hyph.txt
* [ %Hyphen -> 0 || %Hyphen ?* _ ]
* .o.
* [ %Hyphen -> " " ] ;
* ! removes '%Hyphen' and inserts an empty string
*
*****

*****
* "commands"
*
* xfst[0]: read regex < AdjReduprules.txt
* xfst[1]: read lexc < AdjRedup.txt
* xfst[2]: compose net
* xfst[1]: compile-replace lower

```



```

* xfst[1]: save stack AdjRedup.fst
* xfst[1]: clear
* xfst[0]: read regex < hyph.txt
* xfst[1]: load stack AdjRedup.fst
* xfst[2]: compose net
* xfst[1]: up thanDA
* t h a n D A +Adj+Unmarked
* xfst[1]: up thanDA thanDA
* t h a n D A +Adj+Unmarked+Redup+Intensifier
*
*****

```

The second part shows the commands (to be saved in an xfst script file), which results in the complete analyzer for adjectives. MORE NEEDS TO BE SAID HERE AS WELL — WHAT ARE ALL THESE FILES??? WHERE ARE THEY? WHAT'S IN THEM? ETC.

The example above also illustrates the general strategy with respect to reduplication. The lexicon (lexc file) itself must be amended to allow for whole word reduplication in principle. The reduplication itself must be formulated by means of regular expressions, delimited by the multicharacter brackets " \wedge [" and " \wedge "]. Then, a bracket filter is composed onto the lexicon, which aims at blocking all strings containing unmatched brackets. Next, the compile-replace algorithm is called, which translates the reduplication formulated by " $[\dots]^2$ " into well-formed strings of this scheme: [...]-[...]. After having accomplished reduplication, the hyphen and the brackets are eliminated as already described above.

3.2 Echo Formation

In order to allow for the type of reduplication where the first consonant is replaced in the reduplicated part, replace rules need to be put into place.

THIS NEXT BIT WRITTEN BY TINA WOULD SEEM TO BE INTRESTING IT SHOULD AT LEAST MAKE IT INTO A FOOTNOTE, BUT I NEED IT TO BE EXPLAINED BETTER AND CAREFULLY WITH RESPECT TO EXAMPLES AND EXAMPLE CODES

There are two problems in this kind of morphology one has to deal with carefully: 1) The compile-replace brackets can never be on the same line as a regular expression (you could not for example add it to the LEXICON Suffix and finish

there) 2) The { }, which is here used as a separation between the two words deriving from the reduplication does have an unwanted consequence. After putting everything on the stack one gets the right output but one cannot apply up a reduplicated word: apply up bacca bacca will result in a fault. and 0 are not an option since does result in the same problem and 0 will not form a white space. If one chooses to use a symbol between the words, e.g. a hyphen, the apply up will work: apply up bacca-bacca- will result in an analysis. (I am not sure what the reason is this what the tokenizer does you wanted us to build? I have not solved this yet but I am trying; up to then I use the she uses the records with me I cannot test if this solution returns an analysis; perhaps this is also a fault of the windows version because hers seems to work. I cannot see the difference between turning and implementing in the Lex)

The rules shown below exemplify just two cases: either the first consonant (onsets tend to be just simple consonants) is replaced by a *v*, or if there is no onset as in (11), a *v* is inserted. Similar rules are formulated for reduplication versions with *t* or *ʃ*.

(11) EXAMPLE HERE

```
define Cons [ b | c | d | f | g | h | j | k | l | m
| n | p | q | r | s | t | v | w | x | z ];

Cons -> v || ?* %_ _ ?* "@P.ECHO.v@" ! use { } instead of %_
.O.
a -> v a , e -> v e , i -> v i , o -> v o , u -> v u || ?* %_ _ ?*
%% "@P.ECHO.v@"
```

NEED TO EXPLAIN EXACTLY WHAT THIS DOES

4 Issues in Potential Ambiguity

4.1 Politeness and the Subjunctive

When addressing the second person of the plural in the subjunctive, Urdu has two grammatically ambivalent forms with different endings:

+2P+Pl:O+Subjunct FutFlagPl; !used when addressing people of lower status, children

+2P+Pl:EN+Subjunct FutFlagPl; !used when addressing people of higher status, perhaps encode courtesy?

The two forms differ in their discourse functions. Usually, the first form is used only when addressing children or people of lower social status; the second form, in contrast, is needed mainly for addressing people of equal or higher social status.

4.2 Infinitives vs. Verbal Nouns

Another peculiarity of Urdu verbs is that the infinitive can actually serve as a noun.[EXAMPLES]The resulting questions then are whether these forms should be analyzed in the verb or in the noun lexicon and how the verbal origin of the nouns can be annotated in the sublexical rules. We put forward the following solutions to these questions. Any verbal morphology should be included in the verb lexicon, including infinitives, no matter as to their actual function. As to the sublexical rules, the infinitival suffix +Inf could be given the following annotation:

+Inf (^NTYPE verbal).

This would result in providing a value verbal for the feature NTYPE, which tells us that the origin of the noun is a verb. The feature must of course be present to receive the value, which is intentional, because the feature is only present in the syntax if the string is analyzed as a noun.

4.3 Deadjectival Nouns

Like Urdu nouns, adjectives are divided into two groups: those which have suffixes that change to show gender and number (marked adjectives) and those which do not (unmarked adjectives).

Many adjectives in Urdu can also be used as nouns, which brings up the question, whether this should be dealt with in the morphology or the syntax of the grammar. My first intention was to deal with this phenonema solely in the morphology by creating different continuation classes with the according tags.

As previous attempts to handle the problem have shown, this turns out to be quite difficult when implemented in the grammar (or why did Martin Forst implement this in the syntax instead of the morphology?)

A very simple possibility would be: A : (^ SUBJ) = !, but the morphology should be changing as well, or can that be done in the syntax? Problem in morph: syntax only realizes Adj at the mo, not Noun -ٲ probably my mistake

References

- Abbas Malik, M.G. 2006. Hindi Urdu machine transliteration system. MSc Thesis, Paris 7.
- Abbi, Anvita. 1991. *Reduplication in South Asian Languages. An Areal, Topological and Historical Study..* New Delhi: Allied.
- Beesley, Kenneth and Lauri Karttunen. 2003. *Finite State Morphology*. Stanford, CA: CSLI Publications.
- Beg, Mirza Khalil A. 1988. *Urdu Grammar: History and Structure*. New Delhi: Bahri Publications.
- Butt, Miriam, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, and Christian Rohrer. 2002. The Parallel Grammar project. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING), Workshop on Grammar Engineering and Evaluation*, pages 1–7. Taipei, Taiwan.
- Butt, Miriam and T.Holloway King. 2007. Urdu in a parallel grammar development environment. *Language Resources and Evaluation* Special Issue on Asian Language Processing: State of the Art Resources and Processing.
- Butt, Miriam, Tracy Holloway King, María-Eugenia Niño, and Frédérique Segond. 1999. *A Grammar Writer's Cookbook*. Stanford, CA: CSLI Publications.
- Dalrymple, Mary. 2001. *Lexical Functional Grammar*. New York: Academic Press.
- Glassman, Eugene H. 1977. *Spoken Urdu*. Lahore: Nirali Kitaben.
- Harris, Alice C. and Lyle Campbell. 1995. *Historical Syntax in Cross-Linguistic Perspective*. Cambridge: Cambridge University Press.
- Hopper, Paul J. and Elizabeth C. Traugott. 1993. *Grammaticalization*. Cambridge: Cambridge University Press.
- Kaplan, Ronald M., John T. Maxwell III, Tracy Holloway King, and Richard Crouch. 2004. Integrating finite-state technology with deep LFG grammars.

In *Proceedings of the European Summer School on Logic, Language and Information (ESSLLI) Workshop on Combining Shallow and Deep Processing for NLP*.

Kellogg, S. H. 1893. *Grammar of the Hindi Language*. Delhi: Munshiram Manoharlal Publishers Pvt. Ltd. Second Edition, reprinted 1990.

McGregor, R.S. 1968. *The Language of Indrajit of Orchā*. Cambridge: Cambridge University Press.

Rahman, S. and Sarmad Hussain. 2003. Development of character based urdu nastaleeq font. *Asian Media and Communication Bulletin* 33(2). AMIC, School of Communication and Information, Nanyang Technological Univ., Singapore.

Wells, J.C. 1997. SAMPA computer readable phonetic alphabet. In D. Gibbon, R. Moore, and R. Winski, eds., *Handbook of Standards and Resources for Spoken Language Systems*. Berlin and New York: Mouton de Gruyter. Part IV, section B.