

Semantik komparierter Adjektive
Seminar „Relationale Grammatik“
CIS, WS 2009/10

Hans Leiß

Universität München
Centrum für Informations- und Sprachverarbeitung

11.01.2010

Die Semantik von Adjektiven bei Böttner

Erinnerung:

Klassifikatorische Adjektive (CA)

Klassifikatorische Adjektive, z.B. Farb- und Formadjektive,

- können nicht (oder schlecht) kompariert werden,
- bezeichnen eine Eigenschaft von Individuen

Sie haben in der Peirce-Algebra den Typ set .

Prädikativ: $S \rightarrow NP \text{ CopV } CA \ (E(\text{ProdS}(NP, CA)))$

Attributiv: $N' \rightarrow CA \ CN \ (\text{ProdS}(CA, CN))$

Die Reihenfolge mehrerer CA-Attribute hat keine Auswirkung:
weißes rundes Blatt = rundes weißes Blatt

„Intensive“ Adjektive (IA)

„Intensive“ Adjektive, z.B. klein, alt, hoch,

- können kompariert werden,
- können absolut (IApos: ist klein) und relativ (IAcomp: ist kleiner als) gebraucht werden,
- bezeichnen eine (partielle) Ordnung zwischen Individuen.

Sie haben in der Peirce-Algebra den Typ rel.

Prädikativ absolut:

$$S \rightarrow PN \text{ CopV IA} \quad (E(\text{ProdS}(PN, \text{PreIm}(IA, \{c\}))))$$

Prädikativ relativ:

$$S \rightarrow PN \text{ CopV IA als PN} \quad (E(\text{ProdS}(IA, PN1 \times PN2)))$$

Attributiv absolut:

$$N' \rightarrow \text{IApos CN} \quad (\text{ProdS}(CN, \text{PreIm}(IA, \{c\})))$$

Problem der IApos-Semantik von Böttner

Böttner interpretiert absolut gebrauchte IA als Eigenschaften:

ist klein = ist kleiner als das Standardelement c

Problematisch an diesem IApos = PreIm(IA, {c}) ist:

1. das Standardelement muß von der Vergleichsrelation abhängen („normal alt“ \neq „normal groß“), aber die Regeln benutzen ein von IA unabhängiges c ,
2. das Modell muß für jede benannte Vergleichsrelation und *jede definierbare Menge* ein Standardelement $c = c_{IA, N'}$ vorgeben: „kleiner Hund“ \neq „kleiner weißer Elephant“

Damit wird selbst ein endliches Modell kaum noch definierbar!

Alternative Semantik der IA

Wir wollen eine Semantik der IApos implementieren, die

1. die IApos über die Vergleichsrelation IAcomp gewinnt,
2. nicht auf der Existenz von „Standardelementen“ beruht,
3. in nicht-monotoner Weise auf Begleitnomen relativierbar ist:
 - a) $IA\ CN \subseteq CN$, z.B. *kleiner Hund* \subseteq *Hund*,
 - b) $CN_1 \subseteq CN_2 \not\Rightarrow IA\ CN_1 \subseteq IA\ CN_2$, da z.B.
Elephant \subseteq *Tier* $\not\Rightarrow$ *kleiner Elephant* \subseteq *kleines Tier*
4. extensional ist und daher nicht geeignet für „dimensionale“ Adjektive: *Betrüger* \subseteq *Mensch*, aber:
guter Betrüger = *gut* als *Betrüger* $\not\subseteq$ *guter Mensch*

Wegen 3.a) ist die Semantik nicht geeignet für intensionale A.:
vermeintlicher Betrüger $\not\subseteq$ *Betrüger*

Grundidee 1: Relativierung einer Ordnung

Relativiere eine globale Ordnung $<^U$ auf Mengen $N \subseteq U$ zu

$$<^N := <^U \cap N \times N$$

Das geht bei allen partiellen Ordnungen, die mit einer Abbildung $|\cdot| : U \rightarrow \mathbb{S}$ auf eine Skala $(\mathbb{S}, 0, <^{\mathbb{S}})$ definiert sind, also durch

$$a <^U b : \iff 0 \leq^{\mathbb{S}} |a| <^{\mathbb{S}} |b|,$$

auch wenn sie in der Peirce-Algebra nur als $<^U$: rel gegeben sind.

Das heißt: „kleines N“ = „klein unter den N“



Grundidee 2: „Halbierung“ einer Ordnung

Die Menge der bzgl. $<$ intuitiv „kleinen“ Elemente von N berechnen wir (ohne ein Standardelement $c \in N$) rekursiv nach folgender Intuition:

- kein $<$ -maximales Element von N ist klein, aber von den anderen sind alle $<$ -minimalen Elemente von N klein,
- ist N' die Menge der nicht-maximalen und nicht-minimalen Elementen von N , so sind die kleinen Elemente von N' auch kleine Elemente von N .

De facto berechnen wir rekursiv eine Teilrelation von $<^N$, den „unteren Teil $<^N_u$ von $<^N$ “, und nehmen dann das Urbild von N unter dieser Relation:

$$\textit{kleine } N := <^N_u : N$$

Berechnung in PA?

Beachte:

- die Relativierung von $A:\text{rel}$ auf $N:\text{set}$ kann in der Peirce-Algebra durch $A \cap N \times N$ erfolgen, wobei

$$A \times B := A^c \cap (B^c)^\smile = (A \times 1) \cap (1 \times B)$$

- eine Relation A ist eine partielle Ordnung, falls

$$I \subseteq A \wedge A; A \subseteq A \wedge A \cap A^\smile \subseteq I$$

- die Berechnung des „unteren Teils“ von $A:\text{rel}$ ist in einer Peirce-Algebra nicht möglich, da rekursive Berechnungen nicht vorgesehen sind!

Aber: für *endliches* U terminiert in $\mathcal{P}(U)$ jede rekursive Berechnung entlang der Inklusion \subseteq auf rel oder set .

Definition von $R_u = \text{minor}(R: \text{rel}) : \text{rel}$

Der „untere Teil“ $<_u$ einer (beliebigen!) Relation $<$ sei (rekursiv):

$$\text{minor}(R) := \begin{cases} \emptyset, & \text{falls } R \subseteq (R^\sim : 1 \times R : 1) \\ R \cap (\overline{R^\sim : 1} \times R^\sim : 1) \cup \\ \text{minor}(R \cap (R^\sim : 1 \times R : 1)), & \text{sonst,} \end{cases}$$

wobei $R : 1 = \text{Urbildbereich von } R$, $R^\sim : 1 = \text{Bildbereich von } R$

Also: $\text{minor}(R)$ enthält (i) alle von R -minimalen Elementen ausgehenden R -Kanten, und (ii) $\text{minor}(R \cap \text{Bild} \times \text{Urbild})$

Beispiele

- $\text{minor}(\text{Kreis}) = \emptyset = \text{minor}(\text{Clique})$
- $\text{minor}\{\langle a, b \rangle, \langle b, c \rangle, \langle c, d \rangle\} = \{\langle a, b \rangle\} \cup \text{minor}\{\langle b, c \rangle\} = \{\langle a, b \rangle\} \cup (\{\langle b, c \rangle\} \cup \text{minor}(\emptyset)) = \{\langle a, b \rangle, \langle b, c \rangle\}$
- $\text{minor}(<^{\mathbb{N}}) = \{0\} \times \mathbb{N} \cup \text{minor}(<^{\mathbb{N}_{>0}}) = \dots$ (divergiert)

Berechnung des unteren Teils von $R:rel$

Wir haben in der Signatur PEIRCE schon eine Operation

```
val Minor : rel -> rel
```

vorgesehen, die in der Konstruktion von $\mathcal{P}(U)$ so definiert wird:

```
fun Minor(R) =  
  let val domain = PreIm(R,UnitS)  
      val range = PreIm(Conv R,UnitS)  
      val S = ProdR(R, Cart(range, domain))  
  in  
    if Set.size(R) = Set.size(S) then ZeroR else  
    SumR(ProdR(R, Cart(CompS range, range)), Minor(S))  
  end
```

Ebenso haben wir in PEIRCE schon eine Operation

```
val Tc      : rel -> rel
```

zur Bildung der transitiven Hülle einer Relation vorgesehen, die in Peirce : UNIVERSUM -> PEIRCE rekursiv definiert ist:

```
fun Tc(R) =  
  let val T = SumR(Prod(R,R),R)  
  in  
    if Set.size(T) = Set.size(R)  
    then R else Tc(T)  
  end
```

Beachte: $Tc(R)$ und $Minor(R)$ terminieren in $\mathcal{P}(U)$ mit endlichem $R \subseteq U \times U$, sonst i.a. nicht.

Proposition(?): Für endliche R ist $minor(R^+) : 1 = minor(R) : 1$
Dann braucht man bei den IA im Lexikon die transitive Hülle nicht!

Beweis: Nach Definition ist

$$a \in minor(R) : 1 \iff a \text{ ist } R\text{-minimal} \vee \\ a \in minor(R \cap Bild(R) \times Urbild(R)) : 1$$

Wegen $Ur/Bild(R^+) = Ur/Bild(R)$ und R^+ -minimal = R -minimal
bleibt für $M(R) := Bild(R) \times Urbild(R)$ zu zeigen:

$$minor(R \cap M(R)) : 1 = minor(R^+ \cap M(R)) : 1$$

Wegen $(R \cap M(R))$ -minimal = $(R^+ \cap M(R))$ -minimal bleibt z.z.

$$minor(R \cap M(R) \cap M(R \cap M(R))) : 1 = minor(R^+ \cap M(R) \cap M(R \cap M(R))) : 1$$

Beispiel $<^{\text{Tiere}}$

$\langle \text{PA-GLR-R31/pg3.sml} \rangle \equiv$

```
structure Tiere = Peirce(  
  datatype element = a1 | a2 | a3      (* Ameisen *)  
                  | e1 | e2 | e3 | e4 (* Elephanten *)  
                  | h1 | h2 | h3 | h4 (* Hunde *)  
  
  val elements =  
      [a1,a2,a3,h1,h2,h3,h4,e1,e2,e3,e4]  
  
  exception NoValue  
  
  fun ConS "arbeiten" = [a1,a2]  
    | ConS "gesund"   = [a1,a3,e2,e3,h1,h4]  
    | ConS "Ameise"   = [a1,a2,a3]  
    | ConS "Elephant" = [e1,e2,e3,e4]  
    | ConS "Hund"     = [h1,h2,h3,h4]  
    | ConS "weiß"     = [a1,a2,e1,e2,h1,h3,h4]  
    | ConS "Tier"     = elements  
    | ConS _          = raise NoValue
```

Beispiel (Forts.)

$\langle PA-GLR/pg3.sml \rangle \equiv$

```
fun ConR "größer" =
    [(e4,e2),(e3,e2),(e2,e1),    (* partiell *)
     (h4,h3),(h3,h2),(h2,h1),
     (a3,a2),(a2,a1),          (* total *)
     (e1,h4),(h1,a3)]
| ConR "kleiner" =
    map (fn (x,y) => (y,x)) (ConR "größer")
| ConR _ = raise NoValue
)
```

structure NL3 : PARSE = Parse(structure Algebra = Tiere)

Hier wurde nur das Hasse-Diagramm $R - R^2$ angegeben; im Lexikon wird zur transitiven Hülle $R = (R - R^2)^+$ übergegangen, mit A.Tc der Peirce-Algebra Tiere.

Erweiterung der Grammatik

```
fun Dom(r)          = PreIm(r,UnitS)
fun Cart(s1,s2)     = ProdR(Cylin s1,Conv(Cylin s2))
fun MinorS(s,r)    = Dom(Minor(ProdR(r,Cart(s,s))))
%%
%term ...
  | CA of set | IA of rel | IAcomp of rel | als
%%
AP : CA              (CA)                                (* R4 *)
  | IAcomp UQ N'    (Exp(IAcomp,N'))                    (* R26 *)
  | IAcomp EQ N'    (PreIm(IAcomp,N'))                  (* R27 *)
  | IAcomp NQ N'    (CompS(PreIm(IAcomp,N'))))          (* R28 *)
N' : CN              (CN)                                (* R6a *)
...
  | CA N'           (ProdS(CA,N'))                      (* R29 *)
  | IA N'           (MinorS(N',IA))                    (* R31a *)
```

Erweiterung des Lexikons

```
fun CN s t      = ([T.CN(A.ConS s,!line,!line)],t)
fun IA r t      = ([T.IA(A.ConR r,!line,!line)],t)
fun IAcomp r t  = ([T.IAcomp(A.Tc(A.ConR r),!line,!line)],t)
%%
"älter"        => (IAcomp "älter" yytext);
"größer"       => (IAcomp "größer" yytext);
"kleiner"      => (IAcomp "kleiner" yytext);
("alt"|"alter"|"alte"|"altes") => (IA "älter" yytext);
("groß"|"großer"|"große"|"großes")
                                     => (IA "größer" yytext);
("klein"|"kleiner"|"kleine"|"kleines")
                                     => (IA "kleiner" yytext);
"Ameise"|"Ameisen"      => (CN "Ameise" yytext);
"Hund"|"Hunde"          => (CN "Hund" yytext);
"Elephant"|"Elephanten" => (CN "Elephant" yytext);
"Tier"|"Tiere"          => (CN "Tier" yytext);
```

$\langle PA-GLR-R31/pg3.cm \rangle \equiv$
Group structure NL3
is
 pg3.sml
 pg.parse.cm

Auswertungsbeispiele

- CM.make "pg3.cm";

IA1 IA2 CN \neq IA2 IA1 CN:

- NL3.evals "kleines Tier";

Absyn: (minor(kleiner . (Tier x Tier)) : 1)

val it = [un (Set [h2,h1,a3,a2,a1])]

- NL3.evals "großes Tier";

Absyn: (minor(größer . (Tier x Tier)) : 1)

val it = [un (Set [h3,h4,e1,e2,e4,e3])]

- NL3.evals "kleines großes Tier";

Absyn: (minor(kleiner . ((minor(größer . (Tier x Tier)) : 1)

 x (minor(größer . (Tier x Tier)) : 1))) : 1)

val it = [un (Set [h4,h3])]

- NL3.evals "großes kleines Tier";

val it = [un (Set [h1,h2])] : NL3.Semantics.value list

CA IA CN \neq IA CA CN:

- NL3.evals "kleine Hund";

Absyn: (minor(kleiner . (Hund x Hund)) : 1)

val it = [un (Set [h2,h1])] : NL3.Semantics.value list

- NL3.evals "weiße Hund";

Absyn: (weiß . Hund)

val it = [un (Set [h4,h3,h1])] : NL3.Semantics.value list

- NL3.evals "weiße kleine Hund";

Absyn: (weiß . (minor(kleiner . (Hund x Hund)) : 1))

val it = [un (Set [h1])] : NL3.Semantics.value list

- NL3.evals "kleine weiße Hund";

Absyn: (minor(kleiner . ((weiß . Hund) x (weiß . Hund)))) :

val it = [un (Set [h3])] : NL3.Semantics.value list

$CN_1 \subset CN_2 \not\Rightarrow CA CN_1 \subseteq CA CN_2$:

- NL3.evals "jede Ameise ist ein Tier";

Absyn: 1 = (-Ameise + Tier)

val it = [nu true] : NL3.Semantics.value list

- NL3.evals "jede große Ameise ist ein großes Tier";

Absyn: 1 = (- (minor(größer . (Ameise x Ameise)) : 1)
+ (minor(größer . (Tier x Tier)) : 1))

val it = [nu false] : NL3.Semantics.value list

- NL3.evals "jede kleine Ameise ist ein kleines Tier";

Absyn: 1 = (- (minor(kleiner . (Ameise x Ameise)) : 1)
+ (minor(kleiner . (Tier x Tier)) : 1))

val it = [nu true] : NL3.Semantics.value list

- NL3.evals "jede große Ameise ist ein kleines Tier";

Absyn: 1 = (- (minor(größer . (Ameise x Ameise)) : 1)
+ (minor(kleiner . (Tier x Tier)) : 1))

val it = [nu true] : NL3.Semantics.value list

Undokumentiert

In PA-GLR-R31 sind neben den hier angedeuteten Dateien

`pg.lex`, `pg.glr`, `pg3.cm`, `pg.parse.cm`

auch entsprechende

`pg-num.lex`, `pg-num.glr`, `pg3-num.cm`, `pg-num.parse.cm`

angelegt, die in Grammatik und Lexikon Unterschiede im Numerus vorsehen. Das ist aber nicht bei allen Kategorien durchgeführt (z.B. nicht bei Qu und N'.) Das ersetzt das Verzeichnis PA-GLR-R31-Num.

In PA-GLR-R31-NumKas waren entsprechende Unterschiede für Numerus und Kasus angelegt; allerdings ist das ziemlich unzureichend: Da der Lexer nur den *ersten Match* an den Parser liefert, muß man pro Wortform eine Liste *aller Lesarten* angeben, nicht jede Lesart separat als Einerliste!