Practical Grammar Engineering Using HPSG

Frederik Fouvry, Petter Haugereid,
Valia Kordoni, Melanie Siegel

Inhalt

- Organisatorisches
- HPSG Grammatiken
- Werkzeuge
- Kodierungen

- TDL Syntax

Über dieses Seminar

- Ziele:
  - Einführung in die Grammatikentwicklungsumgebung LKB
  - Einführung in die Nutzung der Grammar Matrix
  - Grundlagen der Formalisierung von Grammatiken in getypten Featurestrukturen
- Methoden:
  - Tägliche Kurzeinführungen und dann Übungen
  - Anpassen und Weiterentwickeln von einfachen HPSG Grammatiken, aufbauend auf der Matrix, in LKB
- Scheine:
  - Im Anschluss an den Kurs eine eigene Grammatik für eine beliebige Sprache entwickeln.
  - Basierend auf der Matrix.
  - Dokumentiert und mit Testbeispielen.
  - Bis September 2004.

Die Übungen im Kurs sind eine wichtige Voraussetzung dafür !!!

Organisatorisches

- TN-Liste
- Kurs-Termine:
  - 13.4., 14.4., 15.4., 16.4., 19.4.
  - jeweils 9:00-12:00, 14:00-16:00
- Anmeldung für Scheine an der Coli

Wofür braucht man Implementierungen von Grammatiken?

- **Forschung**: Formalisierung von linguistischen Theorien, komplexe Interaktionen von Sprachphänomenen, cross-linguale Generalisierungen
- **Didaktik**: Lehre von Theorien oder Analysen in formaler Morphologie, Syntax und Semantik. Experimente von StudentInnen.
- **Anwendungen**: Forschungsprototypen oder kommerzielle Anwendungen, z.B. Informationsextraktion, automatische email-Beantwortung, Grammar Checking,...

HPSG Grammatiken

- Lingo English Resource Grammar
- GerGram deutsche HPSG
- JACY japanische HPSG
- Griechische HPSG
- NorSource Norwegische HPSG
- Italian HPSG at Celi
- Weitere Projekte: Koreanisch, Spanisch, und andere

Einsatz von HPSG Grammatiken

- Maschinelle Übersetzung (Verbmobil LOGON)
- Automatische Email-Beantwortung (YY)
- Forschungsprojekte zu CALL
- Business Intelligence, Email-Kategorisierung, Kreativ-Tools (DeepThought)
- Question Answering (QETAL)

LKB

- Grammatikentwicklungsumgebung für HPSG Grammatiken
- Mit Entwicklungshilfen zum Debuggen wie z.B. Chart Display, Vergleich von Feature Strukuren, Baum-Display, Typenhierarchie-Display.
- Läuft mit Lisp.
- Anbindung an Evaluationstool [incr tsdb()] zum Testen größerer Datenmengen.

Copestake 2002, in der Bibliothek!

TDL

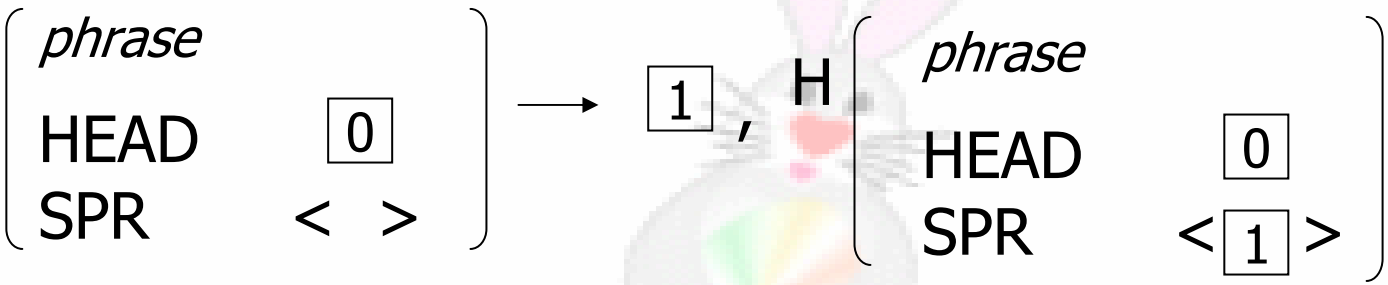
- Beschreibungssprache für Typdefinitionen und typbasierte Grammatiken.
- Krieger, Hans-Ulrich and Schäfer, Ulrich (1994): TDL - A Type Description Language for HPSG.

  Part 2: User Guide.

  ftp://lt-ftp.dfki.uni-sb.de/pub/papers/local/D-94-14.ps.Z

Grammatikregeln

- Zum Beispiel head-specifier-rule:




```
Head-specifier-rule := phrase &
  [HEAD #0,
    SPR < >,
    ARGS < phrase & #1 & [SPR < >],
           phrase & [HEAD #0,
                     SPR < #1 >] >].
```

Lexikoneinträge

- In HPSG Grammatiken steht die meiste Information im Lexikon.
- Die Typenhierarchie stellt Generalisierungen über diese Information auf.
- Information in Lexikoneinträgen ist Orthographie, Part-of-Speech, Kongruenz, Valenz, Semantik.
- Die meiste Information ist dabei im Typ kodiert.

Lexikoneinträge

- Zum Beispiel „lacht“:

Feature-Struktur:

word

ORTH „lacht“
HEAD *verb*
SPR < [HEAD noun] >
COMPS < >

Eintrag im Lexikon:

lacht := word &
[ORTH „lacht“
HEAD verb,
SPR < [HEAD noun] >,
COMPS < >].

Bzw. in der ERG:

```
laugh_v1 := v_unerg_le &
 [ STEM < "laugh" >,
   SYNSEM.LKEYS.KEYREL.PRED "_laugh_v_rel" ].
```

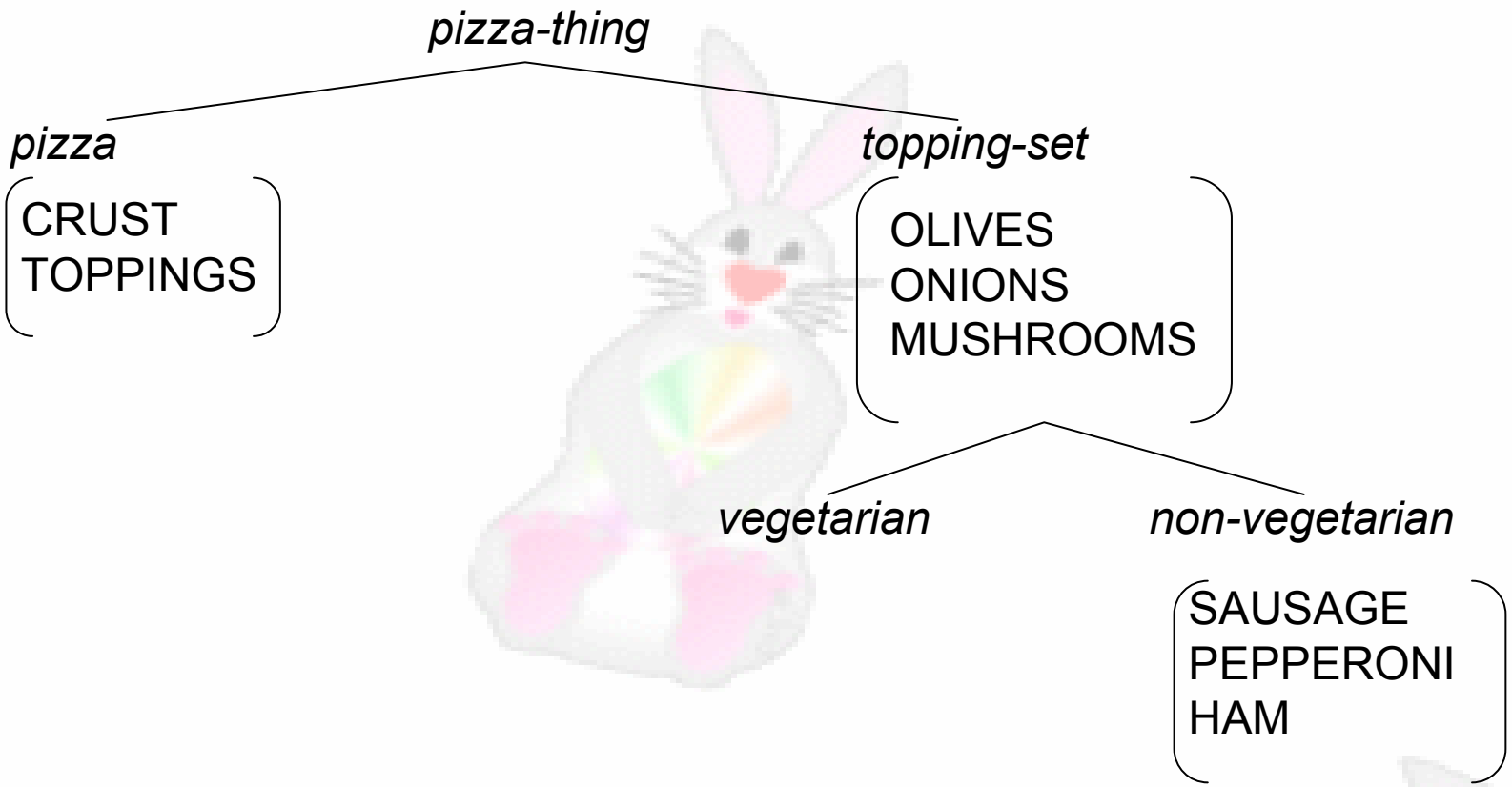
Eine Typhierarchie…

- …beschreibt die existierenden Objekte.
- …organisiert diese in Klassen mit gemeinsamen Eigenschaften.
- …beschreibt die generellen Eigenschaften der Objektarten (Merkmale und Attribut-Wert-Deklarationen)

Typen

- Warum Klassifikation in Typen?
  - Vermeidung von unsinnigen Feature-Spezifikationen, z.B. Kasus bei Verben
  - Möglichkeit, Restriktionen für ganze Typen von Elementen anzugeben.
  - Hierarchische Anordnung erlaubt Generalisierungen und vermindert Redundanzen (und damit auch Fehler).

Die Pizza-Typhierarchie

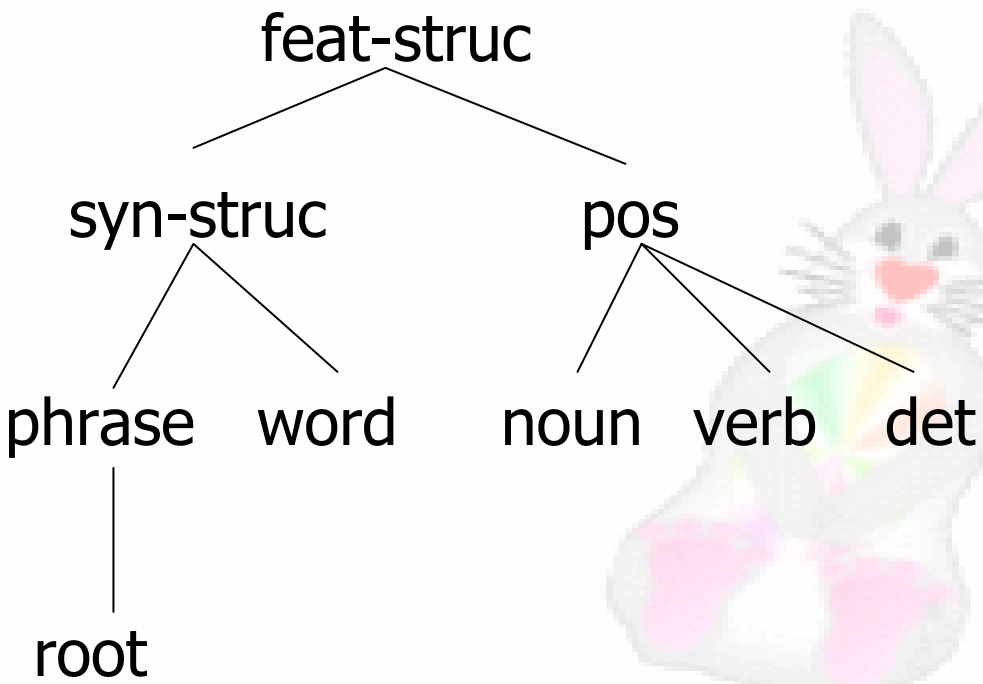


Die Pizza-Typhierarchie in TDL

```
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].
```

…

Typen - Beispiel

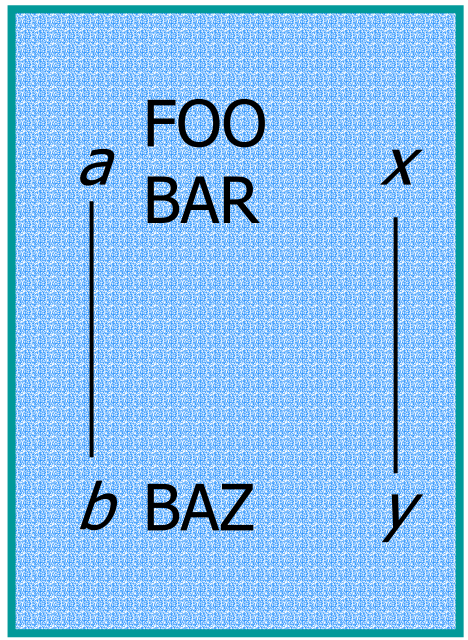



```
feat-struc := *top*.
syn-struc := feat-struc &
                [ HEAD pos,
                  SPR *list*,
                  COMPS *list* ].
pos := feat-struc.
noun := pos.
verb := pos.
det := pos.
phrase := syn-struc &
                [ ARGS *list* ].
word := syn-struc &
                [ ORTH string ].
root := phrase &
                [ HEAD verb,
                  SPR < >,
                  COMPS < > ].
```

Unifikation: Kombination von Information, Sicherung von Kompatibilität

TFS_1: $a\begin{pmatrix} \text{FOO} & x \\ \text{BAR} & x \end{pmatrix}$ TFS_2: $a\begin{pmatrix} \text{FOO} & x \\ \text{BAR} & y \end{pmatrix}$

TFS_3: $b\begin{pmatrix} \text{FOO} & y \\ \text{BAR} & x \\ \text{BAZ} & x \end{pmatrix}$ TFS_4: $a\begin{pmatrix} \text{FOO} & \boxed{1} & x \\ \text{BAR} & \boxed{1} \end{pmatrix}$



$TFS_1 \sqcap TFS_2 \equiv TFS_2$; $TFS_1 \sqcap TFS_3 \equiv TFS_3$; $TFS_3 \sqcap TFS_4 \equiv b\begin{pmatrix} \text{FOO} & \boxed{1} & y \\ \text{BAR} & \boxed{1} & \\ \text{BAZ} & & x \end{pmatrix}$

Pizza-Unifikation

pizza		
CRUST	thick	
TOPPINGS	OLIVES	+
	HAM	-

⊔

pizza		
TOPPINGS	OLIVES	+
	ONIONS	+

≡

pizza		
CRUST	thick	
TOPPINGS	OLIVES	+
	ONIONS	+
	HAM	-

Subkategorisierung und Kongruenz

- Die COMPS und SPR Features eines lexikalischen Items spezifizieren eine *Liste* von subkategorisierten Feature-Strukturen und Kongruenzrestriktionen für diese Strukturen.
- Die `head-complement` und `head-specifier` Regeln spezifizieren, wie Bäume zusammengesetzt werden.