Grammar Engineering
March 29, 2004
Introduction, overview,
HPSG basics
Overview

- The BIG Picture
- The LinGO Grammar Matrix
- Course requirements/workflow
- Pick a language, any language
- HPSG basics
- Other approaches
The BIG Picture: Precision Grammars

- relate surface strings to semantic representations
- distinguish grammatical from ungrammatical sentences
- knowledge engineering approach to parsing
- can be used for both parsing and generation
The BIG Picture: Applications

- language documentation/linguistic hypothesis testing
- machine translation
- automated email response
- augmentative and assistive communication
- computer assisted language learning
- IR (from structured or unstructured data)
- . . .
The BIG Picture: Hybrid approaches (1/2)

- Naturally occurring language is noisy
  - Typos
  - “mark-up”
  - Addresses & other non-linguistic strings
  - False starts
  - Hesitations
  - …

- Allowing for the noise within the grammar would reduce its precision

- And then there’s ambiguity, unknown words, …
The BIG Picture: Hybrid approaches (2/2)

- Combine symbolic (aka deep) and stochastic (aka shallow) approaches:
  - Statistical parse selection
  - (Statistical) named entity recognition and POS tagging in a preprocessing step (for unknown word handling)
  - Tiered systems with a shallow parser as a fall back for the precision parser

- Coming the other direction, deep grammars can provide richer linguistic resources for training statistical systems (e.g., MT systems).
The LinGO Grammar Matrix (1/3)

- One of the primary impediments to deploying precision grammars is that they are expensive to build.
- The Grammar Matrix aims to address this by providing a starter-kit which allows for quick initial development while supporting long-term expansion.
- The Grammar Matrix also represents a set of hypotheses about cross-linguistic universals.
The LinGO Grammar Matrix (2/3)

- A sampling of hypotheses:
  - Words and phrases combine to make larger phrases.
  - The semantics of a phrase is determined by the words in the phrase and how they are put together (Frege).
  - Some rules for phrases add semantics, and some don’t.
  - Most phrases have an identifiable head daughter.
More hypotheses:

- Heads determine which types of arguments they require, and how they combine semantically with those arguments.
- Modifiers determine which kinds of heads they modify, and how they combine semantically with those heads.
- No lexical or syntactic rule can remove semantic information.
Over 9 weekly lab exercises, each student will build a Matrix-based grammar of a different language.

On Mondays, I’ll announce what you need to have prepared in order to do the Wednesday lab.

Class time on Wednesdays will be lab time, to start each exercise.

Labs are due (submitted via E-Submit) notionally on Fridays, effectively by midnight Sunday night.
Make use of EPost!

There are no required readings, but if you do not have a strong background in syntax, I strongly recommend Sag et al 2003.

Copestake 2002 provides an extensive introduction to the LKB.

http://courses.washington.edu/ling471
Pick a language, any language (1/2)

- Each student must pick a different language.
- No English.
- Undergrads have priority for languages they already know.
Pick a language, any language (2/2)

- Languages with non-Latin alphabets will need to be done in translation (sorry)

- Languages with complex morphophonology might require some fudging (sorry again)

- If you aren’t working on a language you already know, pick a language with a good descriptive or teaching grammar available.
HPSG Basics

- Context-free(-like) grammar
- Feature structures
- Multiple inheritance type hierarchy
- Unification
- Rich lexical entries
- Constructions
$CF(-\text{like})G$

- $S \rightarrow \text{NP VP}$
- $S$
  \[ \begin{array}{c}
    \text{NP} \\
    \text{VP} \\
  \end{array} \]
- Problems:
  - Quickly get too many rules (try dealing with case, subcategorization, and agreement...)
  - Unconstrained: why not write rules like $D \rightarrow \text{NP S}$?
  - Loss of generality: what do intrans-sg-V and ditrans-pl-V have in common?
Solution: Add features

- Same idea of rewrite rules, but the labels on the nodes are now bundles of information, expressed as feature value pairs.

- Underspecification: Only specify those features that you care about. (e.g., the VP rule doesn’t care about the number value of NP objects).

- Capture generalizations: all verbs are [HEAD verb], regardless of their agreement properties, transitivity, etc.

- Allow values to be feature structures (and lists of feature structures) and the rules become quite simple.
Multiple inheritance type hierarchy

A type hierarchy ...

- ... states what kinds of objects we claim exist (the types).
- ... organizes the objects hierarchically into classes with shared properties (the IST relations).
- ... states what general properties each kind of object has (the feature and feature value declarations).
Technical note: Types v. instances

• The LKB distinguishes between types and instances.

• Instances are the maximally specific items in the hierarchy which the parser/generator can use in processing sentences.

• Types are used in the definition of instances.

• Types can have multiple parents.

• Instances can only have one parent.
Unification

- Phrase structure rules provide some information about the phrases they build.
- The words (or phrases) that combine as the daughters of those phrase structure rules provide more.
- How to combine that information? Unification, which we’ll come back to below.
A Pizza Type Hierarchy

pizza-thing

pizza

CRUST,
Toppings

topping-set

OLIVES,
ONIONS,
MUSHROOMS

vegetarian

non-vegetarian

SAUSAGE,
PEPPERONI,
HAM
<table>
<thead>
<tr>
<th>TYPE</th>
<th>FEATURES/VALUES</th>
<th>IST</th>
</tr>
</thead>
<tbody>
<tr>
<td>pizza-thing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pizza</td>
<td>[CRUST { thick, thin, stuffed }]</td>
<td>pizza-thing</td>
</tr>
<tr>
<td></td>
<td>[TOPPINGS \textit{topping-set}]</td>
<td></td>
</tr>
<tr>
<td>topping-set</td>
<td>\textit{topping-set}</td>
<td>pizza-thing</td>
</tr>
<tr>
<td></td>
<td>[OLIVES { +, - }]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[ONIONS { +, - }]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[MUSHROOMS { +, - }]</td>
<td></td>
</tr>
<tr>
<td>vegetarian</td>
<td></td>
<td>topping-set</td>
</tr>
<tr>
<td>non-vegetarian</td>
<td>[SAUSAGE { +, - }]</td>
<td>topping-set</td>
</tr>
<tr>
<td></td>
<td>[PEPPERONI { +, - }]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[BBQ CHICKEN { +, - }]</td>
<td></td>
</tr>
</tbody>
</table>
pizza-thing := *top*.
pizza := pizza-thing &
    [ CRUST crust,
      TOPPINGS topping-set ].
crust := *top*.
thick := crust.
thin := crust.
stuffed := crust.
topping-set := pizza-thing &
    [ OLIVES bool,
      ONIONS bool,
      MUSHROOMS bool ]
...

Unification

\[
\begin{align*}
\text{pizza} & \quad \text{CRUST} \quad \text{thick} \\
\text{TOPPINGS} & \quad \begin{bmatrix} 
\text{OLIVES} & + \\
\text{HAM} & - 
\end{bmatrix} \\
\end{align*}
\]

\[
\begin{align*}
\text{pizza} & \quad \text{TOPPINGS} \\
& \quad \begin{bmatrix} 
\text{OLIVES} & + \\
\text{ONIONS} & + 
\end{bmatrix} \\
\end{align*}
\]
Unification

\[
\begin{align*}
&\text{pizza} \\
&\text{CRUST} \quad \text{thick} \\
&\text{Toppings} \\
&\quad \text{Olives} + \\
&\quad \text{Onions} + \\
&\quad \text{Ham} -
\end{align*}
\]
Unification

\[
\begin{pmatrix}
\text{pizza} \\
\text{CRUST} \text{ thick} \\
\text{TOPPINGS} \begin{cases} 
\text{OLIVES} & + \\
\text{HAM} & - 
\end{cases}
\end{pmatrix} \sqsubseteq 
\begin{pmatrix}
\text{pizza} \\
\text{CRUST} \text{ thin} \\
\text{TOPPINGS} \begin{cases} 
\text{OLIVES} & + \\
\text{ONIONS} & + 
\end{cases}
\end{pmatrix}
\]
Unification
Unification

```
pizza
CRUST thick
TOPPINGS [OLIVES + HAM + ]
```

```
pizza
CRUST thick
TOPPINGS vegetarian
```
Unification
Unification

\[
\begin{array}{c}
pizza \\
\text{CRUST} \\
\text{TOPPINGS}
\end{array}
\] \quad \begin{array}{c}
thick \\
\text{OLIVES} \\
\text{HAM}
\end{array}
\quad \begin{array}{c}
pizza \\
\text{CRUST} \\
\text{TOPPINGS}
\end{array}
\]
Unification
A Pizza Type Hierarchy

pizza-thing

pizza

- CRUST, TOPPINGS
  - vegetarian
  - non-vegetarian
    - SAUSAGE, PEPPERONI, HAM
  - topping-set
    - OLIVES, ONIONS, MUSHROOMS
A New Theory of Pizzas

\[
pizza : \begin{cases} 
\text{CRUST} & \{ \text{thick, thin, stuffed} \} \\
\text{ONE-HALF} & \text{topping-set} \\
\text{OTHER-HALF} & \text{topping-set}
\end{cases}
\]
Unification

\[
pizza\left[\text{ONE-HALF, ONIONS \scriptsize{+}, OLIVES \scriptsize{-}}\right] \sqsubseteq \left[pizza\left[\text{OTHER-HALF, ONIONS \scriptsize{-}, OLIVES \scriptsize{+}}\right]\right]
\]
Unification

\[
\begin{pmatrix}
pizza \\
\text{ONE-HALF} \\
\text{OTHER-HALF}
\end{pmatrix}
\begin{pmatrix}
onions & + \\
olives & - \\
onions & - \\
olives & +
\end{pmatrix}
\]
Identity Constraints (Tags)

pizza

CRUST

ONE-HALF

thin

OLIVES

ONIONS

OTHER-HALF

OLIVES

ONIONS

OLIVES

ONIONS
Unification
Unification

\[ \text{pizza} \]
\[ \begin{array}{c}
\text{ONE-HALF} \\
\text{OTHER-HALF}
\end{array} \]
\[ \begin{array}{c}
\text{ONIONS} & + \\
\text{OLIVES} & - \\
\text{MUSHROOMS} & -
\end{array} \]
Unification

\[
\begin{align*}
pizza
\text{ONE-HALF} & \quad \Box \\
\text{OTHER-HALF} & \quad \Box \\
\end{align*}
\begin{align*}
\text{ONIONS} & \quad + \\
\text{OLIVES} & \quad - \\
\text{MUSHROOMS} & \quad -
\end{align*}
\]
Unification

\[
pizza \\
\text{ONE-HALF} \quad \square \quad \text{ONIONS} \quad + \\
\text{OTHER-HALF} \quad \square \quad \text{OLIVES} \quad +
\]

\[
pizza \\
\text{ONE-HALF} \quad \square \quad \text{vegetarian}
\]

\[
pizza \\
\text{ONE-HALF} \quad \square \quad \text{SAUSAGE} \quad + \\
\text{HAM} \quad -
\]
Unification
Rich lexical entries (1/2)

- In HPSG/Matrix grammars, most of the information is encoded in the lexicon.
- The type hierarchy serves as a means of capturing generalizations across that information.
- Lexical item specify their orthography, part of speech, agreement information, valence requirements, semantic contribution, and argument linking.
Rich lexical entries (2/2)

- Most of that information is stated on various supertypes, so that an actual lexical entry (instance) specifies only its lexical type, orthography, and “key” relation.
- Lexical rules relate base lexical entries to other lexical entries (e.g., plural nouns, passive verbs...).
Constructions (1/2)

- A few very general phrase structure rules do most of the work.
  - head-specifier
  - head-complement
  - head-subject
  - head-filler
  - head-modifier
We also find that some mildly and some extremely quirky constructions require their own special rules.

- relative clauses (of various sorts)
- *just because ... doesn’t mean*
- noun noun compounds
- appositives
- ...

The ERG currently has 105 syntactic constructions.
HPSG Basics

- Context-free(-like) grammar
- Feature structures
- Multiple inheritance type hierarchy
- Unification
- Rich lexical entries
- Constructions
Other approaches

- The LinGO consortium specializes in large HPSG grammars.
- Other broad-coverage precision grammars have been built in/by/with:
  - LFG (ParGram: Butt et al 1999)
  - F/XTAG (Doran et al 1994)
  - ALE/Controll (Götz & Meurers 1997)
- Proprietary formalisms at Microsoft and Boeing.
Bring for next time

- Your choice of language
- A transitive verb
- An intransitive verb
- Two nouns
- Determiners or particles required in NPs (as appropriate)
- An understanding of the basics of case and agreement in your language
- Knowledge of how to use emacs.
Overview

- The BIG Picture
- The LinGO Grammar Matrix
- Course requirements/workflow
- Pick a language, any language
- HPSG basics
- Other approaches