Natuurlijke Taalverwerking I
Unificatie-Grammatica

Week 3

Overzicht

- Vervolg DCGs
- Unificatie-grammatica
- Subsumptie en unificatie
- Unificatie-grammatica in Hdrug
- Hdrug demo

Grammar engineering

- Ondersteun ontwikkeling en onderhoud van grote grammatica's
- Formalisme:
  - Begrijpelijke notatie
  - Expressiviteit
  - Gebruik macro's, overerving, generalisaties
- Platform:
  - Visualisatie
  - Evaluatie

Recapitulatie

- Context-vrije grammatica:
  - Simpel
  - Interne structuur van een categorie niet zichtbaar
  - Grote, herhalende grammatica's
- Definite Clause Grammar:
  - Krachtig
  - Categorieën met argumenten
  - Vervlechting met Prolog code

Definite Clause Grammars

- Het gebruik van argumenten groeit met de tijd:
  - `np(Num,Per,Sem) --> ...`
- We kunnen natuurlijk structuur bouwen:
  - `np(agr(Num,Per),Sem) --> ...`
- Structuur is niet geformaliseerd:
  - Geen eenvoudige toegang
  - Geen typing
- Parsing als deductie
  - Niet efficiënt
  - (Links-recursie)

Unificatie-grammatica

- Een variant van definite clause grammar.
- Argumenten worden aangegeven met attributen/waarden
  - *case, subj, agr, ...*
- Attributen worden verpakt in een attribute-value structure.
- Attribute-value structures zijn vergelijkbaar met Prolog-termen.
- Unificatie van attribute-value structures.
- Voordelen:
  - Leesbare, begrijpelijke code
  - Sluit aan bij taalkundige traditie

Een attribute-value structure

$$\begin{bmatrix} \text{num} & \text{pl} \\ \text{per} & 1 \end{bmatrix}$$

- Agreement voor ‘we’
- Structuur met de attributen *num* en *per*, met bijbehorende waarden.
- Kan ook gezien worden als een functie die *pl* retourneert voor *num* en *1* voor *per*.

Structuur in structuur

$$\begin{bmatrix} \text{cat} & \text{np} \\ \text{agr} & \begin{bmatrix} \text{num} & \text{pl} \\ \text{per} & 1 \end{bmatrix} \end{bmatrix}$$

- ‘we’ als NP.
- De waarde van een attribuut kan ook weer een attribute-value structure zijn.

Reentrancy

$$\begin{bmatrix} \text{cat} & \text{s} \\ \text{sem} & \begin{bmatrix} \text{subj} & \boxed{1}\ \text{Jip} \\ \text{pred} & \text{scheren} \\ \text{obj} & \boxed{1} \end{bmatrix} \end{bmatrix}$$

- Mogelijke semantiek voor ‘Jip scheert zichzelf’.
- Deze attribute-value structure is *reentrant*: twee attributen delen een waarde.
- De gedeelde waarden kunnen ook attribute-value structures zijn.
- Aangeduid met *co-indexing*.

Woordenboek

$$\begin{bmatrix} \text{cat} & \text{np} \\ \text{agr} & \begin{bmatrix} \text{num} & \text{sg} \\ \text{per} & 3 \end{bmatrix} \end{bmatrix} \rightarrow \text{Jip}$$

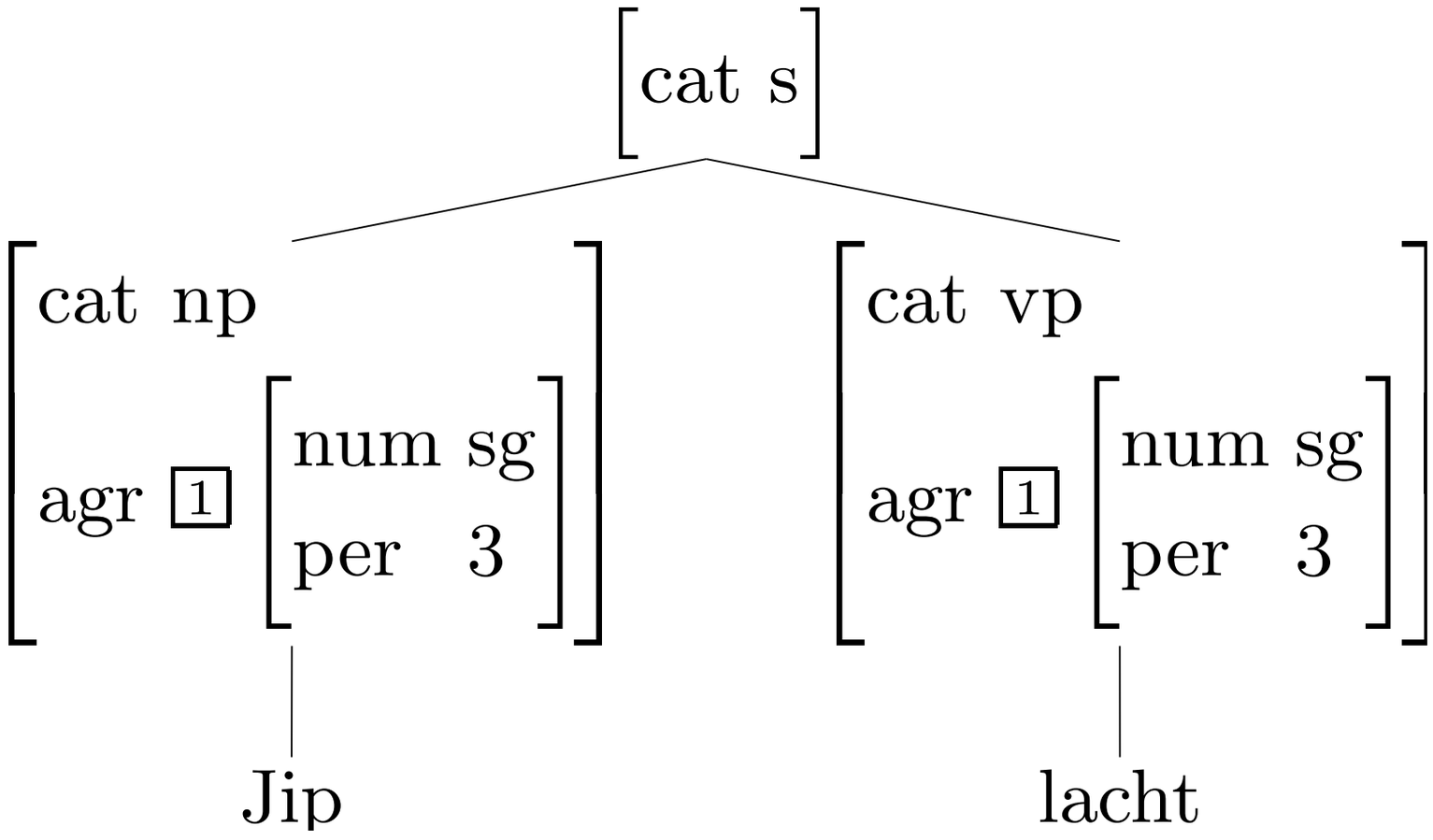
$$\begin{bmatrix} \text{cat} & \text{vp} \\ \text{agr} & \begin{bmatrix} \text{num} & \text{sg} \\ \text{per} & 3 \end{bmatrix} \end{bmatrix} \rightarrow \text{lacht}$$

Regels

$$\begin{bmatrix}\text{cat} & \text{s}\end{bmatrix} \rightarrow \begin{bmatrix}\text{cat} & \text{np} \\ \text{agr} & \boxed{1}\end{bmatrix}\begin{bmatrix}\text{cat} & \text{vp} \\ \text{agr} & \boxed{1}\end{bmatrix}$$

- Representatie van een categorie via een attribuut.
- Agreement via reentrancy/co-indexing.

Eenvoudig voorbeeld



- “Jip lacht”

SUBSUMPTIE EN UNIFICATIE

Subsumptie

- AV_1 subsumeert AV_2 ($AV_1 \sqsubset AV_2$) als AV_2 alle informatie bevat die AV_1 bevat.

$$\begin{bmatrix} \text{head} & \begin{bmatrix} \text{cat} & \text{v} \end{bmatrix} \end{bmatrix} \sqsubset \begin{bmatrix} \text{head} & \begin{bmatrix} \text{cat} & \text{v} \\ \text{vform} & \text{fin} \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \text{head} & \begin{bmatrix} \text{cat} & \text{v} \end{bmatrix} \end{bmatrix} \sqsubset \begin{bmatrix} \text{head} & \begin{bmatrix} \text{cat} & \text{v} \end{bmatrix} \\ \text{obj} & \text{nil} \end{bmatrix}$$

Subsumption (2)

$$\begin{bmatrix} \text{head} & \begin{bmatrix} \text{cat} & \text{v} \\ \text{vform} & \text{fin} \end{bmatrix} \end{bmatrix} \not\sqsubseteq \begin{bmatrix} \text{head} & \begin{bmatrix} \text{cat} & \text{v} \end{bmatrix} \\ \text{obj} & \text{nil} \end{bmatrix}$$

Subsumptie in Prolog

- Subsumptie kan gecontroleerd worden met `terms:subsumes_chk(?General,?Specific)`
- Dit predikaat is waar als *Specific* gesubsumeerd wordt door *General*.

```
| ?- use_module(library(terms)).
| ?- subsumes_chk(f(X), f(a)).
true
| ?- subsumes_chk(f(a), f(X)).
no
| ?- subsumes_chk(A-A, B-C).
no
| ?- subsumes_chk(A-B, C-C).
true
```

Unificatie

- De unificatie van AV_1 en AV_2 ($AV_1 \sqcup AV_2$) is de meest algemene attribute-value structure AV_3 die wordt gesubsumeerd door AV_1 en AV_2.

$$\begin{bmatrix} \text{agr} & \begin{bmatrix} \text{per} & 3 \end{bmatrix} \\ \text{case} & \text{nom} \end{bmatrix} \sqcup \begin{bmatrix} \text{agr} & \begin{bmatrix} \text{num} & \text{pl} \end{bmatrix} \end{bmatrix} =$$

$$\begin{bmatrix} \text{agr} & \begin{bmatrix} \text{num} & \text{pl} \\ \text{per} & 3 \end{bmatrix} \\ \text{case} & \text{nom} \end{bmatrix}$$

Unificatie (2)

$$\begin{bmatrix} \text{head} & \begin{bmatrix} \text{cat} & \text{v} \\ \text{sem} & \boxed{1} \end{bmatrix} \\ \text{sem} & \boxed{1} \end{bmatrix} \sqcup \begin{bmatrix} \text{sem} & \begin{bmatrix} \text{pred} & \text{lopen} \\ \text{arg1} & \text{jan} \end{bmatrix} \end{bmatrix} =$$

$$\begin{bmatrix} \text{head} & \begin{bmatrix} \text{cat} & \text{v} \\ \text{sem} & \boxed{1} \begin{bmatrix} \text{pred} & \text{lopen} \\ \text{arg1} & \text{jan} \end{bmatrix} \end{bmatrix} \\ \text{sem} & \boxed{1} \end{bmatrix}$$

Unificatie (3)

- Als de attribute-value structure AV_3, die wordt gesubsumeerd door AV_1 en AV_2 niet bestaat, is unificatie niet mogelijk.

$$\begin{bmatrix} & \begin{bmatrix} \text{per} & 3 \\ \text{num} & \text{sg} \end{bmatrix} \\ \text{agr} & \\ \text{case} & \text{nom} \end{bmatrix} \quad \not\sqcup \quad \begin{bmatrix} \text{agr} & \begin{bmatrix} \text{num} & \text{pl} \end{bmatrix} \end{bmatrix}$$

Unificatie in Prolog

- Zie Logisch Programmeren...

```
lists:member(M,X), lists:member(M,Y).
A = B.
[...]
```

HDRUG

Attribute-value structures in Prolog

$$\begin{bmatrix} \text{cat} & \text{np} \\ \text{agr} & \begin{bmatrix} \text{num} & \text{pl} \\ \text{per} & 1 \end{bmatrix} \end{bmatrix}$$

```
np(agr(pl,1))
```

$$\begin{bmatrix} \text{cat} & \text{np} \\ \text{agr} & \begin{bmatrix} \text{per} & 1 \end{bmatrix} \end{bmatrix}$$

```
np(agr(_,1))
```

Implementatie in Hdrug

- Declaratie van attributen
- Notatie voor attribute-value structures
- Notatie voor grammaticaregels en woordenboeken

Definities

- Iedere attribute-value structure is van een bepaald *type*.
- Het type bepaalt welke *attributen* gedefinieerd zijn.

Hdrug voorbeeld

```
% types
top([sign,head,agr]).

% type(Naam,SubTypes,Attributen)
type(agr,[],[per,num,det]).
type(head,[],[agr,case,vform]).
type(sign,[],[cat,head,comp,ds]).
```

Hdrug en attribute-value structures

- Een pad is een Prolog term gevolgd door attributen, gescheiden door een dubbele punt (:):
  - `X:cat`
  - `X:agr:num`
- Het <=>/2 predikaat stelt twee paden gelijk aan elkaar:
  - `VP:sem:subj <=> NP:sem`
  - `NP:cat <=> np`

Notatie van grammatica-regels

- We stappen van DCG-regels af.
- Gewone Prolog regels en feiten kunnen we eenvoudiger gebruiken met een aparte parser.
- Regels met als hoofd:

```
regel(Id,Moeder,Dochters)
```

NP --> Det N

```
%% regel(Id,Moeder,Dochters)
regel(np_det_n,NP,[Det,N]) :-
    NP:cat <=> np,
    Det:cat <=> det,
    N:cat <=> n,
    N:head <=> NP:head,
    Det:head:agr <=> N:head:agr.
```

$$\begin{bmatrix} \text{cat} & \text{np} \\ \text{head} & \boxed{1} \end{bmatrix} \rightarrow \begin{bmatrix} \text{cat} & \text{det} \\ \text{head} & \begin{bmatrix} \text{agr} & \boxed{2} \end{bmatrix} \end{bmatrix} \begin{bmatrix} \text{cat} & \text{n} \\ \text{head} & \boxed{1} \begin{bmatrix} \text{agr} & \boxed{2} \end{bmatrix} \end{bmatrix}$$

VP --> V

```
regel(vp_v,VP,[V]) :-
    VP:cat <=> vp,
    V:cat <=> v,
    VP:head <=> V:head,
    V:comp  <=> nil.
```

$$\begin{bmatrix} \text{cat} & \text{vp} \\ \text{head} & \boxed{1} \end{bmatrix} \rightarrow \begin{bmatrix} \text{cat} & \text{v} \\ \text{comp} & \text{nil} \\ \text{head} & \boxed{1} \end{bmatrix}$$

S --> NP VP

```
regel(s_np_vp,S,[NP,VP]) :-
    S:cat <=> s,
    NP:cat <=> np,
    VP:cat <=> vp,
    NP:head:case <=> nom,
    S:head <=> VP:head,
    NP:head:agr <=> VP:head:agr.
```

$$\begin{bmatrix} \text{cat} & \text{s} \\ \text{head} & \boxed{1} \end{bmatrix} \rightarrow \begin{bmatrix} \text{cat} & \text{np} \\ \text{head} & \begin{bmatrix} \text{agr} & \boxed{2} \\ \text{case} & \text{nom} \end{bmatrix} \end{bmatrix} \begin{bmatrix} \text{cat} & \text{vp} \\ \text{head} & \boxed{1} \begin{bmatrix} \text{agr} & \boxed{2} \end{bmatrix} \end{bmatrix}$$

VP --> V NP

```
regel(vp_v_np,VP,[V,NP]) :-
    VP:cat <=> vp,
    V:cat <=> v,
    NP:cat <=> np,
    NP:head:case <=> acc,
    VP:head <=> V:head,
    V:comp <=> np.
```

$$\begin{bmatrix} \text{cat} & \text{vp} \\ \text{head} & \boxed{1} \end{bmatrix} \rightarrow \begin{bmatrix} \text{cat} & \text{v} \\ \text{comp} & \text{np} \\ \text{head} & \boxed{1} \end{bmatrix} \begin{bmatrix} \text{cat} & \text{np} \\ \text{head} & \begin{bmatrix} \text{case} & \text{acc} \end{bmatrix} \end{bmatrix}$$

N --> duif

```
woord(duif, N) :-
    N:cat <=> n,
    N:head:agr:per <=> 3,
    N:head:agr:num <=> sg,
    N:head:agr:det <=> de.
```

$$\begin{bmatrix} \text{cat} & \text{n} \\ \text{head} & \begin{bmatrix} \text{agr} & \begin{bmatrix} \text{per} & 3 \\ \text{num} & \text{sg} \\ \text{det} & \text{de} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

NP --> hij

```
woord(hij, NP) :-
    NP:cat <=> np,
    NP:head:agr:per <=> 3,
    NP:head:agr:num <=> sg,
    NP:head:case <=> nom.
```

$$\begin{bmatrix} \text{cat} & \text{np} \\ \text{head} & \begin{bmatrix} \text{agr} & \begin{bmatrix} \text{per} & 3 \\ \text{num} & \text{sg} \end{bmatrix} \\ \text{case} & \text{nom} \end{bmatrix} \end{bmatrix}$$

NP --> hem

```
woord(hem,NP) :-
    NP:cat <=> np,
    NP:head:agr:per <=> 3,
    NP:head:agr:num <=> sg,
    NP:head:case <=> acc.
```

$$\begin{bmatrix} \text{cat} & \text{np} \\ \text{head} & \begin{bmatrix} \text{agr} & \begin{bmatrix} \text{per} & 3 \\ \text{num} & \text{sg} \end{bmatrix} \\ \text{case} & \text{acc} \end{bmatrix} \end{bmatrix}$$

V --> lacht

```
woord(lacht, V) :-
    V:cat <=> v,
    V:head:agr:per <=> 3,
    V:head:agr:num <=> sg,
    V:head:vform <=> fin,
    V:comp <=> nil.
```

$$\begin{bmatrix} \text{cat} & \text{v} \\ \text{head} & \begin{bmatrix} \text{agr} & \begin{bmatrix} \text{per} & 3 \\ \text{num} & \text{sg} \end{bmatrix} \\ \text{vform} & \text{fin} \end{bmatrix} \\ \text{comp} & \text{nil} \end{bmatrix}$$

V --> lachen

```
woord(lachen, V) :-
    V:cat <=> v,
    V:head:agr:per <=> 3,
    V:head:agr:num <=> pl,
    V:head:vform <=> fin,
    V:comp <=> nil.
```

$$\begin{bmatrix} \text{cat} & \text{v} \\ \text{head} & \begin{bmatrix} \text{agr} & \begin{bmatrix} \text{per} & 3 \\ \text{num} & \text{pl} \end{bmatrix} \\ \text{vform} & \text{fin} \end{bmatrix} \\ \text{comp} & \text{nil} \end{bmatrix}$$

Macros/templates

- Sommige regels bevatten identieke code.
- Sommige code/principes wil je een naam geven (leesbaarheid, begrijpelijkheid).
- In een regel kun je andere Prolog predikaten aanroepen.
- Argumenten zijn (delen van) attribute-value structures.

Macros/templates

```
regel(s_np_vp,S,[NP,VP]) :-
    S:cat <=> s,
    NP:cat <=> np,
    VP:cat <=> vp,
    NP:head:case <=> nom,
    S:head <=> VP:head,
    subject_verb_agreement(NP,VP).

subject_verb_agreement(NP,VP) :-
    NP:head:agr <=> VP:head:agr.
```

Hdrug demo