Implementing Urdu Grammar as Open Source Software

Muhammad Humayoun
Department of Mathematics
University of Savoie
France
mhuma@univ-savoie.fr

Harald Hammarström
Aarne Ranta
Department of Computer Science
Chalmers University of Technology
Sweden
{harald2, aarne}@cs.chalmers.se

CLT 07, University of Peshawar
Pakistan
Introduction

- *Indo-European* → *Indo-Iranian* → *Indo-Aryan*

- Written from right to left using Perso-Arabic Script.

- Grammar and Vocabulary influenced by Arabic, Persian and the native languages of South Asia

- Widely Spoken in Pakistan, Jammu & Kashmir and India

- Also spoken all over the world due to big South Asian Diaspora

- **Urdu-Hindi**: share grammar, almost all phonology and lot of vocabulary

- Urdu-Hindi together is the second most widely spoken language (Native + second language)
Contribution

- **Orthography component**: A Unicode Infrastructure to accommodate Perso-Arabic script of Urdu

- **Morphology component**:
  - A type system that covers the language abstraction completely
  - An inflection engine that covers word-and-paradigm morphological rules for all word classes

- **Lexicon**: Automatically extracted, 4,816 words generating 137,182 word forms.

- **Grammar component**: A small fragment of syntax
Plan

- Urdu Orthography
- Urdu Morphology
- The Lexicon
- Urdu Syntax
- A complete example
- Conclusion & Future Work
Plan

- Urdu Orthography
- Urdu Morphology
- The Lexicon
- Urdu Syntax
- A complete example
- Conclusion & Future Work
Urdu Orthography

- An alphabet of 57 letters and 15 diacritic marks
- The use of diacritic marks: optional
- Morphology and the lexicon saved in ASCII characters
  - Reusability for Hindi in future, by adding a lexicon and the transliteration scheme
  - Easy manipulation on different platforms
Urdu Orthography

- Transliteration scheme (Transliterator)
  - Unicode value of each letter is mapped with a unique ASCII string
  - Clear, strict and reversible

- Useful tools
  - Urdu Extractor
  - Keyboard input method

- A GUI application

- Implemented in Java by using ICU4J and Swing packages
public class UrduUnicode
{
    public static final char alif = \u0627;
    public static final char bay = \u0628;
    public static final char pay = \u067e;
    ....
}

public class UrduRoman
{
    public static final String alif = "a";
    public static final String bay = "b";
    public static final String pay = "p";
    ....
}

private static final String unicode_to_Roman_rules =
    UrduUnicode.alif + ">" + UrduRoman.alif + ",;" +
    UrduUnicode.bay + ">" + UrduRoman.bay + ",;" +
    ....

public static final Transliterator unicode_to_roman =
    Transliterator.createFromRules("RomanUrdu-Unicode", unicode_to_Roman_rules, 0);

String romanText =
    TransliteratorUr._ur.unicode_to_roman.transliterate("Unicode Text");
# Urdu Orthography - Transliteration

### Examples:

<table>
<thead>
<tr>
<th>Urdu</th>
<th>Transliteration</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kitab</td>
<td>k(i)tab</td>
<td>book</td>
</tr>
<tr>
<td>koshish</td>
<td>k(a)wX(i)X</td>
<td>struggle</td>
</tr>
<tr>
<td>bulaọ</td>
<td>b(o)law^</td>
<td>to call</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arabic</th>
<th>Transliteration</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ك</td>
<td>k</td>
<td>(a)</td>
</tr>
<tr>
<td>ئ</td>
<td>(i)</td>
<td></td>
</tr>
<tr>
<td>ت</td>
<td>t</td>
<td>X</td>
</tr>
<tr>
<td>ا</td>
<td>(o)</td>
<td></td>
</tr>
<tr>
<td>ب</td>
<td>a</td>
<td>w^</td>
</tr>
</tbody>
</table>
Plan

- Urdu Orthography
- Urdu Morphology
- The Lexicon
- Urdu Syntax
- A complete example
- Conclusion & Future Work
Morphology Development
Current state of art

- Special-purpose languages based on finite-state technology

- The most well-known among others: XFST, LexC (Xerox)
  - **Implementation**: regular expressions, string types
  - **Complexity**: linear runtime, advance compilation techniques
  - **Efficiency**: fast, but large networks

- Non-regular formalisms might be difficult to implement efficiently enough
Some open questions

- Does the direct coding allow to implement one’s linguistic abstraction adequately?
  - Expressiveness

- Is such model extensible and reusable?
  - How much cost to add a lexical item?
  - Will refinement of information require global re-design
  - Will it cause inconsistencies

- How can it be integrated into applications
  - API and user Interfaces, Software Localization, modularity?
Morphology is implemented in Functional Morphology (FM)

An open source toolkit or domain embedded language for morphology development in Haskell

Designed by Markus Forsberg & Aarne Ranta
- Chalmers University of Technology, Sweden
- Sept 2004, International Conference on Functional Programming

Inspired by Gérard Huet’s Zen toolkit
- Computational processing of Sanskrit, 2002

Idea: Dealing with grammars as reusable software libraries
Functional Morphology (cont)

- Haskell: a functional programming language
  - High level of abstraction
  - Higher-order functions
  - Type classes
  - Polymorphism
  - These features: good for capturing linguistic generalizations

- Functional Morphology treats
  - The part of speech (word classes) as data types
  - Their Inflection as finite functions

- Tools (API functions, Analyzer, Synthesizer, Exporter)
Morphology + Orthography

**Urdu Script** (Unicode enabled Urdu)

**Transliteration**

**ASCII / Roman Urdu**

**Language Dependant (Urdu)**

**Morphology (Types, Rules, Lexicon)**

**Language Independent Module**

**FM API**

**Dictionary format**

**Analyzer**

**Synthesizer**

**Exporter**

**Functional Morphology Toolkit**

- XFST and LexC
- GF (Grammatical Framework)
- XML
- SQL
- Full-form lexicon, tables and LATEX
Nouns in Urdu: type system

- Urdu Noun Inflects in **Number** (*Singular, Plural*) and **Case** (*Nominative, Oblique, Vocative*)

  ```haskell
data Number  =  Singular | Plural
               deriving (Show, Eq, Enum, Ord, Bounded)

data Case    =  Nominative | Oblique | Vocative
               ....

data NounForm = NF Number Case
               ....

type Noun     =  NounForm → Str
               ....
```

- Inherent parameter: **Gender** (*Masculine, Feminine*)

  ```haskell
data Gender   =  Masculine | Feminine
               deriving (Show, Eq, Enum, Ord, Bounded)
```
Nouns in Urdu

- Nouns are divided into 15 groups based on their inflection.
- Masculine & Feminine proper names: 1 group for each.
- A group as running example:
  - Singular masculine nouns ending with (ا, ɑ), (ہ, h) and (ع, e / ئ / ء)
- Making:
  - If a word ends with letter (ا, ɑ) or (ہ, h) then:
    - **Plural nominative, singular oblique**: last letter is replaced by (ے, e)
    - **Plural oblique**: the last letter is replaced by (وں, oŋ)
    - **Plural vocative**: last letter is replaced by (و, o)
  - If a word ends with (ع, e / ئ / ء): above mentioned letters will be added at the end without replacing any existing letter.
# Nouns in Urdu: inflection engine

## Example Noun: (لڑکا, boy)

<table>
<thead>
<tr>
<th></th>
<th>Nominative</th>
<th>Oblique</th>
<th>Vocative</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Singular</strong></td>
<td>لڑکا</td>
<td>لڑکے</td>
<td>لڑکے</td>
</tr>
<tr>
<td><strong>Plural</strong></td>
<td>لڑکے</td>
<td>لڑکوں</td>
<td>لڑکو</td>
</tr>
</tbody>
</table>

**noun_lRka :: DictForm → Noun**

`noun_lRka lRka nf = mkNoun sg pl sg_obl pl_obl sg_voc pl_voc nf`

where

- `sg = lRka`
- `pl = lRk ++ "E"`
- `sg_obl = pl`
- `pl_obl = lRk ++ "wN"`
- `sg_voc = pl`
- `pl_voc = lRk ++ "w"`
- `lRk = if (end =="e") then lRka else (tk 1 lRka)`
- `end = dp 1 lRka`
Nouns in Urdu: inflection engine

- An general function for the Inflection table of nouns

```plaintext
mkNoun::String→String→String→String→String→String→Number→Case→String
mkNoun  sg  pl  sg_Obl  pl_Obl  sg_Voc  pl_Voc  n  c =
case n of
  Singular → case c of
    Nominative        → sg
    Oblique           → sg_Obl
    Vocative          → sg_Voc
  Plural → case c of
    Nominative        → pl
    Oblique           → pl_Obl
    Vocative          → pl_Voc
```

<table>
<thead>
<tr>
<th></th>
<th>Nom</th>
<th>Obl</th>
<th>Voc</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sg</strong></td>
<td>larka</td>
<td>larki</td>
<td>larka</td>
</tr>
<tr>
<td><strong>Pl</strong></td>
<td>larko</td>
<td>larkon</td>
<td>larko</td>
</tr>
</tbody>
</table>
Nouns in Urdu: inflection engine

- An interface function for this group of nouns
  \[ n1 :: \text{DictForm} \rightarrow \text{Entry} \]
  \[ n1\ df = \text{masculine}\ (\text{noun}\_\text{IRka}\ df) \]

  *\text{DictForm}: a string type
  *\text{masculine}: a function for masculine words

- Defined in the Lexicon:
  \[ n1\ l(a)R'ka \] (ləɽkɑ, َْرُ َْکا)
  \[ n1\ b(o)r'q(a)e \] (bʊrkɑ, ٌرُقُ ٌ برْقِ)
  \[ n1\ p(a)r'd(a)h \] (pəɾdɑ, َْرُدُ َْ پِردَه)

  …
Nouns in Urdu: inflection engine

An interface function for this group of nouns

\[ n1 :: \text{DictForm} \rightarrow \text{Entry} \]
\[ n1\,df = \text{masculine (noun_IRka df)} \]

*DictForm*: a string type

*masculine*: a function for masculine words

Defined in the Lexicon:

\[ n1\,l(a)R'ka \quad (ləɽkɑ, \text{ لَرْكَا}) \]
\[ n1\,b(o)r'q(a)e \quad (bʊrka, \text{ برَقَع}) \]
\[ n1\,p(a)r'd(a)h \quad (pərḍɑ, \text{ پرَده}) \]

…
**Nouns in Urdu: inflection engine**

**Some other noun-groups in the Lexicon**

- **Singular masculine nouns ending with** (ان, آنا)
  
  n2 k(o)n‘waN (Well, کوناں, كُنواں)
  
  n2 d(o)|hwaN (smoke, دُہوان, دُهوان)

- **Singular masculine nouns not ending with** (آ, آتا), (ه, ها), (ع, عا / ئا / ئة) and (ان, انا)

  n3 m(a)r’d (Man, مرد, مرد)
  
  n3 Aftab (Sun, آفتاب, آفتاب)

<table>
<thead>
<tr>
<th></th>
<th>Singular</th>
<th>Plural</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nom</td>
<td>کوناں</td>
<td>کونوں</td>
</tr>
<tr>
<td>Obl</td>
<td>کونوں</td>
<td>کونوں</td>
</tr>
<tr>
<td>Voc</td>
<td>کونوں</td>
<td>کونوں</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Singular</th>
<th>Plural</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nom</td>
<td>مرد</td>
<td>مرد</td>
</tr>
<tr>
<td>Obl</td>
<td>مرد</td>
<td>مردوں</td>
</tr>
<tr>
<td>Voc</td>
<td>مرد</td>
<td>مردوں</td>
</tr>
</tbody>
</table>
Nouns in Urdu: inflection engine

Some other noun-groups in the Lexicon

- **Singular feminine nouns ending with (ي, y)**
  
n4 k(o)r'sy (Chair, kursi, کرسي)

- **Singular feminine nouns ending with (ا, a), (آن, an), (ون, on)**
  
n5 maN (Mother, man, مان)

- **Singular feminine nouns ending with (يا, ya)**
  
n6 g(o)R'ya (Doll, gujriya, گریہا)

- **... n7 .... n14 ....**

- **Worst-case function for Noun**
  
n15 *singular* singular_obl *singular_voc* plural plural_obl plural_voc
# Urdu Verbs

<table>
<thead>
<tr>
<th>Root</th>
<th>Infinitive</th>
<th>Oblique</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Intransitive / Transitive / Ditransitive etc</strong></td>
<td>bən بن</td>
<td>bənəنا بننا</td>
</tr>
<tr>
<td></td>
<td>to build (by unknown)</td>
<td></td>
</tr>
<tr>
<td><strong>Direct Causative</strong></td>
<td>bənəنا بننا</td>
<td>bənəنا بننا</td>
</tr>
<tr>
<td></td>
<td>to build (by self)</td>
<td></td>
</tr>
<tr>
<td><strong>Indirect Causative</strong></td>
<td>bənvəنا بنوا</td>
<td>bənvəنا بنوا</td>
</tr>
<tr>
<td></td>
<td>to build (by third person)</td>
<td></td>
</tr>
</tbody>
</table>

We divide verbs in the following categories:

- Basic stem form, direct & indirect causatives exist
- Only Basic stem form exists
- Basic stem form & direct causative form exist
- Basis stem form & indirect causative form exist
- **6 groups** have been implemented for verbs
Urdu Verbs

- Urdu verb inflects in:
  - Gender, Number
  - Person
    - First
    - Second \{casual, familiar, respectful\}
    - Third \{near, distant\}
  - Tense
    - Subjunctive
    - Perfective
    - Imperfective
**Urdu Verbs: type system**

- **Category: Basic stem form, direct & indirect causatives exist**

  ```haskell
  type Verb = VerbForm → Str
  data VerbForm =
      VF Tense Person Number Gender |
      Caus1 Tense Person Number Gender |
      Caus2 Tense Person Number Gender |
      Inf | Caus1_Inf | Caus2_Inf |
      Inf_Fem | Caus1_Inf_Fem | Caus2_Inf_Fem |
      Inf_Obl | Caus1_Inf_Obl | Caus2_Inf_Obl |
      Root | Caus1_Root | Caus2_Root |
  data Person = Pers1 |
      Pers2_Casual | Pers2_Familiar | Pers2_Respect |
      Pers3_Near | Pers3_Distant |
  data Tense = Subj | Perf | Imperf
  ```
An general function for the Inflection table

```
mkGenVerb :: DictForm → DictForm → DictForm → DictForm → DictForm → DictForm → Verb
mkGenVerb root r1 r2 vf caus1 caus2 (Root) = root
mkGenVerb root r1 r2 vf caus1 caus2 (Inf) = vf
mkGenVerb root r1 r2 vf caus1 caus2 (Inf_Obl) = (tk 1 vf) ++ "E"
mkGenVerb root r1 r2 vf caus1 caus2 (Inf_Fem) = (tk 1 vf) ++ "y"
mkGenVerb root r1 r2 vf caus1 caus2 (VF t p n g) = mkVAnalysis root t p n g
mkGenVerb root r1 r2 vf caus1 caus2 (Caus1 t p n g) = mkVAnalysis r1 t p n g
mkGenVerb root r1 r2 vf caus1 caus2 (Caus2 t p n g) = mkVAnalysis r2 t p n g
```
Urdu Verbs: inflection engine

\[ mkVAnalysis :: \text{String} \rightarrow \text{Tense} \rightarrow \text{Person} \rightarrow \text{Number} \rightarrow \text{Gender} \rightarrow \text{String} \]

\[ mkVAnalysis \text{ root tense p n g} = \]
\[ \text{case tense of} \]
\[ \text{Subjunctive} \rightarrow \text{case p of} \]
\[ \text{Pers1} \rightarrow \text{case n of} \]
\[ \text{Singular} \rightarrow \text{case g of} \]
\[ \_ \rightarrow \text{mkEnding b root } "^wN" \ "wN" \]
\[ \text{Plural} \rightarrow \text{case g of} \]
\[ \_ \rightarrow \text{mkEnding1 b root } "^yN" \ "yN" \]
\[ \_ \rightarrow \text{mkSubjunctive root p n g} \]
\[ \text{where} \]
\[ t = \text{dp 1 root} \]
\[ b = \text{inStr t } ["A","a","w"] \]

\[ \text{Perfective} \rightarrow \text{mkPerfective root p n g} \]

\[ \text{Imperfective} \rightarrow \text{case p of} \]
\[ \text{Pers2_Familiar} \rightarrow \text{case n of} \]
\[ \text{Singular} \rightarrow \text{case g of} \]
\[ \text{Masc} \rightarrow \text{root ++ } "^tE" \]
\[ \text{Fem} \rightarrow \text{root ++ } "^ty" \ + " + root ++ } "^tyN" \]

..............................................
..............................................
..............................................
Urdu Verbs: in lexicon

■ v4 bnna bnana bnnwana
  (بَنِنَا، بُنِانا، بُنَنَا)

Root: بَن
Inf: بَنَّا
Inf_Obl: بَنِي
Inf_Fem: بَنِی

VF Subj Pers1 Sg Masc/ Fem : بَنُنِی
VF Subj Pers1 Pl Masc/ Fem : بَنِئ
VF Subj Pers2_Casual Sg Masc/ Fem : بَن
VF Subj Pers2_Casual Pl Masc/ Fem : بَن

.....

Caus1 Root: بَن
Caus1 Inf: بَنَنا
Caus1 Inf Obl: بَنِنَا

Caus1 Subj Pers1 Sg Masc/Fem: بَنِئ
Caus1 Subj Pers1 Pl Masc/Fem: بَنِئ
Caus1 Subj Pers2_Casual Sg Masc/Fem: بَن
Caus1 Subj Pers2_Casual Pl Masc/Fem: بَن

.....

Caus2 Root: بَنَا
Caus2 Inf: بَنَانا
Caus2 Inf Obl: بَنَانَا

Caus2 Subj Pers1 Sg Masc/Fem: بَنَا
Caus2 Subj Pers1 Pl Masc/Fem: بَنَا
Caus2 Subj Pers2_Casual Sg Masc/Fem: بَنَا
Caus2 Subj Pers2_Casual Pl Masc/Fem: بَنَا
Other word classes

- **Adjectives**
  - **Marked** (Inflects in number, case and gender)
    - Ends with (أ، ɑ): ڏیلا، ڏیلی، ڏیلے : Blue
    - Ends with (آن، ئینغ): داېن، داېنغ : Right
  - **Unmarked**
    - No inflection: خوش : Happy
    - Ends with (ي، ی)، made from nouns or Adjectives
      - desi دیسی: local
    - Inflects in degree (Persian's style)
      - باد، بادتر، بادترین : bad, worse, worst

- **Adverbs**

- **The closed classes**
  - Pronouns, PostPositions, Particles, Interjunctions, Conjunctions, Negations, Questions and Numerals
Plan

- Urdu Orthography
- Urdu Morphology
- The Lexicon
- Urdu Syntax
- A complete example
- Conclusion & Future Work
The Lexicon

- A wide-coverage lexicon is a key part of any morphological implementation.
- Aim: to build a lexicon automatically with minimal human efforts.
- A tool **extract** is used which is provided with the Functional Morphology.
- It requires a **paradigm file** and a **corpus**.
- To build a **corpus**:
  - A reasonable amount of Urdu Unicode text was collected from the web (news and literature domain).
  - All the html tags & other non-related information were thrown away by a tool (developed with this work) and save the file as text file.
  - Urdu Unicode text is then converted into ASCII Urdu by transliteration tool.
- **The lexicon**: extracted by applying **paradigms** on **corpus**.
The Lexicon – Example Paradigm(1)

- Singular Feminine nouns not ending with (ا، a), (ن، N), (و، w)
- (كتاب, Kitāb, book), (گاجر, gājer, carrot)

regexp Not_awN = char* (char- ("a" | "N" | "w"));

paradigm n9 [x:Not_awN] =
  x { (x &
      (x+"yN" | x+"wN" | x+"w")
  )
};

<table>
<thead>
<tr>
<th></th>
<th>Nom</th>
<th>Oblique</th>
<th>Voc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sg</td>
<td>كتاب</td>
<td>Kitāb</td>
<td></td>
</tr>
<tr>
<td>Pl</td>
<td>كتابيين</td>
<td>Kitāben, (yN)</td>
<td>كتابون</td>
</tr>
</tbody>
</table>

n9 k(i)tab
n9 gajr
n9 kRwahT
n9 kar
.....
Verbs that has basic, direct & indirect causative forms:

paradigm v4 =
  x +"na" x+"ana" x+"wana"
  {
    x+"na" & (x+"ana" | x+"wana")
  };

v4 dyk|hna d(i)k|hana d(i)k|hwana (دِكَهُوْنا , دِكَهُوْنا , دِكَهُوْنا)
v4 bnna bnana bnwana (بنوَا, بنوَا, بنوَا)
The Lexicon: Problems

- Urdu is commonly written without or with a variant number of diacritic marks
  - A fundamental limitation to get a fully vocalized corpus
- Problem: having more versions per word with different diacritics
  - e.g. (کتاب, کتاب) and (کتاب, کتاب) for word (کتاب, book)
  - Point: We should save only one version per word with full diacritics
- Tokens with different diacritics are not always same words
  - e.g. (تیر, تیر, to swim) and (تیر, تیر, arrow)
  - Point: We should save all such words with full diacritics
The Lexicon Extraction - Results

- To assure the correctness:
  - Manually re-checking of the lexicon from word to word
  - Incorrect entries thrown away

- A fundamental limitation
  - The missing diacritics on partly vocalized words are not applied

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Lexicon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size (Words)</td>
<td>Extracted lexicon</td>
</tr>
<tr>
<td>1,520,000 (1.5 million)</td>
<td>9,126</td>
</tr>
<tr>
<td>Words containing Diacritics:</td>
<td>Words containing Diacritics:</td>
</tr>
<tr>
<td>23,696</td>
<td>632</td>
</tr>
<tr>
<td>Unique tokens:</td>
<td>Clean lexicon</td>
</tr>
<tr>
<td>63,700 (4.1%) **</td>
<td>4,816 (52.8%)</td>
</tr>
<tr>
<td></td>
<td>Words containing Diacritics:</td>
</tr>
<tr>
<td></td>
<td>415</td>
</tr>
</tbody>
</table>

**This conforms well to our intuition that high frequent items (postpositions, auxiliaries, particles and pronouns), account for most tokens in Urdu text.**
The Lexicon Extraction - Results

- Why so many incorrect entries?
- The strictness of rules in paradigm file: normal
  - Trade-off: quality vs. coverage

- Spelling mistakes:
  - Original Typos
  - Lack of spaces between words
  - Extra spaces inside words
  - Possible Reason: The use of Urdu on web is relatively new

- Foreign words:
  - Arabic – The verses of Holy Quran in religious text
  - Persian – Poetry in slightly old literary text
  - Lot of proper nouns and English words in the news domain
Plan

- Urdu Orthography
- Urdu Morphology
- The Lexicon
- Urdu Syntax
- A complete example
- Conclusion & Future Work
The Syntax

- Urdu an **SOV** (Subject Object Verb) language
- Relatively **free word order**
- A small fragment of syntax as a separate component on top of morphology by using **Grammatical Framework**

**Grammatical Framework (GF):**
- Grammar formalism based on type theory
- Programming language for defining grammars (formal + natural)
- Grammar = The Abstract syntax and Concrete syntax
- **Abstract syntax** = semantic conditions (correct syntactic structures / trees of a language)
- **Concrete syntax** = mapping abstract syntax into strings along-with the grammatical features for a language (and back, by reversibility)
The Overall Picture

Syntax in GF

GF Morphology + UTF-8 Lexicon
(auto generated code) + Preprocessing

Orthography Component
(GUI Application, tools & Transliteration)

Lexicon

Morphology Component in FM
The Syntax

In our Implementation: A sentence:
- Combination a noun phrase (NP) and a verb phrase (VP)
- Combination of two sentences by adding a conjunction in between
The Syntax

- **Abstract Syntax:**
  
  \[
  \text{fun UsePresS: NP } \rightarrow \text{ VP } \rightarrow \text{ S} ;
  \]

- **Concrete Syntax:**
  
  \[
  \text{UsePresS np vp =}
  \{\text{s = np.s ! Nom ++ vp.s ! Present ! np.p ! np.n ! np.g}\}
  \]

  is ko ḵṯabeṉ leni hēṉ, Ḣe/she is suppose to take the books

---

<table>
<thead>
<tr>
<th>Syntax Chain</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>DemPron → Num → CN → NP</td>
<td>ye ḫo ḵṯabeṉ, ye do ḵṯabeṉ, these two books</td>
</tr>
<tr>
<td>DemPron → PN → NP</td>
<td>vo Ali, ḭe only, that Ali</td>
</tr>
<tr>
<td>NP → PostP → CN → NP</td>
<td>is ko ḵṯabeṉ, Ḩe ko ḵṯabeṉ, to him the books</td>
</tr>
<tr>
<td>Verb_Aux → VP</td>
<td>hēṉ, bi̱ṉ, are</td>
</tr>
<tr>
<td>Verb → Verb_Aux → VP</td>
<td>leni ̱thiṉ, ̱thiṉ, was suppose to take</td>
</tr>
</tbody>
</table>

[More examples sentences](#)
Plan

- Urdu Orthography
- Urdu Morphology
- The Lexicon
- Urdu Syntax
- A complete example
- Conclusion & Future Work
A Complete Example

Transliteration: a(i)s kw ktabyN lyny hyN

Syntactic parsing:

UsePresS
  (UseNP (UsePron mayN_68) kw_18
       (UseN ktab_824))
  (UseVP lyna_2 hwna_0)

Syntax tree
The Overall Picture

Orthography Component
(GUI Application, tools & Transliteration)

Syntax in GF
GF Morphology + UTF-8 Lexicon
(auto generated code) + Preprocessing

Lexicon

Morphology Component in FM
Plan

- Urdu Orthography
- Urdu Morphology
- The Lexicon
- Urdu Syntax
- A complete example
- Conclusion & Future Work
Conclusion

■ Merits:
  ■ Functional Morphology proved to be a very good choice for implementing Urdu morphology
  ■ A comprehensive, reusable & elegant implementation for Urdu that covers the linguistic abstraction (morphology) adequately

■ Limitations:
  ■ Can’t deal with words that have spaces in between
  ■ A partly vocalized Lexicon
  ■ Run time system requires an exact match:
    ■ One cannot check if there exist orthographically different versions of a word
Future Work

- A component that matches the partly vocalized input words with the canonical words in the lexicon
- Algorithms to add missing diacritics on partly vocalized words
- A bigger lexicon
- Analysis of words that contains spaces in between
- A comprehensive implementation for syntax
- Implementation of Hindi, by adding a lexicon and a transliteration scheme
- The Next Big Step: Urdu-English Machine Translation
Some screen shots
Dictionary loaded: DF = 4807 and WF = 137078.

```
[ 'کتاب', 'k(i)tab'>
14. {کتاب, k(i)tab} (1074) N - NF Sg Nom - Fem
15. {کتاب, k(i)tab} (1074) N - NF Sg Obl - Fem
16. {کتاب, k(i)tab} (1074) N - NF Sg Voc - Fem
```

Tagger Mode: Please be patient! It will take some time to build the data structure first time...
Urdu Morphology

Functional Morphology v1.10
(c) Markus Forsberg & Rame Ranta 2004
under GNU General Public License.

Implementation for Urdu
(Muhammad Humayou 2006)
Chalmers University of Technology

Dictionary loaded: DF = 4807 and WF = 137078.

14. {ك (tab) 1074} N - NF Sg Nom - Fem
15. {ك (tab) 1074} N - NF Sg Obl - Fem
16. {ك (tab) 1074} N - NF Sg Voc - Fem

Tagger Mode: Please be patient! It will take some time to build the data structure first time...
Conversion from Urdu to Roman

Urdu Script

Roman Transliteration

Aj ky bat

Unicode

\u0622\u062c\u0020\u06a9\u06cc\u0020\u0628\u0627\u062a

Urdu Transliterator: Running...
Thanks for your attention

Questions / Comments

Homepage of the project

http://www.lama.univ-savoie.fr/~humayoun/UrduMorph/