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.

The LinGO Grammar Matrix (3/3)

- 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.

Course requirements/workflow (1/2)

- 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.

Course requirements/workflow (2/2)

- 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.

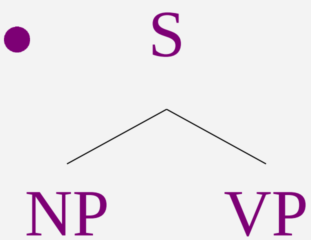
Pick a language, any language (2/2)

- 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(-like)G

- S $\rightarrow$ NP VP
- 

- Problems:
  - Quickly get too many rules (try dealing with case, subcategorization, and agreement...)
  - Unconstrained: why not write rules like D $\rightarrow$ 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).

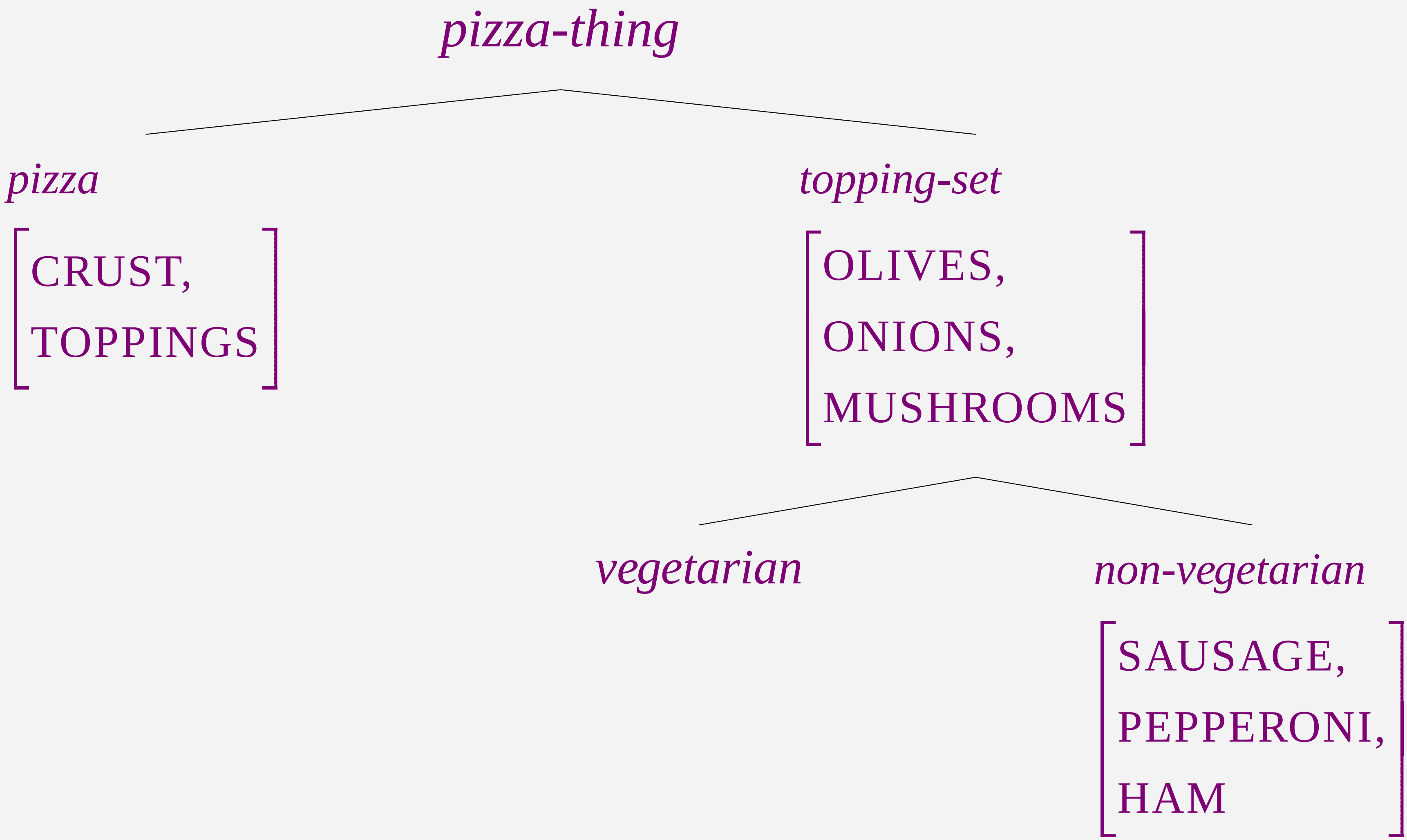
Technical note: Types v. instances

- 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



TYPE	FEATURES/VALUES	IST
pizza-thing		
pizza	$\begin{bmatrix} \text{CRUST} & \{\text{thick, thin, stuffed}\} \\ \text{TOPPINGS} & \textit{topping-set} \end{bmatrix}$	*pizza-thing*
topping-set	$\begin{bmatrix} \text{OLIVES} & \{+, -\} \\ \text{ONIONS} & \{+, -\} \\ \text{MUSHROOMS} & \{+, -\} \end{bmatrix}$	*pizza-thing*
vegetarian		*topping-set*
non-vegetarian	$\begin{bmatrix} \text{SAUSAGE} & \{+, -\} \\ \text{PEPPERONI} & \{+, -\} \\ \text{BBQ CHICKEN} & \{+, -\} \end{bmatrix}$	*topping-set*

```
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

$$\begin{bmatrix} \textit{pizza} & \\ \text{CRUST} & \textit{thick} \\ \text{TOPPINGS} & \begin{bmatrix} \text{OLIVES} & + \\ \text{HAM} & - \end{bmatrix} \end{bmatrix} \sqcup \begin{bmatrix} \textit{pizza} & \\ \text{TOPPINGS} & \begin{bmatrix} \text{OLIVES} & + \\ \text{ONIONS} & + \end{bmatrix} \end{bmatrix}$$

Unification

$$\begin{bmatrix} \textit{pizza} & \\ \text{CRUST} & \textit{thick} \\ \text{TOPPINGS} & \begin{bmatrix} \text{OLIVES} & + \\ \text{ONIONS} & + \\ \text{HAM} & - \end{bmatrix} \end{bmatrix}$$

Unification

$$\begin{bmatrix} \textit{pizza} & \\ \text{CRUST} & \textit{thick} \\ \text{TOPPINGS} & \begin{bmatrix} \text{OLIVES} & + \\ \text{HAM} & - \end{bmatrix} \end{bmatrix} \sqcup \begin{bmatrix} \textit{pizza} & \\ \text{CRUST} & \textit{thin} \\ \text{TOPPINGS} & \begin{bmatrix} \text{OLIVES} & + \\ \text{ONIONS} & + \end{bmatrix} \end{bmatrix}$$

Unification

ϕ

Unification

$$\begin{bmatrix} \textit{pizza} & \\ \text{CRUST} & \textit{thick} \\ \text{TOPPINGS} & \begin{bmatrix} \text{OLIVES} & + \\ \text{HAM} & + \end{bmatrix} \end{bmatrix} \sqcup \begin{bmatrix} \textit{pizza} & \\ \text{CRUST} & \textit{thick} \\ \text{TOPPINGS} & \textit{vegetarian} \end{bmatrix}$$

Unification

$$\phi$$

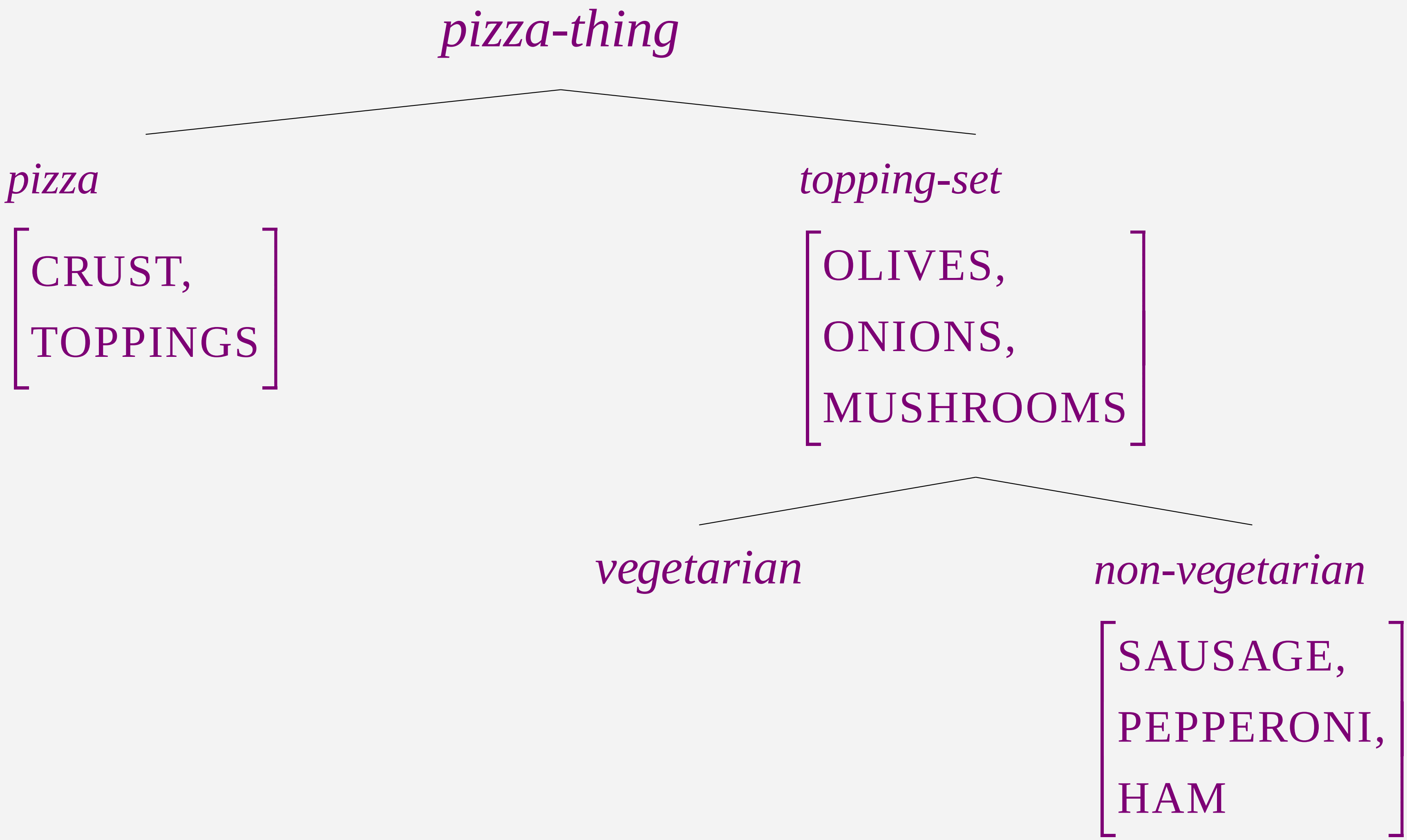
Unification

$$\begin{bmatrix} \textit{pizza} & \\ \text{CRUST} & \textit{thick} \\ \text{TOPPINGS} & \begin{bmatrix} \text{OLIVES} & + \\ \text{HAM} & - \end{bmatrix} \end{bmatrix} \sqcup \begin{bmatrix} \textit{pizza} & \\ \text{CRUST} & \textit{thick} \\ \text{TOPPINGS} & \textit{vegetarian} \end{bmatrix}$$

Unification

ϕ

A Pizza Type Hierarchy



A New Theory of Pizzas

$$\textit{pizza} : \begin{bmatrix} \text{CRUST} & \{\text{thick}, \text{thin}, \text{stuffed}\} \\ \text{ONE-HALF} & \textit{topping-set} \\ \text{OTHER-HALF} & \textit{topping-set} \end{bmatrix}$$

Unification

$$\begin{bmatrix} \textit{pizza} & \\ \text{ONE-HALF} & \begin{bmatrix} \text{ONIONS} & + \\ \text{OLIVES} & - \end{bmatrix} \end{bmatrix} \sqcup \begin{bmatrix} \textit{pizza} & \\ \text{OTHER-HALF} & \begin{bmatrix} \text{ONIONS} & - \\ \text{OLIVES} & + \end{bmatrix} \end{bmatrix}$$

Unification

$$
\begin{bmatrix}
\textit{pizza} & \\
\text{ONE-HALF} & \begin{bmatrix} \text{ONIONS} & + \\ \text{OLIVES} & - \end{bmatrix} \\
\text{OTHER-HALF} & \begin{bmatrix} \text{ONIONS} & - \\ \text{OLIVES} & + \end{bmatrix}
\end{bmatrix}
$$

Identity Constraints (Tags)

pizza		
CRUST	*thin*	
ONE-HALF	OLIVES	[1]
	ONIONS	[2]
OTHER-HALF	OLIVES	[1]
	ONIONS	[2]

Unification

$$\begin{bmatrix} \textit{pizza} & \\ \text{ONE-HALF} & \boxed{1}\begin{bmatrix}\text{ONIONS} & + \\ \text{OLIVES} & - \end{bmatrix} \\ \text{OTHER-HALF} & \boxed{1} \end{bmatrix} \sqcup \begin{bmatrix} \textit{pizza} & \\ \text{OTHER-HALF} & \begin{bmatrix}\text{MSHRMS} & - \\ \text{OLIVES} & - \end{bmatrix} \end{bmatrix}$$

Unification

$$\begin{bmatrix} \textit{pizza} & \\ \text{ONE-HALF} & \boxed{1}\begin{bmatrix}\text{ONIONS} & + \\ \text{OLIVES} & - \\ \text{MUSHROOMS} & -\end{bmatrix} \\ \text{OTHER-HALF} & \boxed{1}\end{bmatrix}$$

Unification

$$
\begin{bmatrix}
\textit{pizza} & & \\
\text{ONE-HALF} & \boxed{1} & \\
\text{OTHER-HALF} & \boxed{1} & \begin{bmatrix} \text{ONIONS} & + \\ \text{OLIVES} & - \\ \text{MUSHROOMS} & - \end{bmatrix}
\end{bmatrix}
$$

Unification

$$
\begin{bmatrix}
\textit{pizza} & & \\
\text{ONE-HALF} & \boxed{1} & \begin{bmatrix} \text{ONIONS} & + \\ \text{OLIVES} & + \end{bmatrix} \\
\text{OTHER-HALF} & \boxed{1} & \textit{vegetarian}
\end{bmatrix}
\sqcup
\begin{bmatrix}
\textit{pizza} & \\
\text{ONE-HALF} & \begin{bmatrix} \text{SAUSAGE} & + \\ \text{HAM} & - \end{bmatrix}
\end{bmatrix}
$$

Unification

$$\phi$$

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

Constructions (2/2)

- 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