# Merkmale und Vererbung in unifikationsbasierten Lexika

**Morphologieanalyse und Lexikonaufbau (14. Vorlesung)**

*Dozent: Gerold Schneider*

## Übersicht



---

# 1. Was sind Merkmalstrukturen (feature structures)?

Eine Merkmalstruktur ist eine partielle Abbildung von Merkmalen auf Werte.

Merkmalstrukturen sind mehrfachverwendbar (engl. *reentrant* ), d.h. mehrere Merkmale können sich einen gemeinsamen Wert teilen.

## 1.1. Beziehungen zwischen Merkmalstrukturen: Subsumption

Durch Subsumption ergibt sich eine natürliche Verbandsstruktur über Merkmalstrukturen. D subsumiert D', wenn D eine Teilmenge der Information von D' enthält.

## 1.2. Operationen über Merkmalstrukturen: Unifikation

Die Unifikation von zwei Merkmalstrukturen D' und D'' ist die allgemeinste Mermalstruktur D, so dass gilt: D' subsumiert D, und D'' subsumiert D.

## 1.3. Typisierte Merkmalstrukturen

Bei typisierten Merkmalstrukturen können nicht alle Merkmale überall auftreten, sondern nur in Merkmalstrukturen eines bestimmten Typs. Es gibt ein Schema, das vorschreibt, wann ein Merkmal auftreten kann und welchen Wertebereich ein Merkmal hat.

# 2. Repräsentation von Lexikonstrukturen über Merkmalstrukturen

(nach [Ide et al. 93]: Outline of a model for lexicaldatabases. In: Information Processing & Management. 29(2). Die folgenden *Fig.*-Angaben beziehen sich auf Abbildungen aus diesem Paper.)

### 2.1. einfaches Beispiel: ein Lexikoneintrag als Merkmalstruktur (*competitor*; Fig. 17)

Problem: einfache Merkmalstrukturen sind nicht mächtig genug, um Lexikonstrukturen zu repräsentieren. ==> man braucht Disjunktion

### 2.2. Beispiel mit Wert-Disjunktion

Ein Merkmal in einer Merkmalstruktur hat disjunktive atomare Werte.

Bsp.: alternative Orthographie in Fig.18 und alternative Orthographie und Aussprache in Fig.19

Problem: Wie funktioniert die Unifikation von Merkmalstrukturen mit Disjunktion?

```
Bsp.:        [a:X]     Û [a:(b,c)]     = [a:(b,c)]
             [a:b]     Û [a:(b,c)]     = [a:b],      da [a:b] Û [a:c] = fail
             [a:(b,c)] Û [a:(c,d)]     = [a:c]
```

## 2.3 Allgemeine Disjunktion (zur Spezifikation alternativer Unterteile eines Eintrags)

In einer Merkmalstruktur gibt es disjunktive Unterstrukturen.

Bsp.: alternative Orthographie mit disjunktiven Merkmalstrukturen auf oberster Ebene in Fig. 20a
alternative Orthographie mit disjunktiven Merkmalstrukturen in Fig. 20b
alternative Orthographie mit diskunktiven Merkmalstrukturen und Ausgeklammerung in Fig. 20c

- erlaubt das Faktorisieren ("Ausklammern") von gemeinsamen Teilen
- erlaubt die Repräsentation von unterschiedlichen Bedeutungen (Fig. 21) und komplexen Schachtelungen (Fig. 22)
- Merkmalstrukturen werden auf *hierarchical normal form* beschränkt. D.h. in einer Merkmalstruktur gibt es nur eine Disjunktion. Damit entspricht die Merkmalstruktur einer Baumstruktur. (Fig. 24 ist Baum zuFig. 22)
- eine *unfactor*-Operation kann Merkmalstrukturen mit Disjunktionüberführen in eine Merkmalstruktur ohne faktorisierende Disjunktion.(Fig. 25) Auf diese Art und Weise kann die Disjunktion bis auf die oberste Ebene eliminiert werden. Eine Merkmalstruktur, die nur noch Disjunktion auf der obersten Ebene enthält, ist in *disjunctive normal form* (DNF). Indem man zwei MS in DNF bringt, kann man ihre Gleichheit überprüfen.

## 2.4 Implementation in einem objekt-orientierten Datenbank-System

Datenbank

> ist "ein System zur Beschreibung, Speicherung und Wiedergewinnung von umfangreichen Datenmengen, die von mehreren Anwendungsprogrammen benutzt werden. Es besteht aus der Datenbasis, in der die Daten abgelegt werden, und den Verwaltungsprogrammen, die die Daten entsprechend den vorgegebenen Beschreibungen abspeichern, auffinden oder weitere Operationen mit den Daten durchführen." (nach Duden Informatik. 1993. S. 157)

Gründe für eine objekt-orientierte DB:

- Objekt-orientierte DB-Systeme bieten die geforderte Ausdrucksstärke und Flexibilität.
- Objekt-orientierte DB-Systeme erlauben strukturierte Objekte, rekursive Typen, Listen und Mengen.
- Objekte werden als Ganzes betrachtet.

# 3. Unifikation im NLP-Lexikon

Lexikoneinträge enthalten vielfach identische Informationen. So benötigen alle Verben:
`[cat: v]`

und alle finiten Verben:
`[cat: v, form: finite]`

Diese identische Information kann in einer Hierarchie (von allgemein zu spezifisch) angeordnet und entsprechend vererbt werden, so dass nur die jeweils abweichende Information neu kodiert werden muss.

## 3.1. Vererbung in PATR-II über lexikalische Templates

(Bsp. [Shieber 86] S.57)

## 3.2. Vererbung mit Default-Werten

(Bsp. Alle Verben nehmen`[subcat:first:NP(nom)]` als Komplement; Ausnahmen wie z.B. "Mir graut vor dir."erhalten eine Sondermarkierung `[subcat:first:NP(dat)]`, die den Defaultüberschreibt.)

## 3.3. Transformationen über Lexikoneinträge (lexikalische Regeln)

(Bsp. Aktiv-Passiv Transformation bzgl. Subkategorisierung)

Im GTU-System wird das über Lexikon-Interface-Regeln gelöst:

```
if_in_lex (wortart=verb, subcat=nom_acc, diath=akt)
      then_in_gram  V[subcat=nom_acc, diath=akt, agr=...]

if_in_lex (wortart=verb, subcat=nom_acc, diath=pass)
      then_in_gram  V[subcat=nom, diath=pass, agr=...]
```

# 4. DATR

Lit.: [Evans und Gazdar 96]

DATR ist eine deklarative Sprache zur Repräsentation einerbeschränkten Klasse von Vererbungsnetzwerken. Es erlaubt sowohl multipleals auch Default-Vererbung.

Wichtigstes Anwendungsgebiet: Lexikoneinträge für die Verarbeitungnatürlicher Sprache

Motivation: eine Sprache, mit der man ausdrücken kann, dass ein Lexem regulär ist bis auf bestimmte aussergewöhnliche Eigenschaften.

Ziele: DATR ist eine Sprache, die

1. genügend ausdrucksstark ist, um Lexikoneinträge mitMerkmalstrukturen zu repräsentieren.
2. alle Generalisierungen über Lexikoneinträge ausdrückenkann.
3. eine explizite Theorie der Inferenz enthält.
4. effizient verarbeitbar ist.
5. eine explizite deklarative Semantik hat.

Information in DATR ist organisiert in einem Netzwerk von Knoten (repräsentiert typischerweise ein Wort). Jeder Knoten ist assoziiert mit einer Menge von Pfad/Wert Paaren, wobei ein Pfad eine Sequenz von Atomen ist und ein Wert ein Atom oder eine Sequenz von Atomen.

## Beispiele zu DATR

**Verb:**

```
<syn cat> == v
<syn type> == main
<mor past> == ("<mor root>" _ed)
<mor pres tense> == "<mor root>"
<mor pres tense sing three> == ("<mor root>" _s)
<mor pres participle> == ("<mor root>" _ing)
```

Wenn es keine spezifischeren Angaben gibt, erben alle Unterknoten von **Verb** diese Angaben. Also auch

```
<mor past participle> == ("<mor root>" _ed)
<mor past tense sing one> == ("<mor root>" _ed)
<mor past tense sing two> == ("<mor root>" _ed)  ...
```

Die Definition für ein Hilfsverb sieht dann so aus:

AUX:

```
< > == VERB
<syn type> == aux
<syn args> == VPCOMP
```

Ein Modalverb kann angesehen werden als ein Spezialfall eines Hilfsverbs mitfolgender Definition:

MODAL1:

```
< > == AUX.
<syn form> == finite
```

Eintrag für ein unregelmässiges Verb

BE_MOR:

```
<mor> == VERB
<mor root> == be
<mor past participle> == been
<mor past tense sing one> == <mor past tense sing three>
<mor past tense sing three> == was
<mor past tense> == were
<mor pres tense sing one> == am
<mor pres tense sing three> == is
<mor pres tense> == are.
```

## Spezifische Varianten:

1. *be* im Satz *Kim is a child.*

```
Be1:
< > == AUX
<mor> == BE_MOR
<syn args> == NPCOMP
```

2. *be* im Satz *Kim is to leave.*

```
Be2:
< > == MODAL1
<mor> == BE_MOR
```

3. *be* im Satz *Kim is leaving.*

```
Be3:
< > == AUX
<mor> == BE_MOR
<syn args car syn form> == prp
```

## Fragen zu DATR

1. Ist DATR nicht nur eine Variante von PATR?

   Nein. PATR wurde entwickelt für Grammatiken und DATR für Lexikoneinträge. Unifikation ist zentral in PATR aber nur marginal in DATR, während Default-Vererbung zentral ist für DATR aber nur wenig wichtig für PATR.

2. Ist DATR irgendeiner Grammatiktheorie verpflichtet?

   Nein. Es kann an unterschiedliche Theorien angebunden werden. Es eignet sich besonders für Theorien, die auf Merkmalstrukturen basieren.