Theoretical Linguistics and Grammar Engineering as Mutually Constraining Disciplines

Tracy Holloway King
2016-07-26
Grammar Engineering Programs

• LFG ParGram
• HPSG Grammar Matrix and DELPH-IN
• Goals:
  – Test underlying tenets of the theory: universality
  – Build resources to be used in applications
ParGram and ParSem

• Broad coverage grammars for diverse languages  
  – Began with English, German, and French  
• XLE as development platform  
• Test universality and parallel nature of functional-structures

• Issues around linguistic analysis  
  – adjectives, copulas, negation, auxiliaries, coordination  
• Issues around grammar engineering  
  – morphology-syntax interface  
  – ambiguity
Grammar Matrix

- Broad coverage, precision grammars for diverse languages
- Leverage expertise from existing grammars to build new ones
- Develop grammars with a common MRS format
- Exchange data and analyses of phenomena
- Tools for field linguists to build grammars
  - test hypotheses
  - encode results
Grammar Engineering to Confirm Hypotheses

• Low effort grammar engineering
  – Platform issues may limit usefulness

• Some examples:
  – LFG Linking Theory (Butt, Dalrymple, Frank 1997)
  – LFG resumptive pronouns (Asudeh 2004)
  – HPSG resumption & extraction (Crysmann 2015) & tone (Crysmann 2009)
  – LFG adjective coordination (Beyaev, Dalrymple, Lowe 2015)
Indeterminacy by Underspecification: Dalrymple, King, Sadler 2009

• How to formally encode indeterminacy
  – Indeterminate forms can simultaneously satisfy conflicting requirements

• Proposal: value of an indeterminate feature (case) is a complex, possibly underspecified feature structure
  – allows incremental, monotonic refinement
  – uses only atomic, boolean-valued features
  – covers indeterminate arguments and predicates
Underspecified, indeterminate case: German

Papageien
parrots
NOM/ACC/DAT/GEN
‘parrots’ (nominative, accusative, dative, or genitive)
Indeterminate nouns with case-assigning predicates

a. Er findet Papageien.
   he finds parrots
   OBJ=ACC NOM/ACC/DAT/GEN
   ‘He finds parrots.’

b. Er hilft Papageien.
   he helps parrots
   OBJ=DAT NOM/ACC/DAT/GEN
   ‘He helps parrots.’

c. Er findet und hilft Papageien.
   he finds and helps parrots
   OBJ=ACC OBJ=DAT NOM/ACC/DAT/GEN
   ‘He finds and helps parrots’
Proposed structure for case on German nouns

Determinate accusative case:

\[
\begin{bmatrix}
\text{NOM} & - \\
\text{ACC} & + \\
\text{GEN} & - \\
\text{DAT} & - \\
\end{bmatrix}
\]

Determinate dative case:

\[
\begin{bmatrix}
\text{NOM} & - \\
\text{ACC} & - \\
\text{GEN} & - \\
\text{DAT} & + \\
\end{bmatrix}
\]

Indeterminate case:

\[
\begin{bmatrix}
\text{NOM} \\
\text{ACC} \\
\text{GEN} \\
\text{DAT} \\
\end{bmatrix}
\]
Indeterminate noun with dative verb

a. Er hilft Papageien.
   he helps parrots
   OBJ=DAT NOM/ACC/GEN/DAT
   ‘He helps parrots.’

b. \[
\begin{array}{c}
\text{OBJ} \\
\text{CASE}
\end{array}
\begin{array}{c}
\text{PRED} \\
\text{‘PARROTS’}
\end{array}
\begin{array}{c}
\text{NOM} \\
\text{ACC} \\
\text{GEN} \\
\text{DAT +}
\end{array}
\]
Coordinating indeterminate noun with acc and dat verb

a. Er findet und hilft Papageien.
   he finds and helps parrots
   OBJ=ACC OBJ=DAT NOM/ACC/DAT/GEN
   ‘He finds and helps parrots’

b. \[
\text{PRED} \begin{array}{c}
\text{‘FIND’} \\
\text{PRED} \begin{array}{c}
\text{‘PARROTS’} \\
\text{OBJ} \begin{array}{c}
\text{CASE} \\
\text{OBJ} \begin{array}{c}
\text{PRED} \begin{array}{c}
\text{‘HELP’} \\
\text{OBJ} \begin{array}{c}
[] \\
\end{array} \end{array} \\
\end{array} \end{array} \end{array} \end{array} \\
\end{array}
\]
Determinate adjectives with indeterminate nouns

a. *Er findet und hilft alten Papageien.
   he finds and helps old parrots
   OBJ=ACC OBJ=DAT DAT NOM/ACC DAT/GEN

   ‘He finds and helps old parrots.’

b. Ill-formed f-structure:

   \[
   \begin{array}{c}
   \text{PRED} \quad \text{‘PARROTS’} \\
   | \quad | \\
   \text{NOM} \quad - \\
   \text{ACC} \quad +/– \\
   \text{GEN} \quad - \\
   \text{DAT} \quad + \\
   \end{array}
   \]
Indeterminate adjectives with indeterminate noun

**rosa: [no case restrictions]**

a. Er findet und hilft rosa Papageien.
   he finds and helps pink parrots
   OBJ=ACC OBJ=DAT NOM/ACC DAT/GEN

   ‘He finds and helps pink parrots.’

b. \[
\begin{array}{c}
\text{PRED} \quad \text{‘PARROTS’} \\
\text{NOM} \\
\text{ACC} \\
\text{GEN} \\
\text{DAT} \\
\end{array}
\]
Indeterminacy Grammar Engineering Implementation

• One exemplar of each type
• Basic constituent- and functional-structure rules
• Tested whether analysis would work
  – all grammatical sentences accepted
  – all ungrammatical sentences rejected
Linguistic Issues Highlighted by Grammar Engineering

• Discover interesting facts from:
  – implementing each detail
  – working on corpora
    • interaction of phenomena
    • complexity of phenomena

• Caveat: things not working could be due to implementation limitations
Copulas, Adjectives and Subjects: Dalrymple, Dyvik, King 2004

• Second argument of copulars: subject or no?

• Predicate adjective agreement: simple with subjects

Elle est petite.
she.F.Sg is small.F.Sg
‘She is small.’
Subject analysis: adjective and copula

\[
\begin{align*}
\text{SUBJ} & \quad [\text{NUM sg} \quad 1] \\
\text{GEND} & \quad \text{fem} \\
\text{XCOMP} & \quad [\text{PRED} \quad \text{‘small<SUBJ>’}] \\
\text{SUBJ} & \quad [\text{SUBJ} \quad \text{1}] \\
\text{PRED} & \quad \text{‘be<XCOMP>SUBJ’} \\
\end{align*}
\]

petite \quad (↑ \text{PRED}) = \text{‘small<SUBJ>’}
(↑ \text{SUBJ NUM}) =c \quad \text{sg}
(↑ \text{SUBJ GEND}) =c \quad \text{fem}

est \quad (↑ \text{PRED}) = \text{‘be<XCOMP>SUBJ’}
(↑ \text{SUBJ}) = (↑ \text{XCOMP SUBJ})
Non-subject Analysis: adjective and copula

Non-subject analysis with closed copular complement:

\[
\begin{align*}
\text{PRED} & \quad \text{‘be<SUBJ,PREDLINK>’} \\
\text{SUBJ} & \quad \begin{align*}
\text{NUM} & \quad \text{sg} \\
\text{GEND} & \quad \text{fem}
\end{align*} \\
\text{PREDLINK} & \quad \text{PRED} \quad \text{‘small’}
\end{align*}
\]

\[
\text{petite} \quad (\uparrow \text{PRED}) = \text{‘small’} \\
(\text{PREDLINK} \uparrow) \text{SUBJ NUM} = \text{c sg} \\
(\text{PREDLINK} \uparrow) \text{SUBJ GEND} = \text{c fem}
\]
Pro: Raising adjectives

• It is likely/bound/certain to rain.
• They are eager/foolish/loathe to leave.

• Open complement with subject
Con: If the post-copular has a subject

- The problem is that they appear / their appearing / for them to appear.
What did we learn?

• Open complement good for:
  – Agreeing predicate adjectives
  – Raising adjectives

• Open complement bad for:
  – Predicate nouns/adjectives with a subject

• No elegant overall solution
Implementation Guiding Analyses

• Complex Predicates via Restriction:
  – Butt, King, Maxwell 2009
• Complex predicates heavily studied in LFG
• ParGram implementation: restriction

\[
\begin{align*}
n\text{Adiyah} & \quad nE \quad k\text{hAnI} \quad y\text{Ad} \quad k-I \\
N\text{adya.F.Sg} \quad \text{Erg story.F.Sg.Nom} \quad \text{memory.F.Sg.Nom} \quad \text{do-Perf.F.Sg} \\
& \quad \text{‘Nadya remembered a/the story.’}
\end{align*}
\]

a. \((↑ \text{PRED}) = ‘y\text{Ad/memory}<\text{OBJ}’ | (↑ \text{PRED}) = ‘y\text{Ad/memory}’\)

b. \((↑ \text{PRED}) = ‘\text{kar/do}< \text{SUBJ OBJ}’\)

c. Standard LFG PRED: \((↑ \text{PRED}) = ‘y\text{Ad/memory-kar}<\text{SUBJ,OBJ}’\)
Restriction Operator (Kaplan & Wedekind 1993)

- Restrict out features of f-structure
- New f-structure is identical except for restricted features
- Allows dynamic composition of predicates

```
Original f-structure  F-structure with case restricted out

[PRED  'Nadya'
NUM  sg
PERS  3
CASE  ERG ]

[PRED  'Nadya'
NUM  sg
PERS  3 ]
```
Lexical entries and f-structure for example

a. Standard LFG PRED: \((\uparrow \text{PRED}) = \text{‘memory-do<SUBJ,OBJ>’}\)

b. Proposed restricted PRED: \((\uparrow \text{PRED}) = \text{‘do<SUBJ,’memory<OBJ>’}’\)

\[
\begin{array}{c}
\text{PRED} & \text{'do<SUBJ,’memory<OBJ>’}’} \\
\text{SUBJ} & \text{[ PRED ’Nadya’ ]} \\
\text{OBJ} & \text{[ PRED ’story’ ]}
\end{array}
\]
How it works

• F-structure is formed by dynamic composition of the subcategorization frames from *do* and *memory*

• Restriction operator invoked on annotated c-structure rules

\[
\begin{align*}
Vcp \rightarrow & N \\
\downarrow \text{PRED} = \uparrow \text{PRED} \\
(\uparrow \text{PRED ARG2}) = (\downarrow \text{PRED})
\end{align*}
\]
Lexical entries

- *do* contributes subject and needs a further predicate
  
  \[(\uparrow \text{PRED}) = 'do<\text{SUBJ} \%\text{Pred} >'\]

- *memory* contributes an object
  
  \[(\uparrow \text{PRED}) = 'memory<\text{OBJ}>'\]

- combined as complex predicate
  
  \[(\uparrow \text{PRED}) = 'do<\text{SUBJ}',\text{memory}<\text{OBJ}>'\]
Complex Predicates via Restriction

• Creates the predicates and functional-structures we want
• Accounts for stacked complex predicates
• Allows a ParGram Urdu analysis compatible with output of other grammars

• Theory:
  – Should this be used theoretically?

• Grammar Engineering:
  – Issues with interaction with lexical rules
  – Hassle to maintain
Insights into Architecture Issues

- Passive-Causative Interactions
- Urdu and Turkish ParGram grammars
- Passive done by lexical rule
- Causative done by restriction
- Interaction gave exactly the wrong results
Urdu Causatives

• Morphological
• Causer subject
• Underlying subject of verb becomes oblique

a. yassIn=nE kHAnA kHa-yA
   Yassin=Erg food.M.Sg.Nom eat-Perf.M.Sg
   ‘Yassin ate food.’

b. nAdyA=nE yassIn=kO kHAnA kHil-A-yA
   Nadya=Erg Yassin=Dat food.M.Sg.Nom eat-Caus-Perf.M.Sg
   ‘Nadya had Yassin eat (fed Yassin).’
Causative Analysis

eat.Causative
kHilvAyA ⇔ kHA +Verb +Cause +Perf +Masc +Sg

\[ V \rightarrow V_{\text{BASE}} \quad \text{CAUSE} \]
\[ \downarrow \text{PRED}_{\text{SUBJ}} = \uparrow \text{PRED}_{\text{SUBJ}} \quad \uparrow = \downarrow \]
\[ (\downarrow \text{SUBJ}) = (\uparrow \text{OBJ-GO}) \]
\[ (\uparrow \text{PRED ARG2}) = (\downarrow \text{PRED}) \]

1. +Cause: \((\uparrow \text{PRED}) = \text{’cause< SUBJ %Pred ’}\)

2. eat \((\uparrow \text{PRED}) = \text{’eat< SUBJ OBJ ’}\)
Passives

• Formed by combining the verb jA ‘go’ with the perfect form of the main verb
• Object of the base verb is realized as a subject
• Subject/agent of the base verb is realized as an adjunct

a. yassIn=nE  kHAnA  kHa-yA
   YassIn= Erg  food.M.Sg.Nom  eat-Perf.M.Sg
   ‘Yassin ate food.’

b. kHAnA  yassIn=sE  kHa-yA  ga-yA
   food.M.Sg.Nom  YassIn=Inst  eat-Perf.M.Sg  go-Perf.M.Sg
   ‘The food was eaten by Yassin.’
Passive with Causative

- Subject of the causative becomes an adjunct (sE marked)

  a. nAdyA=\textit{nE} yassIn=kO kHanA kHil-A-yA
     \hspace{1cm} Nadya=Erg Yassin=Dat food.M.Sg.Nom \textbf{eat-Caus-Perf.M.Sg}
     \hspace{1cm} ‘Nadya had Yassin eat (fed Yassin).’

  b. yassIn=kO nAdyA=\textit{sE} kHanA kHil-A-yA ga-yA
     \hspace{1cm} Yassin=Dat Nadya=Inst food.M.Sg.Nom \textbf{eat-Caus-Perf.M.Sg go-Perf.M.Sg}
     \hspace{1cm} ‘The food was fed to Yassin by/through Nadya.’

- Subject of original main verb cannot be passivized

  *nAdyA=\textit{nE} yassIn=\textit{sE} kHanA kHil-A-yA ga-yA
     \hspace{1cm} Nadya=Erg Yassin=Inst food.M.Sg.Nom \textbf{eat-Caus-Perf.M.Sg go-Perf.M.Sg}
     \hspace{1cm} ‘Nadya made the food be eaten by/through Yassin.’

- Grammar had reverse pattern
Architecture Issue with Implementation

• Passive via lexical rules when compiling lexicon

   a. \( \text{TRANS}(P) = \oplus(\text{PASSIVE}(\uparrow\text{PRE}) = 'P<(\uparrow\text{SUBJ}) \ (\uparrow\text{OBJ})>') \)

   b. \( \text{PASSIVE}(P) = (\uparrow\text{SUBJ}) \longrightarrow \text{NULL} \)
      \( (\uparrow\text{OBJ}) \longrightarrow (\uparrow\text{SUBJ}) \)

• Causative via restriction applies afterwards

  Ungrammatical derivation of passive+causative:

  a. Original Predicate: \( (\uparrow\text{PRE}) = '\text{eat}<\text{(\uparrow\text{SUBJ}) \text{(\uparrow\text{OBJ})}>' \)

  b. Lexical Rule Passive: \( (\uparrow\text{PRE}) = '\text{eat}<\text{NULL} \text{(\uparrow\text{SUBJ})}>' \)

  c. Restriction Causative:
     \( (\uparrow\text{PRE}) = '\text{cause} <\text{SUBJ,'eat}<\text{NULL, OBL-GO}>' \)
Architecture continued

• Have passive also apply via restriction

  Grammatical derivation of passive+causative:
  
a. Original Predicate(↑PRED)=‘eat<(↑SUBJ) (↑OBJ)>’
  b. Restriction Causative: (↑PRED) = ‘cause<SUBJ,’eat<OBL-GO,OBJ>’>
  c. Restriction Passive: (↑PRED) = ‘cause<NULL,’eat<OBL-GO,SUBJ>’>

• Works but not satisfactory
• Argument alternations in argument-structure
  – but not implemented
• Traditional lexical rules: wrong order of application when causative is morphological but passive syntactic