Natuurlijke-Taalverwerking 1

Week 3

Definite Clause Grammar (vervolg)

Overzicht

- DCG
- Hoofdzinnen en bijzinnen
- Betekenis
- Generatie
- Automatisch Vertalen
- Meer dan context-vrije grammatica
- Een toepassing

DCG

- Notatie voor grammatica regels
- Categorieën: Prolog termen
- Extra Prolog condities met `{ .. }`
- Automatisch omgezet naar Prolog regels

DCG

```
s --> np(N), vp(N).
det --> [].
n --> [eieren].
```

```
s(P0,P2) :- np(N,P0,P1), vp(N,P1,P2).
det(P0,P) :- P0 = P.
n(P0,P) :- 'C'(P0,eieren,P).
```

Prolog-lijsten als String-Posities

de olifant ontsnapt

```
0 de 1 olifant 2 ontsnapt 3
```

```
0 = [de,olifant,ontsnapt]
1 = [olifant,ontsnapt]
...
3 = [ ]
```

```
word([de,olifant,ontsnapt],de,[olifant,ontsnapt]).
word([olifant,ontsnapt],olifant,[ontsnapt]).
word([ontsnapt],ontsnapt,[]).
```

My first parser (mens)

```
parse(Words) :-
    s(Words,[]).

s --> np, vp.
np --> det, n.
vp --> v.
vp --> v, np.
det --> [de].
det --> [het].
det --> [].
n --> [man].
n --> [huis].
v --> [verlaat].
v --> [schreeuwt].
```

My first parser (computer)

```
parse(Words) :-
  s(Words,[]).

'C'([Word|Words],Word,Words).

%% ...  en verder de grammaticaregels:
s(P0,P2) :- np(P0,P1), vp(P1,P2).
np(P0,P3) :- det(P0,P1), n(P1,P2).
vp(P0,P1) :- v(P0,P1).
vp(P0,P2) :- v(P0,P1), np(P1,P2).
det(P0,P1) :- 'C'(P0,de,P1).
det(P0,P1) :- 'C'(P0,het,P1).
det(P0,P0).
n(P0,P1) :- 'C'(P0,man,P1).
n(P0,P1) :- 'C'(P0,huis,P1).
v(P0,P1) :- 'C'(P0,verlaat,P1).
v(P0,P1) :- 'C'(P0,schreeuwt,P1).
```

Hoofdzinnen en bijzinnen

(1) a. de directeur voert de flamingo
b. omdat de directeur de flamingo voert

(2) a. de ijsbeer eet gulzig
b. omdat de ijsbeer gulzig eet

(3) a. de directeur voert de flamingo regelmatig
b. omdat de directeur de flamingo regelmatig voert

(4) a. de ijsbeer belt de flamingo soms op
b. omdat de ijsbeer de flamingo soms op belt

(5) a. de ijsbeer zou de flamingo willen eten
b. omdat de ijsbeer de flamingo zou willen eten

Hoofdzinnen en bijzinnen

Een hoofdzin lijkt op een bijzin, afgezien van de positie van de persoonsvorm

Hoofdzinnen en bijzinnen

```
%% voert de flamingo
vp --> v np
%% eet gulzig
vp --> v adv

%% omdat de directeur de flamingo voert
cp --> [omdat] np vpb

%% de flamingo voert
vpb --> np v

%% gulzig eet
vpb --> adv v
```

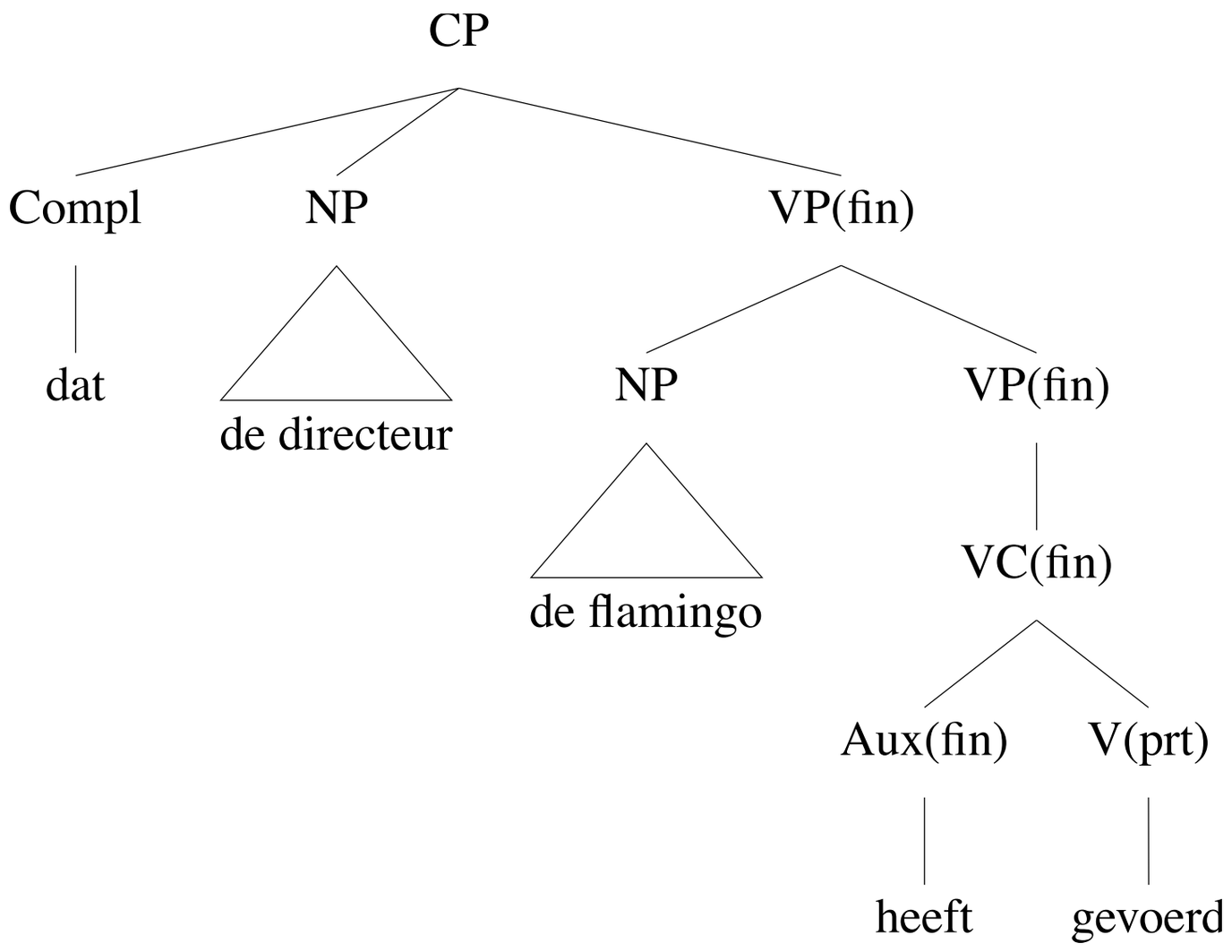
Hoofdzinnen en bijzinnen

(6) a. de directeur heeft de flamingo gevoerd
b. dat de directeur de flamingo heeft gevoerd
c. dat de directeur de flamingo gevoerd heeft

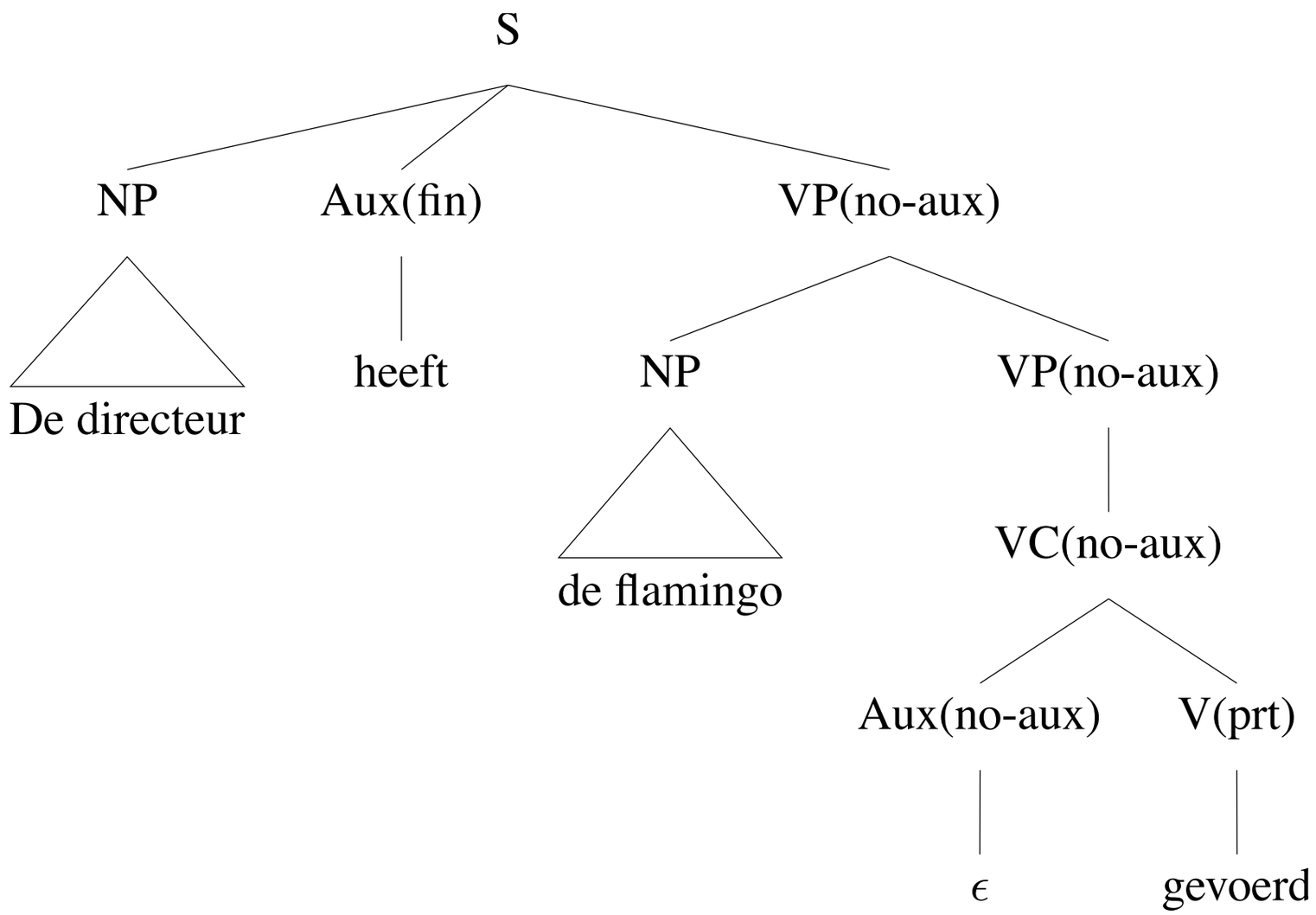
Hoofdzinnen en bijzinnen

```
%% heeft de flamingo gevoerd
vp --> aux np prt

%% de flamingo gevoerd heeft
%% de flamingo heeft gevoerd
vpb --> np vc

%% heeft gevoerd
vc  --> aux prt
%% gevoerd heeft
vc  --> prt aux

%% voert
vc  --> v
```

Bijzinnen


CP
Compl
NP
VP(fin)
dat
de directeur
NP
VP(fin)
de flamingo
VC(fin)
Aux(fin)
V(prt)
heeft
gevoerd

Hoofdzinnen



DCG

Argument om informatie over de persoonsvorm door te geven:

```
s --> np, v(fin),   vp(no_v).
s --> np, aux(fin), vp(no_aux).
cp --> compl, np, vp(fin).

vp(V) --> np, vp(V).
vp(V) --> adv, vp(V).
vp(V) --> vc(V).

vc(V) --> v(V).
vc(V) --> aux(V), v(prt).
vc(V) --> v(prt), aux(V).

v(fin) --> [voert].
v(no_v) --> [].

aux(fin) --> [heeft].
aux(no_aux) --> [].
```

DCG . . .

- omdat de directeur de flamingo's wil voeren
- omdat de directeur de flamingo's wil hebben gevoerd
- omdat de directeur de flamingo's gevoerd wil hebben
- omdat de directeur de flamingo's zou willen hebben gevoerd
- omdat de directeur de flamingo's gevoerd zou willen hebben

Automatisch vertalen

Iedere student bezit een fiets

$$\Downarrow$$

$$\forall x(student(x) \rightarrow \exists y(bike(y) \wedge possess(x, y)))$$

$$\Downarrow$$

Every student owns a bike

- Bepaal de betekenis van de zin in de brontaal,
- Genereer een zin in de doeltaal met dezelfde betekenis.

Semantiek (Betekenis)

- Betekenis van een zin is opgebouwd uit de betekenis van de zinsdelen,
- De betekenis van zinsdelen is opgebouwd uit de betekenis van de samenstellende delen (zinsdelen en/of woorden).
- Aan een DCG kan een argument worden toegevoegd dat de betekenis weergeeft.

Semantiek en DCG

```
tijd(uur(U,M)) --> uur(U), [uur], min(M).
tijd(uur(U,M)) --> min(M), [over], uur(U).
tijd(uur(U,M)) --> min(M), [minuten,over],
                       uur(U).

min(15) --> [vijftien].
min(15) --> [kwart].
uur(3) --> [drie].

?- tijd(Uur,[kwart,over,drie],[]).

Uur = uur(3,15)
```

Semantiek en DCG

```
?- s(Sem,[ismail,spreekt,bahasa],[]).

Sem = spreken(i,b)

s(Sem) --> np(Sub), vp(Sub,Sem).
vp(Subj,Sem) --> v(Sub,Obj,Sem), np(Obj).
np(i) --> [ismail].
np(b) --> [bahasa].
v(Subj,Obj,spreken(Subj,Obj)) --> [spreekt].
```

Generatie

- Parsing is het omzetten van zinnen in boomstructuren of betekenis,
- Generatie is het omzetten van betekenis in zinnen.

Generatie en DCG

- Een DCG wordt een generator, wanneer de lijst woorden geheel of gedeeltelijk variabel is.

```
?- s([A,B,C,D],[]).
```

Generatie van willekeurige zinnen

```
?-  s([A,B,C,D],[]), write([A,B,C,D]), nl, fail.
...
[de,partijbons,nadert,vandaag]
...
[een,neoliberale,boerendochter,ontsnapt]
...
[een,feestvarken,heeft,gesnurkt]
```

Generatie op basis van betekenis

```
tijd(uur(U,M)) --> uur(U), [uur], min(M).
tijd(uur(U,M)) --> min(M), [over], uur(U).

min(15) --> [vijftien].
min(15) --> [kwart].
uur(3) --> [drie].
```

```
?- tijd(uur(3,15),Words,[]).

Words = [drie,uur,vijftien] ;
Words = [vijftien,over,drie] ;
Words = [kwart,over,drie] ;
....
```

Automatisch vertalen

```
vertaal(NL,ENG) :-
   tijd(Semantiek,NL,[]),
   time(Semantiek,ENG,[]).

time(uur(H,M)) --> hour(H), minute(M).
time(uur(H,M)) --> minute(M), [past], hour(H).
```

Hoe krachtig is DCG?

- Krachtiger dan CFG!
- Bewijs:
  - ⋆ Er zijn talen waarvoor geen CFG te geven valt,
  - ⋆ Maar waarvoor wel een DCG bestaat.

COPY-taal

- WW: een willekeurige reeks a's en b's, gevolgd door dezelfde reeks.

```
s(L) --> w(L), w(L)
w([a|Rst]) --> [a], w(Rst).
w([b|Rst]) --> [b], w(Rst).
w([]) --> [].
```

```
?- s(L,[a,a,b,a,a,b],[]).
L = [a,a,b]
```

Nederlandse werkwoordsclusters

dat ik de olifanten voer
dat ik Henk de olifanten help voeren
dat ik Cecilia Henk de olifanten zie helpen voeren
....

- NP $n+1$ hoort bij werkwoord n,
- Dit patroon van kruisende afhankelijkheden vertoont gelijkenis met COPY-taal.
- Wordt gebruikt in bewijs dat natuurlijke taal niet context-vrij is (Huybregts 1984), met data uit het Züritüütsch.