# Syntax natürlicher Sprachen

## Vorlesung 10: Getypte Merkmalstrukturen und Unifikation

A. Wisiorek
Folien: Martin Schmitt

Centrum für Informations- und Sprachverarbeitung,
Ludwig-Maximilians-Universität München

21.12.2021

# Einsatzbereiche von Unifikation

- Logik-Programmierung (z. B. Prolog)
- Pattern Matching (z. B. in funktionalen Programmiersprachen)
- Typinferenz (vor allem funktionale Programmiersprachen wie Haskell, Scala, etc. Eingeschränkt aber z. B. auch C#)
- Merkmalstrukturen (zur Beschreibung komplexer Objekte, z. B. grammatischer Merkmale)

# Getypte Merkmalstrukturen

- Carpenter, Bob (1992). *The Logic of Typed Feature Structures*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press.

1. Formale Grundlagen
   - Merkmalstrukturen
   - Subsumption
   - Unifikation
   - Bedingungen

2. *Implementierung
   - *Merkmalstrukturen
   - *Unifikation

# 1.1. Merkmalstrukturen

# Typen

## Definition (Type, $\sqsubseteq$)

Sei **Type** eine endliche Menge von Typen mit **Vererbungshierarchie** $\sqsubseteq$.

Wenn für $A, B \in$ **Type** gilt, dass $A \sqsubseteq B$, dann

- *erbt* $B$ Informationen von $A$.
- ist $A$ Obertyp von $B$. (Alle $A$-Attribute sind auch $B$-Attribute.)
- $A$ *subsumiert* $B$ ($B$ wird von $A$ subsumiert).
- ist $A$ „allgemeiner oder gleich" $B$.
- ist $B$ „spezieller oder gleich" $A$.

# Vererbung

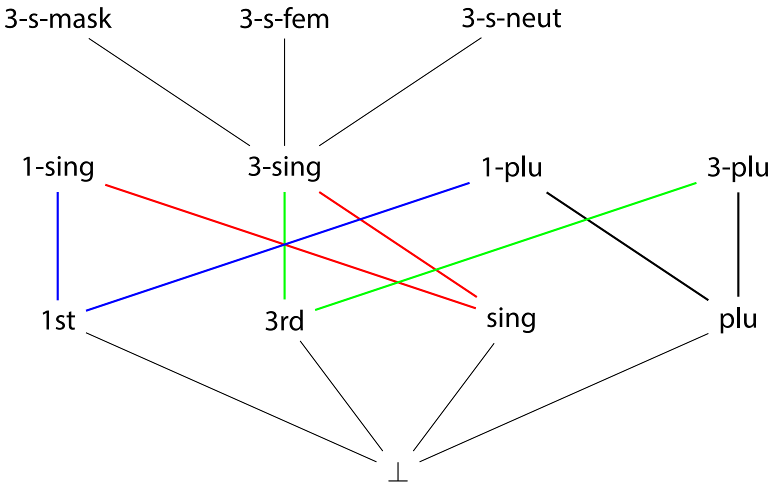
## Eigenschaften von Type, $\sqsubseteq$

- wohldefinierte Unifikationsoperation
- transitiv ($\forall A, B, C \in \textbf{Type}.\ A \sqsubseteq B \land B \sqsubseteq C \implies A \sqsubseteq C$)
- reflexiv ($\forall A \in \textbf{Type}.\ A \sqsubseteq A$)
- antisymmetrisch ($\forall A, B \in \textbf{Type}.\ A \sqsubseteq B \land B \sqsubseteq A \implies A = B$) (keine Vererbungsschleifen)

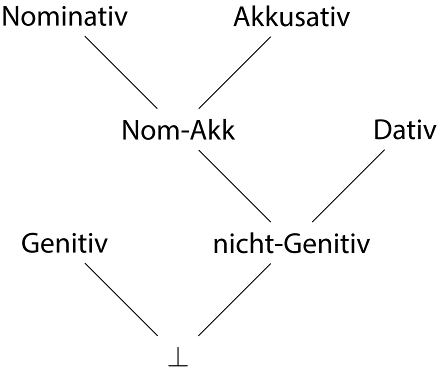
  $\Rightarrow$ *partielle Ordnung* (d. h. nicht alle Elemente von **Type** müssen miteinander vergleichbar sein)
- Existenz eines eindeutigen allgemeinsten Typs ($\exists_1 A \in \textbf{Type}.\ \forall B \in \textbf{Type}.\ A \sqsubseteq B$)

  $\Rightarrow \perp$ definiert als kleinstes Element von **Type** bzgl. $\sqsubseteq$

# Beispiel: Typhierarchie



# Noch ein Beispiel: Typhierarchie




Vgl. z. B. die Paradigmen:

der Hund, des **Hundes**, dem Hund, den Hund

**das** Buch, des **Buches**, dem Buch, **das** Buch

# Merkmale

## Definition (Feat)

Sei **Feat** eine *endliche* Menge von Merkmalen (engl. *features*).

(Ohne weitere Anforderungen an Struktur oder Eigenschaften)

## Beispiel

**Feat** $= \{$GEN, CASE, NUM, AGR, PER, MOOD, CAT, TENSE$\}$

# Merkmalstrukturen

## Definition

Eine Merkmalstruktur über **Type** und **Feat** ist definiert als Tupel $F = (Q, \bar{q}, \theta, \delta)$ mit:

- $Q$ : endliche Menge von Knoten (Einträge)
- $\bar{q} \in Q$ : Wurzelknoten
- $\theta : Q \to \textbf{Type}$ : totale Typisierungsfunktion
- $\delta : \textbf{Feat} \times Q \to Q$ : partielle Merkmal-Wert-Funktion

Sei $\mathcal{F}$ die Menge aller Merkmalstrukturen.

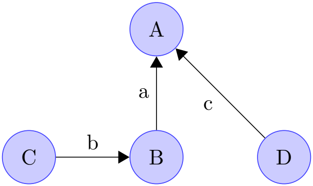
$$\begin{bmatrix} \text{CAT} & N \\ \text{AGR} & \begin{bmatrix} \text{CASE} & \mathit{Nom} \\ \text{NUM} & \mathit{Pl} \end{bmatrix} \end{bmatrix}$$

# Visualisierung als Graph I

## Beschrifteter Graph

Ein *beschrifteter* Graph ist definiert als Tupel $G = (V, E, l_V, l_E, L_V, L_E)$ mit

- $V$ : Menge der Knoten (engl. *vertices*)
- $E \subseteq V \times V$ : Menge der Kanten (engl. *edges*)
- $l_V : V \rightarrow L_V$ : Beschriftungsfunktion für Knoten (engl. *label*)
- $l_E : E \rightarrow L_E$ : Beschriftungsfunktion für Kanten
- $L_X$ : Menge von Beschriftungen für $X$



# Visualisierung als Graph II

## Visualisierung

Der Graph zu einer Merkmalstruktur $F = (Q, \bar{q}, \theta, \delta)$ ist gegeben durch:

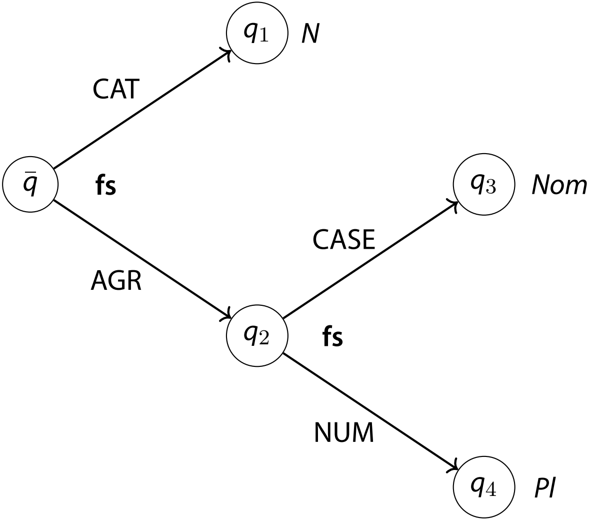
- $V := Q$
- $E := \{(q_1, q_2) \mid \exists f.\ \delta(f, q_1) = q_2\}$
- $L_V :=$ **Type**; $l_V := \theta$
- $L_E :=$ **Feat**; $l_E(q_1, q_2) := \{f \mid \delta(f, q_1) = q_2\}$

## Anmerkung

Zur Vereinfachung werden einelementige Mengen ohne Mengenklammern geschrieben.

Also $a$ statt $\{a\}$.

# Beispiel: Graphdarstellung



## Variablen

- **Var** sei eine abzählbar unendliche Menge von Variablen.
- Häufig wird $\mathbf{Var} = \mathbb{N}$ benutzt.
- Es gibt aber auch andere Möglichkeiten;
  z. B. im NLTK: ASCII-Identifier (`?x`, `?y`, $\ldots$)

## Definition (Zuweisungsfunktion, Valuation)

Eine Zuweisung $\alpha : \mathbf{Var} \to \mathcal{F}$ ist eine totale Funktion, die alle Variablen an Merkmalstrukturen (Knoten, Einträge) bindet.

# Reentrance

## Reentrance (dt. *Wiedereintritt*)

Durch das Aufstellen von Bedingungen (s. später) können Variablen an verschiedene Teile von Merkmalstrukturen gebunden werden. *Diese müssen <u>gleich</u> sein.*

## Beispiel

$$\begin{bmatrix} \text{ORTH} & \textit{folgt} \\ \text{SYN} & \begin{bmatrix} \text{SBJ} & \boxed{1} \\ \text{OBJ} & \boxed{2} \end{bmatrix} \\ \text{SEM} & \begin{bmatrix} \text{AGT} & \boxed{1} \\ \text{PAT} & \boxed{2} \end{bmatrix} \end{bmatrix} \qquad \begin{bmatrix} \text{ORTH} & \textit{folgt} \\ \text{SYN} & \begin{bmatrix} \text{SBJ} & \boxed{1}\textit{Hund} \\ \text{OBJ} & \boxed{2}\textit{Katze} \end{bmatrix} \\ \text{SEM} & \begin{bmatrix} \text{AGT} & \boxed{1} \\ \text{PAT} & \boxed{2} \end{bmatrix} \end{bmatrix}$$

# 1.2. Subsumption

1. Formale Grundlagen
   - Merkmalstrukturen
   - Subsumption
   - Unifikation
   - Bedingungen

2. **Implementierung*
   - **Merkmalstrukturen*
   - **Unifikation*

# Subsumption

## Erweiterung auf Merkmalstrukturen

$F = (Q, \bar{q}, \theta, \delta)$ subsumiert $F' = (Q', \bar{q}', \theta', \delta')$, genau dann wenn es eine totale Funktion $h : Q \rightarrow Q'$ gibt, sodass:

- $h(\bar{q}) = \bar{q}'$
- $\theta(q) \sqsubseteq \theta'(h(q))$ für alle $q \in Q$
- $h(\delta(f, q)) = \delta'(f, h(q))$ für alle $q \in Q$ und $f \in$ **Feat**,
  für die $\delta(f, q)$ definiert ist

## Beispiel

$$\bar{q}\begin{bmatrix} q_1\ \text{CAT} & N \end{bmatrix} \sqsubseteq \bar{q}' \begin{bmatrix} q_1'\ \text{CAT} & N \\ q_2'\ \text{GEN} & mask \end{bmatrix} \qquad h(\bar{q}) = \bar{q}'\ h(q_1) = q_1'$$

1. Formale Grundlagen
   - Merkmalstrukturen
   - Subsumption
   - Unifikation
   - Bedingungen

2. **Implementierung*
   - **Merkmalstrukturen*
   - **Unifikation*

# Unifikation I

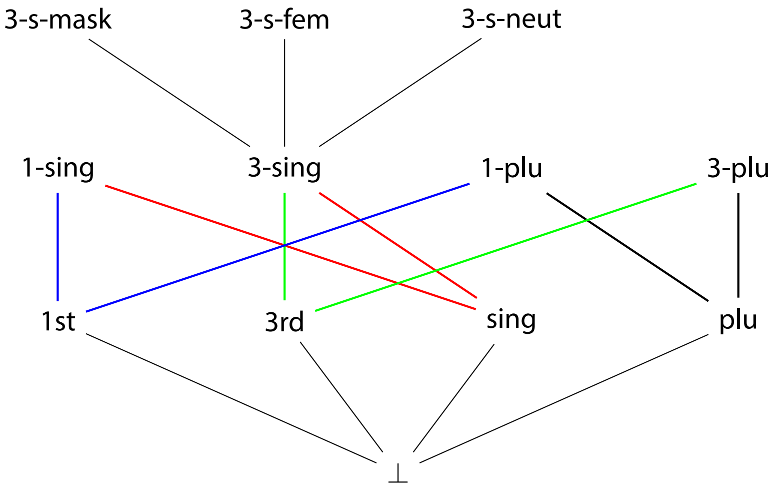
## Unifikation ($\sqcup$) für Typen

- Das Ergebnis der Unifikation zweier Typen $A, B \in$ **Type** ist ihre kleinste obere Schranke in **Type** bzgl. $\sqsubseteq$.
- Diese kann auch undefiniert sein (Typen unifizieren nicht).
- $A \sqcup B = C \iff A \sqsubseteq C$ und $B \sqsubseteq C$ und

$$\forall D \in \textbf{Type}.\ A \sqsubseteq D \land B \sqsubseteq D \implies C \sqsubseteq D$$

(Vgl. Mengenvereinigung und Untermengenbeziehung)

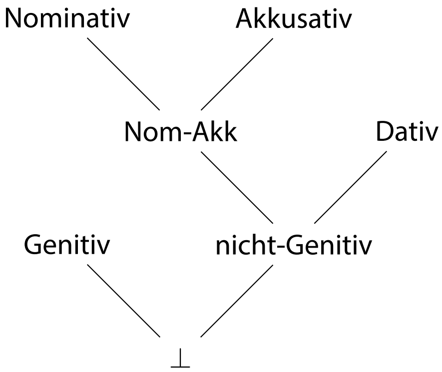
# Beispiel: Typunifikation




1st ⊔ plu = 1-plu

sing ⊔ 3-s-mask = 3-s-mask

# Noch ein Beispiel: Typunifikation




nicht-Genitiv $\sqcup$ Nominativ = Nominativ

Nom-Akk $\sqcup$ Dativ = *undefiniert*

# Unifikation II

## Unifikation ($\sqcup$) für Merkmalstrukturen

- Idee: Unifikation ebenfalls kleinste obere Schranke bzgl. der Subsumptionsrelation $\sqsubseteq$ auf Merkmalstrukturen
- Algorithmus in zwei Schritten:
  1. Identifiziere korrespondierende (äquivalente) Knoten
  2. Unifiziere deren Typen

## Formale Definition: Identifikation (Schritt 1)

Für Merkmalstrukturen $F = (Q, \bar{q}, \theta, \delta)$ , $F' = (Q', \bar{q}', \theta', \delta')$ mit $Q \cap Q' = \emptyset$ sei die *Äquivalenzrelation* $\equiv$ wie folgt definiert:

- $\bar{q} \equiv \bar{q}'$
- $\delta(f, q) \equiv \delta'(f, q')$ wenn beide Seiten definiert und $q \equiv q'$

## Formale Definition: Typunifikation (Schritt 2)

Die Unifikation von $F$ und $F'$ ist dann wie folgt definiert:

$$F \sqcup F' = \big((Q \cup Q')/_{\equiv}, [\bar{q}]_{\equiv}, \theta^{\equiv}, \delta^{\equiv}\big)$$

mit

$$\theta^{\equiv}([q]_{\equiv}) = \bigsqcup \{(\theta \cup \theta')(q') \mid q' \equiv q\}$$

und

$$\delta^{\equiv}(f, [q]_{\equiv}) = \begin{cases} [(\delta \cup \delta')(f, q)]_{\equiv} & \text{falls } (\delta \cup \delta')(f, q) \text{ definiert} \\ \text{undefiniert} & \text{sonst} \end{cases}$$

## Notation (für $\equiv$ Äquivalenzrelation über $X$)

- $[x]_{\equiv} = \{y \in X \mid y \equiv x\}$
- $X/_{\equiv} = \{[x]_{\equiv} \mid x \in X\}$

# Beispiel: (Formale) Unifikation

$$q_1 \begin{bmatrix} q_2 \text{ CAT} & \textit{N} \\ q_3 \text{ AGR} & \begin{bmatrix} q_4 \text{ NUM} & \textit{Sg} \\ q_5 \text{ CAS} & \textit{nicht-Gen} \end{bmatrix} \end{bmatrix} \qquad q_6 \begin{bmatrix} q_7 \text{ ORTH} & \textit{Hund} \\ q_8 \text{ AGR} & \begin{bmatrix} q_9 \text{ NUM} & \textit{Sg} \\ q_{10} \text{ CAS} & \textit{Nom} \end{bmatrix} \end{bmatrix}$$

1. Identifikation korrespondierender Knoten
    - $q_1 \equiv q_6$ (Initialisierung)
    - Nach 1 Schritt mit $\delta$:
        - $q_3 \equiv q_8$
    - Nach 2 Schritten mit $\delta$:
        - $q_4 \equiv q_9$
        - $q_5 \equiv q_{10}$

# Beispiel: (Formale) Unifikation

$$q_1 \begin{bmatrix} q_2 \text{ CAT} & N \\ q_3 \text{ AGR} & \begin{bmatrix} q_4 \text{ NUM} & Sg \\ q_5 \text{ CAS} & \textit{nicht-Gen} \end{bmatrix} \end{bmatrix} \qquad q_6 \begin{bmatrix} q_7 \text{ ORTH} & \textit{Hund} \\ q_8 \text{ AGR} & \begin{bmatrix} q_9 \text{ NUM} & Sg \\ q_{10} \text{ CAS} & \textit{Nom} \end{bmatrix} \end{bmatrix}$$

2. Typunifikation
   - $Q_U = \{\{q_1, q_6\}, \{q_2\}, \{q_7\}, \{q_3, q_8\}, \{q_4, q_9\}, \{q_5, q_{10}\}\}$
   - $\bar{q}_U = \{q_1, q_6\}$
   - $\theta^{\equiv}(\{q_2\}) = N$, $\theta^{\equiv}(\{q_7\}) = \textit{Hund}$, $\theta^{\equiv}(\{q_3, q_8\}) = \textbf{fs}$, $\theta^{\equiv}(\{q_4, q_9\}) = Sg$, $\theta^{\equiv}(\{q_5, q_{10}\}) = \textit{Nom}$, $\theta^{\equiv}(\{q_1, q_6\}) = \textbf{fs}$
   - $\delta(\text{CAT}, \{q_1, q_6\}) = \{q_2\}$, $\delta(\text{ORTH}, \{q_1, q_6\}) = \{q_7\}$, …

# Theoretische Resultate

**Lemma**

Wenn $F \sqcup F'$ definiert ist, dann ist $F \sqcup F' \in \mathcal{F}$ eine Merkmalstruktur.

**Theorem**

$F \sqcup F'$ ist die *kleinste obere Schranke* von $F$ und $F'$ in $(\mathcal{F}, \sqsubseteq)$, falls $F$ und $F'$ eine obere Schranke haben.

Für Beweise siehe (**Carpenter:Log-TyFeat**).

# 1.4. Bedingungen

# Bedingungen I

## Pfade

- Sequenzen von Merkmalen werden *Pfade* genannt.
- **Path** $=$ **Feat**$^*$ sei die Menge aller Pfade.
- Für $p \in$ **Path**, $F \in \mathcal{F}$ sei $F@p$ der Knoten in $F$, den man am Ende von Pfad $p$ erhält.

## Beispiele

- AGR–NUM
- SYN–SBJ–AGR–NUM
- ORTH
- $\varepsilon$ (der leere Pfad)

# Bedingungen II

## Definition (Beschreibung Desc)

Die Menge der Beschreibungen über **Type** und **Feat** sei die kleinste Menge, die folgende Bedingungen erfüllt:

- $A \in$ **Desc**, für alle $A \in$ **Type**
- $p : d \in$ **Desc**, für $p \in$ **Path**, $d \in$ **Desc**
- $x \in$ **Desc**, für alle $x \in$ **Var**
- $d \wedge e \in$ **Desc**, für $d, e \in$ **Desc**

## Beispiel

- AGR–NUM : *Sg*
- SYN–SBJ : $\boxed{1} \wedge$ SEM–AGT : $\boxed{1}$

## Erfülltheit

Die Erfülltheitsrelation $\models^{\alpha}$ zwischen Merkmalstrukturen und Beschreibungen ist gegeben durch:

- Für $A \in$ **Type**, $F \models^{\alpha} A \iff A \sqsubseteq \theta(\bar{q})$
- $F \models^{\alpha} p : d \iff F@p \models^{\alpha} d$
- Für $x \in$ **Var**, $F \models^{\alpha} x \iff \alpha(x) = F$
- $F \models^{\alpha} d \wedge e \iff F \models^{\alpha} d$ und $F \models^{\alpha} e$

**Sei $F$ eine Merkmalstruktur.**

$$F = \begin{bmatrix} \text{CAT} & \boxed{1} N \\ \text{POS} & \boxed{2} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & \mathit{Sg} \\ \text{CAS} & \mathit{Nominativ} \end{bmatrix} \end{bmatrix}$$

$$\alpha(\boxed{1}) = \alpha(\boxed{2})$$

**Welche Beschreibungen aus Desc erfüllt $F$?**

- $F \models^{\alpha} N$ ? Nein!
- $F \models^{\alpha} \text{CAT} : N$ ? Ja!
- $F \models^{\alpha} \text{AGR–CAS} : \mathit{nicht\text{-}Genitiv}$ ? Ja!
  Denn: $\mathit{nicht\text{-}Genitiv} \sqsubseteq \mathit{Nominativ}$
- $F \models^{\alpha} \text{POS} : N$ ? Ja!

# Beschreibungen als Merkmalstrukturen

## *MGSat* (allgemeinster Erfüller)

Zu jeder *konsistenten* (widerspruchsfreien) Beschreibung $d \in$ **Desc** gibt es eine Merkmalstruktur $MGSat(d) \in \mathcal{F}$ mit der Eigenschaft

$$\forall F \in \mathcal{F}.\, F \models d \iff MGSat(d) \sqsubseteq F$$

## Konstruktion

- Für $A \in$ **Type**: $MGSat(A) = \begin{bmatrix} \text{A} \end{bmatrix}$
- $MGSat(f_1 f_2 \ldots f_n : d) = \begin{bmatrix} f_1 & \begin{bmatrix} f_2 & \ldots \begin{bmatrix} f_n & MGSat(d) \end{bmatrix} \end{bmatrix} \end{bmatrix}$
- Wenn **Var** $= \mathbb{N}$, dann $MGSat(1) = \begin{bmatrix} \boxed{1} \end{bmatrix}$
- $MGSat(d \wedge e) = MGSat(d) \sqcup MGSat(e)$

# Bedingungsprüfung per Unifikation: Beispiel

## Grammatikregel mit Constraint

```
NP[CAS=?y] →DET[GEN=?x,CAS=?y] N[GEN=?x]
```

## Bedingungen als Beschreibungen

- type : *NP* ∧ CAS : $\boxed{2}$
- type : *DET* ∧ GEN : $\boxed{1}$ ∧ CAS : $\boxed{2}$
- type : *N* ∧ GEN : $\boxed{1}$

## Bedingungen als Merkmalstrukturen

$$\begin{bmatrix} \text{type} & \textit{NP} \\ \text{CAS} & \boxed{2} \end{bmatrix} \rightarrow \begin{bmatrix} \text{type} & \textit{DET} \\ \text{GEN} & \boxed{1} \\ \text{CAS} & \boxed{2} \end{bmatrix} \begin{bmatrix} \text{type} & \textit{N} \\ \text{GEN} & \boxed{1} \end{bmatrix}$$

1. Formale Grundlagen
   - Merkmalstrukturen
   - Subsumption
   - Unifikation
   - Bedingungen

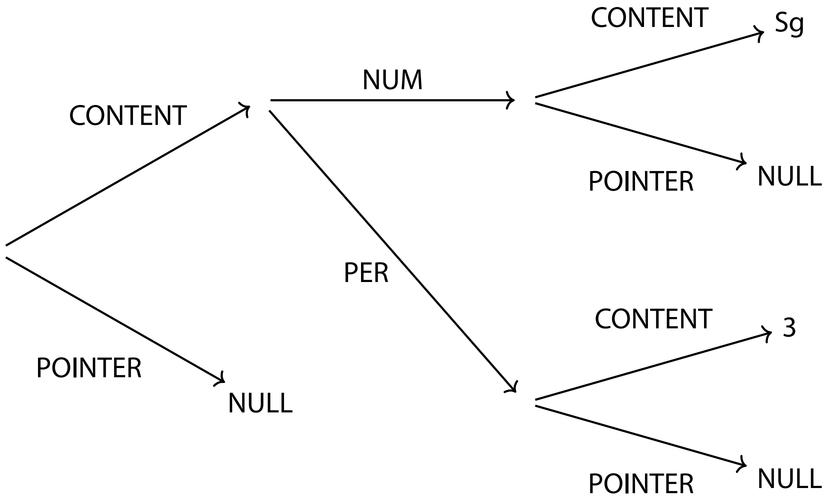
2. *Implementierung*
   - **Merkmalstrukturen*
   - **Unifikation*

# 2.1. *Merkmalstrukturen*

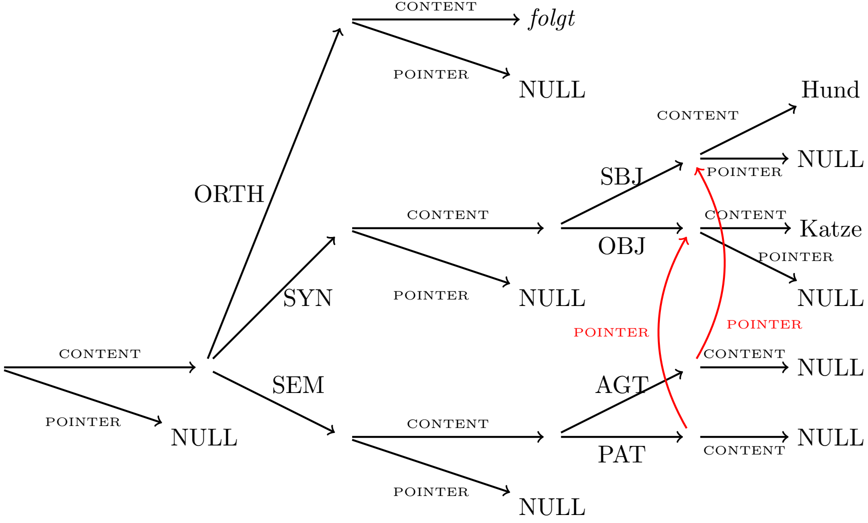
1. Formale Grundlagen
   - Merkmalstrukturen
   - Subsumption
   - Unifikation
   - Bedingungen

2. *Implementierung
   - *Merkmalstrukturen
   - *Unifikation

# Merkmalstrukturen: Implementierung mit Zeigern



# Reentrance (Wiedereintritt)



# 2.2. *Unifikation*

1. Formale Grundlagen
   - Merkmalstrukturen
   - Subsumption
   - Unifikation
   - Bedingungen

2. **Implementierung*
   - **Merkmalstrukturen*
   - **Unifikation*

# Unifikationsalgorithmus

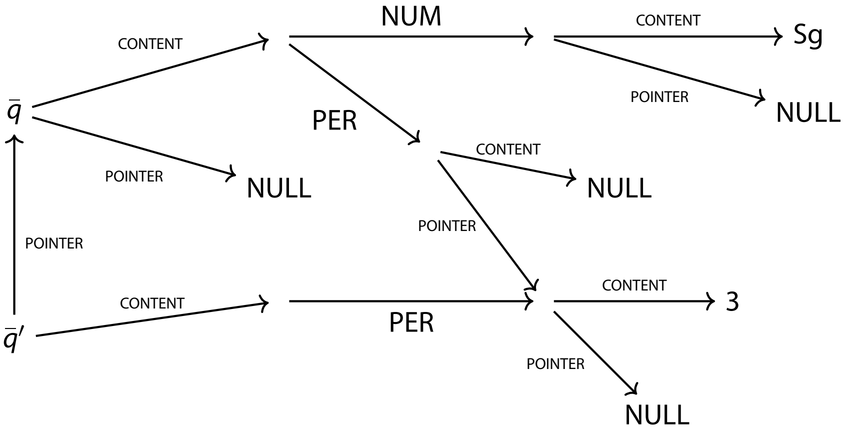
```
function unify(f1-orig, f2-orig)
    f1 ← deref(f1-orig)                                  ▷ Verfolgen von .pointer
    f2 ← deref(f2-orig)
    if f1, f2 ∈ Type then
        return unifyValues(f1, f2)                       ▷ z. B. per Typhierarchie
    if f1, f2 ∈ 𝓕 then
        for all feat2 ∈ f2 do
            feat1 ← erstelle oder finde entsprechendes Feature in f1
            if unify(feat1, feat2) = failure then
                return failure
        return f1
    ...
```

# Unifikationsalgorithmus II

```
function unify(f1-orig, f2-orig)
    …
    if f1 ∈ ContPoint ∧ f1.content is NULL then
        f1.pointer ← f2
        return f2
    if f2 ∈ ContPoint ∧ f2.content is NULL then
        f2.pointer ← f1
        return f1
    if f1, f2 ∈ ContPoint then
        f2.pointer ← f1
        return unify(f1.content, f2.content)
    return failure                        ▷ Kein Fall vorher hat zugetroffen
```

# Unifikation: Implementierung mit Zeigern

$$\begin{bmatrix}\text{NUM} & \mathit{Sg}\end{bmatrix} \sqcup \begin{bmatrix}\text{PER} & \mathit{3}\end{bmatrix}$$



# Rückblick auf heutige Themen

1. Formale Grundlagen
   - Merkmalstrukturen
   - Subsumption
   - Unifikation
   - Bedingungen

2. **Implementierung*
   - **Merkmalstrukturen*
   - **Unifikation*