An Urdu LFG Grammar

Miriam Butt

Universität Konstanz

February 2009, Kathmandu

Introduction

ParGram ("Parallel Grammar"): NLP project based on Lexical Functional Grammar (LFG)

- multilingual grammar development project
- large-scale, robust, parallel computational grammars
- so far:
  - larger grammars for English, German, French, Norwegian, Chinese and Japanese
  - smaller grammars for Bahasa Indonesian, Malagasy, Turkish and Welsh.

Introduction

Possible Applications:

- Machine Translation (made simpler because of deep analysis and parallelism across languages).
- Text Summarization (*parsing* of large corpora, *generation* of summaries)
- Question-Answer Systems (*parsing* of large corpora, *generation* of answers) — successful company built on this (POWERSET).

Introduction

Advantages of the ParGram Approach:

- LFG allows for a modular architecture:
  - morphology, syntax and semantics are encoded at independent levels providing necessary flexibility
  - each level of analysis uses different types of representations (e.g., trees vs. AVMs vs. logical formula)
  - all of the levels interact, accounting for interactions across modules
  - an LFG grammar is *reversible*: a grammar can be used for **both** parsing and generation

ParGram

Advantages of the ParGram Approach:

- The "parallel" in ParGram means:
  - Analyses should abstract away from language particular features as much as possible.
  - A common set of grammatical features is chosen based on common decisions across a range of differing languages.
  - The in-built multilingual perspective means one avoids pitfalls & shortcomings often found in monolingual NLP efforts.

ParGram

Particulars of Grammar Development

- XLE Grammar Development Platform (available by license from PARC)
- integrates: tokenizer (FST), morphological analyzer (FST), syntactic rules (LFG), transfer component (Prolog rewriting rules) used for machine translation and semantic construction
- XLE is written in C; powerful & efficient

Urdu grammar at Konstanz

- A small Urdu grammar has been developed at Konstanz.
- Urdu is the only South Asian language within ParGram; interesting from a typological point of view.
- **Research Question:** Can the existing small grammar be scaled up to a robust and large-scale grammar within the ParGram context?
- **Some Challenges:**
  - Massive use of complex predicates (about 30% of any text)
  - Free Word-Order, dropping of arguments (problem for generation)
  - Complex interaction between morphology, syntax and semantics (e.g., tense/aspect, case marking, reduplication, Ezafe construction)

Existing Resources

- Much work on some necessary basic resources has recently been done — most of it at CRULP in Lahore (fonts, corpora, dictionaries, POS-taggers, etc.)
- Some morphological analyzers have been worked on — however, while in principle these are stand-alone systems, the ParGram context assumes certain types of analyses. So we have to build our own.
- CRULP has worked on an LFG-based Urdu grammar. However, it was designed for parsing, not generation, so did not deal with word order or dropped arguments.

Urdu grammar at Konstanz

Therefore: a new project will start on scaling up the existing small Konstanz Urdu grammar and developing additional tools.

- 3 years of funding
- official start date: March 1st, 2009
- goal: reach broad coverage for Urdu grammar (both parsing and generation)

Tokenization and Transliteration

- We use the default tokenizer included in XLE at the moment.
- **Goal:** the Urdu grammar should be able to process Urdu, English and Devanagari (Hindi) text.
- Reason:
  - English words occur in Urdu texts quite frequently.
  - The major difference between Urdu and Hindi from an NLP perspective lies in the script, so we should be able to kill two birds with one stone.
- We therefore use ASCII in our morphological analyzer.

Transliteration

Urdu/Hindi

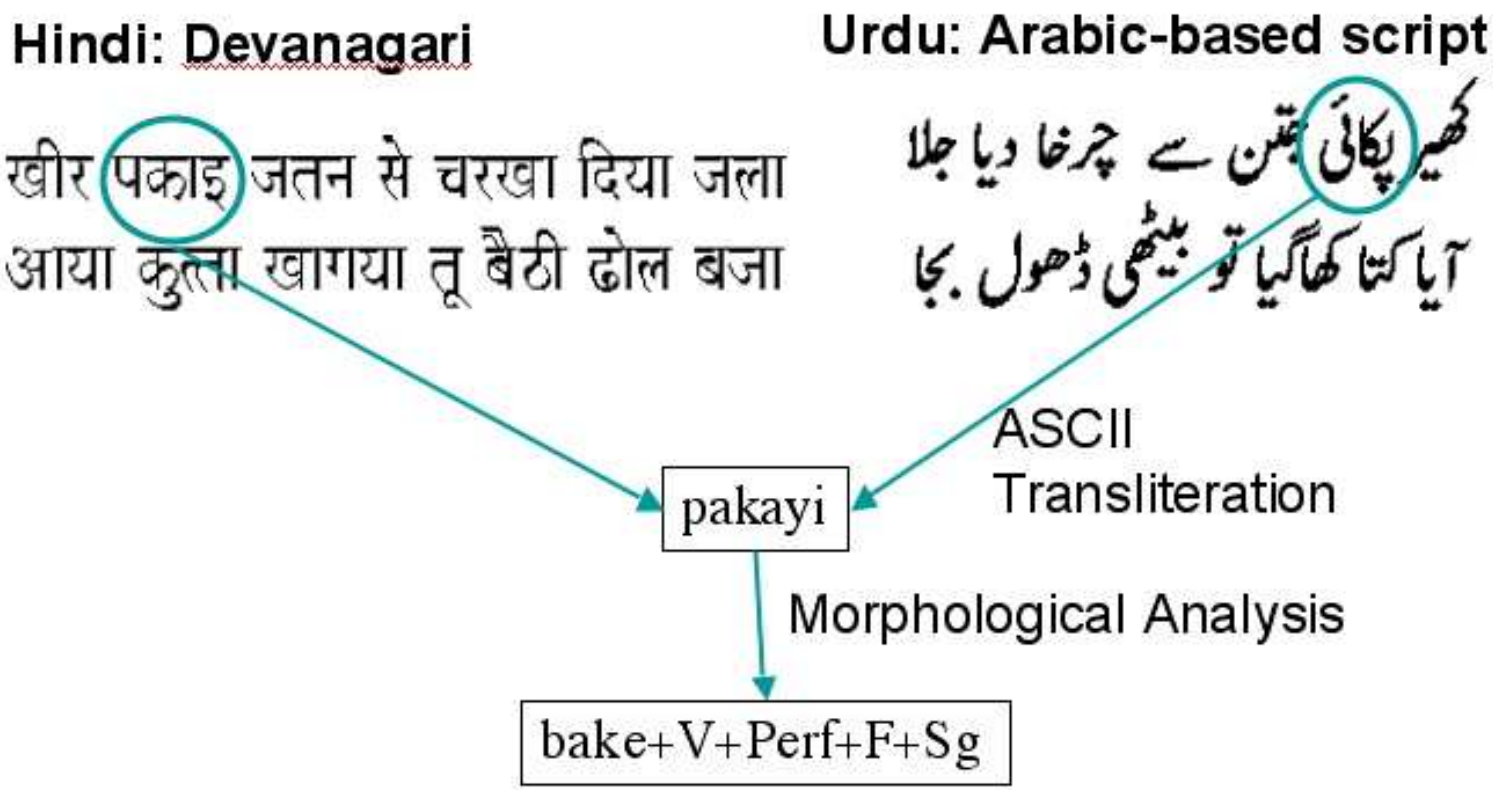
Example: Poetry and Riddles by Amir Khusro (1253-1325).

Hindi	Urdu
खीर पकाइ जतन से चरखा दिया जला	کھیر پکائی جتن سے چرخا دیا جلا
आया कुत्ता खागया तू बैठी ढोल बजा	آیا کتا کھاگیا تو بیٹھی ڈھول بجا

"You prepared the kheer (rice pudding) with much hard work
and lit up the lamp.
The dog came and ate it all,
nothing for you now than to sit and play the drum."

Transliteration

Two Scripts: One Analysis



Transliteration

- Currently we are integrating an XFST transliteration system from the Urdu Script to ASCII (developed at CRULP).
- This transliterator will eventually form part of the tokenizer.
  - transliterator will rewrite Urdu into ASCII (parsing) and back out again (generation)
  - transliterator will rewrite Hindi Devanagari into ASCII (parsing) and back out again (generation)

Overall Architecture

tokenizer & transliterator & morphology (XFST)

↓

syntax (c- and f-structure) [LFG proper]

↓

semantics (XFR ordered rewriting)

Overall Modular architecture of ParGram Urdu grammar

Syntax

- syntax component is at the core of Urdu grammar
- theoretical background: LFG
- well-studied ($\sim$ 30 years) framework with computational usability
- c- and f-structures used for syntactic representation
  - c-structure: basic constituent structure ("tree") and linear precedence ($\sim$ what parts belong together)
  - f-structure: encodes grammatical relations and functional information.

Syntax

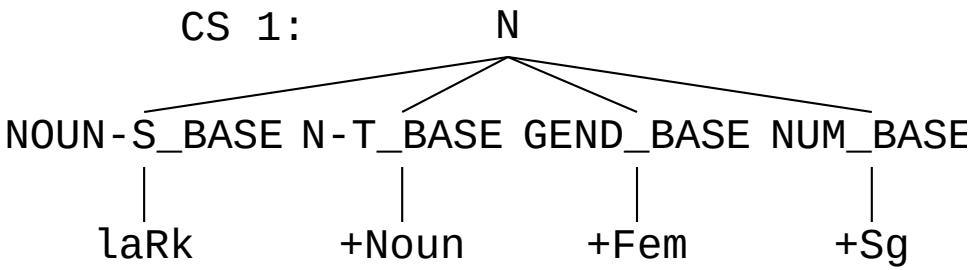
- current size: 40 phrase-structure rules, annotated for syntactic function (large grammars have about 360 rules)
- coverage (parsing only): basic clauses with free word order, verbal complex, tense and aspect, causative verbs, complex predicates

Morphology

- implemented using sophisticated and very fast, efficient finite-state machines (as you will see in tomorrow's tutorial)
- using this technology, the grammar can deal with the full range of inflectional and derivational morphology in Urdu, including difficult phenomena such as reduplication (kHAnA vAnA)
- functions as "black box", abstract *morphological tags* are provided as input to XLE
- the morphology-syntax interface is powerful and flexible (see reduplication and complex predicates).

Morphology

- sample output of morphological analyzer: MORPHOLOGY
  `laRk+Noun+Fem+Sg`
- tags are used as input for syntax component: INTERFACE
  `+Fem GEND xle @(GEND fem)`
  `+Sg NUM xle @(NUM sg)`
- sublexical features are displayed in the syntax: SYNTAX



"laRkI"

$$\begin{bmatrix} \text{PRED} & \text{'laRk'} \\ \text{NTYPE} & \begin{bmatrix} \text{NSEM} & [\text{COMMON count}] \\ \text{NSYN} & \text{common} \end{bmatrix} \\ 1 & \text{GEND fem, NUM sg, PERS 3} \end{bmatrix}$$

Sample Analyses

In what follows, we provide a discussion and sample analyses of some interesting/challenging phenomena within Urdu.

- Free Word Order
- Case Marking
- Complex Predicates and Causatives (demo)

Free Word Order

Urdu (and most of the other South Asian languages) have *free word order* (discourse driven, but free from the point of view of syntax).

- If so desired, encode the discourse functions at the level of i(nformation)-structure (Bengali demo).
- Encode free word order at syntax via the *shuffle operator* in LFG/XLE. The shuffle operator is notated with a comma (,).
  S $\longrightarrow$ NP*, PP*, V
- For generation, will need OT marks to only produce one of the many word orders.

Case Marking

- In most South Asian languages *case marking* plays a large role.
- They help determine the *grammatical relations* of a clause.
- Solution for Urdu:
  - encode the case markers as lexical items
  - have the case markers provide information about grammatical relations (and semantics, where relevant) via *inside-out* functional application.
- This solution has come to be known as *Constructive Case* (cf. Nordlinger 1998 for Australian languages).

Case Markers: Urdu dative/accusative *ko*

- When *ko* used as a dative (`option 1`), it can be associated either with subjects (SUBJ, `option 1a`)) or indirect objects (OBJ_{theta}, `option 1b`).
- As an accusative (`option 2`), the *ko* denotes semantic specificity or definiteness and is restricted to objects (OBJ).

(1) ko K *

$$
\begin{array}{ll}
\{\ (\uparrow \text{CASE}) = \text{dat} & \texttt{option 1} \\
\quad \{\ (\text{SUBJ}\uparrow) & \texttt{option 1a} \\
\quad\ |(\text{OBJth}\uparrow)\ \} & \texttt{option 1b} \\
|(\uparrow \text{CASE}) = \text{acc} & \texttt{option 2} \\
\quad (\text{OBJ}\uparrow) & \\
\quad (\uparrow \text{SEM-PROP SPECIFIC}) = +\}. &
\end{array}
$$

Case Markers: Urdu ergative *ne*

- The complete entry for the ergative case marker *ne* is shown in (2).
- It is much simpler than the entry for the dative/accusative *ko*. This is primarily because ergatives can only appear on subjects.

(2) ne K * $(\uparrow \text{CASE}) = \text{erg}$
$(\text{SUBJ} \uparrow)$

Complex Predicates: The Most You Can Do

(3) tara=ne amu=ko bacce=se hat^{h}i
Tara=Erg Amu=Dat child.Obl=Inst elephant.M.Sg.Nom
pinc kar-va le-ne di-ya
pinch do-Caus take-Inf.Obl give-Perf.M.Sg
'Tara let Amu have the elephant pinched (by the child).'

- All of the verbal stuff forms one **monoclausal** predicate (one SUBJ, one OBJθ, one OBL, one OBJ, no embeddings).

Tests for Complex Predication

Some tests for monoclausality in Urdu are:

- Verb agreement (agrees with object if subject is marked/non-nominative)
- Anaphora Resolution
- Control

Verb Agreement

If you change the gender of the object, then the verb agrees with it, even though it is not a "semantic" argument of the finite verb (i.e., is not licensed directly by the finite verb).

(4) tara=ne amu=ko bacce=se **billi**
Tara=Erg Amu=Dat child.Obl=Inst cat.F.Sg.Nom
pinc kar-va le-ne **di**
pinch do-Caus take-Inf.Obl give.Perf.F.Sg
'Tara let Amu have the elephant pinched (by the child).'

Argument Structure Approach

- Traditional Theoretical Analyses of Complex Predication within LFG rely heavily on Argument Structure Composition and Linking Principles which map from Argument Structure to Grammatical Functions.
- This theoretical approach has not been taken up readily within computational linguistics — this is primarily because there is still no single stable analysis.

Doing without Argument Structure: The Restriction Operator

- Computational Linguistic approaches within LFG have instead tended to make use of the Restriction Operator introduced by Kaplan and Wedekind (1993).
- This solution requires only the manipulation of f-structures (so no projectional complication in terms of a-structures).
- **But:** the first solution proposed involved a huge amount of undesirable lexical stipulation and cannot account for the full combinatory power of complex predicate formation (Butt 1994).

The Restriction Operator and Syntactic Complex Predicates

- Butt, King, and Maxwell (2003) introduce (and implement) a solution which combines:
  - the Restriction Operator
  - the ability to *integrate* argument structures to form a **complex** PRED.
- The Restriction Operator can thus manipulate the subcategorization frame of a complex predicate.

The Restriction Operator and Syntactic Complex Predicates

- Composition of two argument structures, but a monoclausal f-structure (Butt 1995).
  - In (5b): main verb 'cough' and light verb 'give'.
  - The "lettee" (Nadya) is a dative marked OBJ_θ.

(5) a.
nAdyA	kHANs-I
Nadya.Nom	cough-Perf.F.Sg

'Nadya coughed.'

b.
yassIn=nE	nAdyA=kO	kHANs-n-E	dI-yA
Yassin=Erg	Nadya=Dat	cough-Inf-Obl	give-Perf.M.Sg

'Yassin let Nadya cough.'

The Urdu Permissive

An example with a transitive main verb ('make').

(6) a. nAdyA=nE gHar banA-yA
Nadya=Erg house.Nom make-Perf.M.Sg
'Nadya made a house.'

b. yassIn=nE nAdyA=kO gHar banA-n-E dI-yA
Yassin=Erg Nadya=Dat house.Nom make-Inf-Obl give-Perf.M
'Yassin let Nadya make a house.'

- From the perspective of a restriction analysis, the permissive:
  - "adds" a new subject
  - "demotes" the other verb's subject to a dative-marked indirect object
- Sample lexical entries for 'give' ((7)) and 'make' ((8)).

(7) (↑ PRED)='dE<(↑ SUBJ),%PRED2>'

(8) (↑ PRED)='banA<(↑ SUBJ),(↑ OBJ)>'

- Restriction allows f-structures and predicates to be manipulated in a controlled and detailed fashion.
- The restriction operator can be applied to an f-structure with respect to a certain feature in order to arrive at a restricted f-structure which does not contain that feature.
- Example: Restrict out the embedded subject of the composed PRED.

(9) PRED 'dE<SUBJ,'banA<SUBJ,OBJ>'>'

- The analysis of complex predicates uses restriction as part of the f-structure annotations on phrase structure rules.

(10)

$$\begin{array}{lcc} & (banAnE) & (dIyA) \\ V \longrightarrow & V & Vlight \\ & \downarrow\backslash\textsc{pred}\backslash\textsc{subj}\backslash\textsc{obj-go}=\uparrow\backslash\textsc{pred}\backslash\textsc{subj}\backslash\textsc{obj-go} & \uparrow=\downarrow \\ & (\uparrow \textsc{pred arg}2)=(\downarrow \textsc{pred}) & \\ & (\downarrow \textsc{vform}) =c \text{ inf} & \\ & (\uparrow \textsc{obj-go})=(\downarrow \textsc{subj}) & \\ & (\uparrow \textsc{subj pred}) & \end{array}$$

Permissive Restricted F-structures

- Final complex f-structure:
  - the predicates *dE* 'give' and *banA* 'make' have been composed;
  - the "embedded" SUBJ 'Nadya' has been restricted out as part of the composition.

(11)
$$\begin{bmatrix} \text{PRED} & \text{'dE<SUBJ,'banA<OBJ-GO,OBJ>'>'} \\ \text{SUBJ} & [\ \text{PRED 'Yassin'}\ ] \\ \text{OBJ-GO} & [\ \text{PRED 'Nadya'}\ ] \\ \text{OBJ} & [\ \text{PRED 'gHar'}\ ] \\ \text{TNS-ASP} & [\ \text{ASP perf}, \quad \text{TENSE pres}\ ] \end{bmatrix}$$

C-structure Role in Restriction

- The Urdu CP analysis and Wedekind and Oersnes's analysis of the Danish passive crucially rely on the **syntactic, phrase-structure compositional** aspects of these phenomena.
- There must be a **c-structure node** on which to put the restriction annotation that alters the valency of the verb, creating the final f-structure.

The Urdu Causative and Restriction

- The Urdu causative is created morphologically by affixation.
- **Question:** Can Restriction be extended to provide a uniform analysis of valency changing operations?

An Example

- Agentive transitives have an instrumental causee.

(12) a.

yassin=nE paodA kAT-A
Yassin=Erg plant.M.Nom cut-Perf.M.Sg
'Yassin cut the plant.'

b.

nAdyA=nE yassIn=sE paoda kaT-**vA**-yA
Nadya=Erg Yassin=Inst plant.M.Nom cut-**Caus**-Perf.M.Sg
'Nadya had the plant cut by Yassin.'

F-structures for Causatives

- The basic f-structure for a non-causative verb

(13)
$$\begin{bmatrix} \text{PRED} & \text{'cut}\langle \text{SUBJ,OBJ} \rangle' \\ \text{SUBJ} & [\ \text{PRED 'Yassin'}\ ] \\ \text{OBJ} & [\ \text{PRED 'plant'}\ ] \end{bmatrix}$$

- The basic f-structure for the resulting causative

(14)
$$\begin{bmatrix} \text{PRED} & \text{'cause}\langle \text{SUBJ,'cut}\langle \text{OBL,OBJ} \rangle' \rangle \\ \text{SUBJ} & [\ \text{PRED 'Anjum'}\ ] \\ \text{OBJ-GO} & [\ \text{PRED 'Yassin'}\ ] \\ \text{OBJ} & [\ \text{PRED 'plant'}\ ] \end{bmatrix}$$

Causative Morphology

- The structures in (13) and (14) can be related via the same type of restriction rules used to analyze complex predicates.
- Restriction must take place within the formation of the lexically causativized verb.
- The integration of finite-state style morphologies (Beesley and Karttunen 2003) into the LFG architecture provides a way to do this.

- The Urdu morphology associates a surface form with a lemma and a set of morphological tags.

(15) a. likHA ⇔ likH +Verb +Perf +Masc +Sg
b. likHAyA ⇔ likH +Verb +Caus1 +Perf +Masc +Sg
c. likHvAyA ⇔ likH +Verb +Caus2 +Perf +Masc +Sg

C-structure Rules for Causatives with Restriction

- The lemma and morphological tags are parsed by c-structure sublexical rules (Kaplan et al. 2004).
- The sublexical rules are formally identical to standard c-str rules.
- The *+Caus* tag provides a phrase-structure locus for the restriction operator.

Causatives via Restriction and Predicate Composition

- Causative annotated sublexical c-structure rule

(16) $V \rightarrow$ Vstem CauseMorph

$\downarrow\backslash\text{PRED}\backslash\text{SUBJ}=\uparrow\backslash\text{PRED}\backslash\text{SUBJ}$ $\quad \uparrow=\downarrow$

$(\downarrow\text{SUBJ})= \{ \ (\uparrow\text{OBJ-GO})$

$\mid (\uparrow\text{OBJ})$

$\mid (\uparrow\text{OBL}) \ \}$

$(\uparrow \ \text{PRED} \ \text{ARG2})=(\downarrow \ \text{PRED})$

Complex Predicates

Conclusion:

- Applying the Restriction Operator to Complex Predication has proven to be successful and allows for a generalized approach across morphology and syntax.
- One drawback is that one cannot so far generate when the Restriction Operator is in use.

Semantics

- f-structures within XLE are coded in Prolog
- for semantics, we take Prolog code and apply ordered rewrite rules (XFR) on it
  - reasonable approach, as f-structures are equivalent to quasi-logical forms
- input f-structure is consumed step by step by the rewrite rules
- XLE produces a semantic form as output

Semantics

- world knowledge may also be included
- English ParGram grammar uses WordNet as knowledge base
- back to Powerset: this is where we want to get
  - build up semantic system to construct meaning out of f-structures
  - integrate external semantic knowledge bases to reach broad coverage

Semantics

Semantics of *nAdyA hasI* 'Nadya laughed':

```
xfr(
% Choices:
 [],
% Equivalences:
 [],
% Equalities:
 [],
% Facts:
 [
  cf(1, context_head(t,has:n(7,'**'))),
  cf(1, in_context(t,cardinality(nAdyA:n(1,'**'),sg))),
  cf(1, in_context(t,proper_name(nAdyA:n(1,'**'),name,nAdyA))),
  cf(1, in_context(t,role(sem_subj,has:n(7,'**'),nAdyA:n(1,'**')))),
  cf(1, original_fsattr('SUBJ',has:n(7,'**'),nAdyA:n(1,'**'))),
  cf(1, original_fsattr(gender,nAdyA:n(1,'**'),'-')),
  cf(1, original_fsattr(human,nAdyA:n(1,'**'),'-')),
  cf(1, skolem_byte_position(has:n(7,'**'),7,9)),
  cf(1, skolem_byte_position(nAdyA:n(1,'**'),1,5)),
  cf(1, skolem_info(has:n(7,'**'),has,verb,verb,n(7,'**'),t)),
  cf(1, skolem_info(nAdyA:n(1,'**'),nAdyA,name,name,n(1,'**'),t))
        ],
)
```

Outlook

- with respect to the new Urdu project at Konstanz, we aim at developing further resources
- the semantics system has to be further developed
- we plan on creating a treebank for Urdu based on LFG annotation
- we also aim at integrating statistical tools, helpful for disambiguation between structures and improving robust parsing