Urdu and the Modular Architecture of ParGram

Tina Bögel, Miriam Butt, Annette Hautli, Sebastian Sulger

Universität Konstanz

CLT 2009, Lahore
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.
Possible Applications:

- Machine Translation (made simpler because of deep analysis and parallelism across languages).
Introduction

Possible Applications:

- Machine Translation (made simpler because of deep analysis and parallelism across languages).
- Text Summarization (parsing of large corpora, generation of summaries)
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).
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
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.
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 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.
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?
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:**
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)
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)
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)
Much work on some necessary basic resources has recently been done — most of it at CRULP (fonts, corpora, dictionaries, POS-taggers, etc.)
Existing Resources

- Much work on some necessary basic resources has recently been done — most of it at CRULP (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.
Existing Resources

- Much work on some necessary basic resources has recently been done — most of it at CRULP (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.
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

- 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)
\[\downarrow\]
syntax (c- and f-structure) \(\rightarrow\) prosody (p-structure)
\[\downarrow\]
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 (∼ 30 years) framework with computational usability
- C- and f-structures used for syntactic representation
  - C-structure: basic constituent structure (“tree”) and linear precedence (∼ what parts belong together)
  - F-structure: encodes grammatical relations and functional information.
Syntax

CS 1: ROOT

S

"nAdyA hasI"

KP VCmain

| PRED | 'has<1:nAdyA>'
| PRED | 'nAdyA'

NP Vmain

| NTYPE | NSEM [PROPER PROPER-TYPEName]
| SUBJ | NSYN proper
| SEM-PROP | SPECIFIC+
| CASE | nom, GEND fem, NUM sg, PERS 3
| CHECK | _VMORPH [MTYPE inf]
| _RESTRICTED- |
| TNS-ASP | ASPECT [IMPF -, PERF +, PROG -]
| MOOD | indicative
| 16 CLAUSE-TYPE [decl, LEX-SEM unerg, PASSIVE -, VFORM perf, VTYPE main]

- 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 learned in yesterday’s tutorial)
Morphology

- implemented using sophisticated and very fast, efficient finite-state machines (as you learned in yesterday’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)
Morphology

- implemented using sophisticated and very fast, efficient finite-state machines (as you learned in yesterday’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
Morphology

- implemented using sophisticated and very fast, efficient finite-state machines (as you learned in yesterday’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

CS 1:

```
NOUN-S_BASE N-T_BASE GEND_BASE NUM_BASE
laRk | +Noun | +Fem | +Sg
```

"laRkI"

```
PRED 'laRk'
NTYPE [NSEM [COMMON count]]
1[GEND fem, NUM sg, PERS 3]
```
Sample Analyses

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

- Reduplication
- Urdu Ezafe
- Complex Predicates and Causatives (demo)
The morphological analyzer is powerful enough to deal with phenomena such as *reduplication* (Beesley and Karttunen 1993, Bögel et al. 2007).
Morphology — Reduplication

- The morphological analyzer is powerful enough to deal with phenomena such as reduplication (Beesley and Karttunen 1993, Bögel et al. 2007).
- Urdu contains two very productive patterns for reduplication:
Morphology — Reduplication

- The morphological analyzer is powerful enough to deal with phenomena such as reduplication (Beesley and Karttunen 1993, Bögel et al. 2007).
- Urdu contains two very productive patterns for reduplication:
  - echo reduplication: onset of word is permuted, e.g. dEkHnA vEkHnA ‘seeing and such things’
The morphological analyzer is powerful enough to deal with phenomena such as *reduplication* (Beesley and Karttunen 1993, Bögel et al. 2007).

Urdu contains two very productive patterns for reduplication:

- *echo reduplication*: onset of word is permutated, e.g. dEkHnA vEkHnA ‘seeing and such things’
- *full word reduplication*: whole word is simply repeated, e.g. tanhA tanhA ‘very lonely (lonely lonely)’
The morphological analyzer is powerful enough to deal with phenomena such as \textit{reduplication} (Beesley and Karttunen 1993, Bögel et al. 2007).

Urdu contains two very productive patterns for reduplication:

- echo reduplication: onset of word is permutated, e.g. dEkHnA vEkHnA ‘seeing and such things’
- full word reduplication: whole word is simply repeated, e.g. tanhA tanhA ‘very lonely (lonely lonely)’

we do not need to deal with reduplication in the syntax — the finite-state technology can handle the whole process
Morphology — Reduplication

Our current analysis: follows Beesley & Karttunen and uses regular expressions as part of the tokenizer to pick out those sequences in which material is

1. completely doubled;
2. doubled, but the first consonant of the second word is changed.
Morphology — Reduplication

- The doubled sequence is treated as a complex word in terms of morphological analysis.

- The morphological analyzer treats it as a unit and tags it as a reduplicated item.

  kHAnA vAnA ⇔ kHAnA+Noun+Masc+Sg+Redup
  'food and those kinds of things'

  tHanDA tHanDA ⇔ tHanDA+Adj+Masc+Sg+Redup
  'ice cold (cold cold)'

- Since the morphological analyzer works for parsing and generation, we can generate reduplicated word forms (no listing required).
Morphology-Prosody Interface — Urdu Ezafe

- **Urdu** contains an interesting construction that is not part of **Hindi**.
Urdu contains an interesting construction that is not part of Hindi.

The *Ezafe Construction* is a loan from Persian.

a. sher=e panjAb
   lion=Ez Punjab
   ‘A/The lion of Punjab’

b. sadA=e buland
   voice=Ez high
   ‘high voice’
Urdu contains an interesting construction that is not part of Hindi.

The *Ezafe Construction* is a loan from Persian.

a. sher=e panjAb  
   lion=Ez Punjab  
   ‘A/The lion of Punjab’

b. sadA=e buland
   voice=Ez high
   ‘high voice’

Much has been written about Persian Ezafe.
Morphology-Prosody Interface — Urdu Ezafe

- Urdu contains an interesting construction that is not part of Hindi.

- The *Ezafe Construction* is a loan from Persian.

  a. sher=e panjAb
     lion=Ez Punjab
     ‘A/The lion of Punjab’

  b. sadA=e buland
     voice=Ez high
     ‘high voice’

- Much has been written about Persian Ezafe.

- Urdu Ezafe is easier to model than Persian Ezafe, since Urdu Ezafe can not have recursive complements.
Challenges for analysis of Urdu Ezafe

- Ezafe is an exception to the usual head-final pattern of Urdu.

  panjAb=kA sher
  Punjab=Gen.M.Sg lion.M.Sg
  ‘Punjab’s lion’
Challenges for analysis of Urdu Ezafe

- Ezafe is an exception to the usual head-final pattern of Urdu.
  
  \[ panjAb=\text{Gen.M.Sg} \text{ lion.M.Sg} \]
  ‘Punjab’s lion’

- In Ezafe, the modifier/complement (adjective or noun) is to the right of the head noun, not to the left, as with the genitive.
Challenges for analysis of Urdu Ezafe

- Ezafe is an exception to the usual head-final pattern of Urdu.

  \[ \text{panjAb} = kA \quad \text{sher} \]

  \[ \text{Punjab} = \text{Gen.M.Sg lion.M.Sg} \]

  ‘Punjab’s lion’

- In Ezafe, the modifier/complement (adjective or noun) is to the right of the head noun, not to the left, as with the genitive.

- Questions:
  1. what is status of the Ezafe e?
  2. how to model the construction in the morphology and the syntax?
We analyze the Ezafe $e$ as a **clitic**, not as a piece of inflectional morphology.
We analyze the Ezafe e as a clitic, not as a piece of inflectional morphology.

Evidence: Ezafe attaches to the right edge of a constituent, which is not possible for inflectional morphology.

[mAl O daulat]=e dunyA
material and wealth=Ez world
‘the material and wealth of the world’
We analyze the Ezafe as a clitic, not as a piece of inflectional morphology.

**Evidence:** Ezafe attaches to the right edge of a constituent, which is not possible for inflectional morphology.

\[[\text{mAl O} \text{ daulat}] = \text{e dunyA}\]

material and wealth = Ez world

‘the material and wealth of the world’

The clitic -e licenses the noun/adjective to its right as a modifier of the head noun to its left — which should be represented within the c-structure and the f-structure.
Syntactic Analysis — Urdu Ezafe

CS 1: NP "sher e panjAb"

NPez

N Pez

EzP

N EZ N

sher e panjAb

PRED 'sher'

MOD

{ PRED 'panjAb'

NTYPE NSEM [COMMON count]

NSYN common

30 GEND masc, MOD-TYPE ezafe, NUM sg, PERS 3

CHECK [EZAFA +]

NTYPE NSEM [COMMON count]

NSYN common

1 GEND masc, NUM sg, PERS 3
modifier ('Punjab') is licenced by Ezafe
Syntactic Analysis — Urdu Ezafes

- modifier (‘Punjab’) is licenced by Ezafes
- Ezafes are heads of Ezafe phrase constituents
Syntactic Analysis — Urdu Ezafe

- modifier (‘Punjab’) is licenced by Ezafe
- Ezafe is head of Ezafe phrase constituent
- complement of Ezafe modifies head noun
Syntactic Analysis — Urdu Ezafe

- modifier (‘Punjab’) is licenced by Ezafe
- Ezafe is head of Ezafe phrase constituent
- complement of Ezafe modifies head noun
- **However:** prosodically, the Ezafe -e is part of the word to its left, which is its host.
This is a fact about its \textit{prosodic} properties (part of \textit{prosodic phonology}).
This is a fact about its *prosodic* properties (part of *prosodic phonology*).

Our solution: use LFG’s modular architecture to model this prosodic property at the appropriate level of representation.
This is a fact about its *prosodic* properties (part of *prosodic phonology*).

Our solution: use LFG’s modular architecture to model this prosodic property at the appropriate level of representation.

We need an additional level: p-structure
This is a fact about its *prosodic* properties (part of *prosodic phonology*).

Our solution: use LFG’s modular architecture to model this prosodic property at the appropriate level of representation.

We need an additional level: p-structure

Ezafe is coded as part of the head noun within p-structure:

```
130[127
{1[CL-FORM ezafe, DOMAIN P-WORD, P-FORM shen]}
83
{2[DOMAIN P-PHRASE}
125
{1[DOMAIN P-WORD, P-FORM panjab]}
```
Modular Architecture

- The LFG architecture is modular and allows for differing levels of analysis.
- This seems to allow for just the right kind of flexibility for a linguistically motivated yet computationally efficient treatment of complex phenomena.
- Further example: complex predicates (demo).
- Not touched upon yet, but already part of the system: semantic analysis (demo).
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 output semantic form
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
Conclusion

- Urdu ParGram project devoted to developing a large-scale, broad-coverage LFG parsing and generation grammar using XLE
Conclusion

- **Urdu ParGram project** devoted to developing a large-scale, broad-coverage LFG parsing and generation grammar using XLE
  - pipeline architecture: single components may be used in other contexts
Urdu ParGram project devoted to developing a large-scale, broad-coverage LFG parsing and generation grammar using XLE

- pipeline architecture: single components may be used in other contexts
- informed by well-studied linguistic insights from LFG theory
Conclusion

- Urdu ParGram project devoted to developing a large-scale, broad-coverage LFG parsing and generation grammar using XLE
  - pipeline architecture: single components may be used in other contexts
  - informed by well-studied linguistic insights from LFG theory
- successfully implemented different aspects of Urdu grammar: reduplication, complex predication, Ezafe
Conclusion

- Urdu ParGram project devoted to developing a large-scale, broad-coverage LFG parsing and generation grammar using XLE
  - pipeline architecture: single components may be used in other contexts
  - informed by well-studied linguistic insights from LFG theory
- successfully implemented different aspects of Urdu grammar: reduplication, complex predication, Ezafe
- currently experimenting with additional annotation using p-structure (prosody) and XFR rewriting (semantics)
Urdu ParGram project devoted to developing a large-scale, broad-coverage LFG parsing and generation grammar using XLE

- pipeline architecture: single components may be used in other contexts
- informed by well-studied linguistic insights from LFG theory

- successfully implemented different aspects of Urdu grammar: reduplication, complex predication, Ezafe

- currently experimenting with additional annotation using p-structure (prosody) and XFR rewriting (semantics)

- LFG/XLE methodology: powerful, effective, proven and tested