Natuurlijke Taalverwerking I
Unificatie-Grammatica's II

Week 4

Komende weken

- Vandaag: Dependency structures, informatie-extractie
- Week 5: Ontleedalgoritmes
- Week 6: Het kiezen van de beste parse
- Week 7: Generatie

Overzicht

- Recapitulatie (unificatiegrammatica's)
- Representatie van syntax en semantiek
- Het bouwen van dependency structures
- Informatie-extractie
- Alpino
- Demo

UNIFICATIEGRAMMATICA’S

Recapitulatie

- Unificatiegrammatica's:
  - Syntactische informatie wordt gerepresenteerd met attributen/waarden.
  - Attributen worden verpakt in een attribute-value structure.
  - Waarden kunnen ook zelf weer een attribute-value structure zijn.
  - Agreement en overdragen informatie via unificatie.
- We kunnen attribute-value structures als Prolog termen representeren.
- Prolog biedt unificatie.
- Hulpmiddelen om attribute-value structures te maken en analyseren (o.a. Hdrug).

Types

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

% type(Naam,SubTypes,Attributen)
type(head,[],[agr,case,vform]).
type(agr,[],[per,num,det]).
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`
- **Expliciet toewijzen van een type:**
  - `X => sign`
  - `Y => head`

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

```
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.
```

Overerving

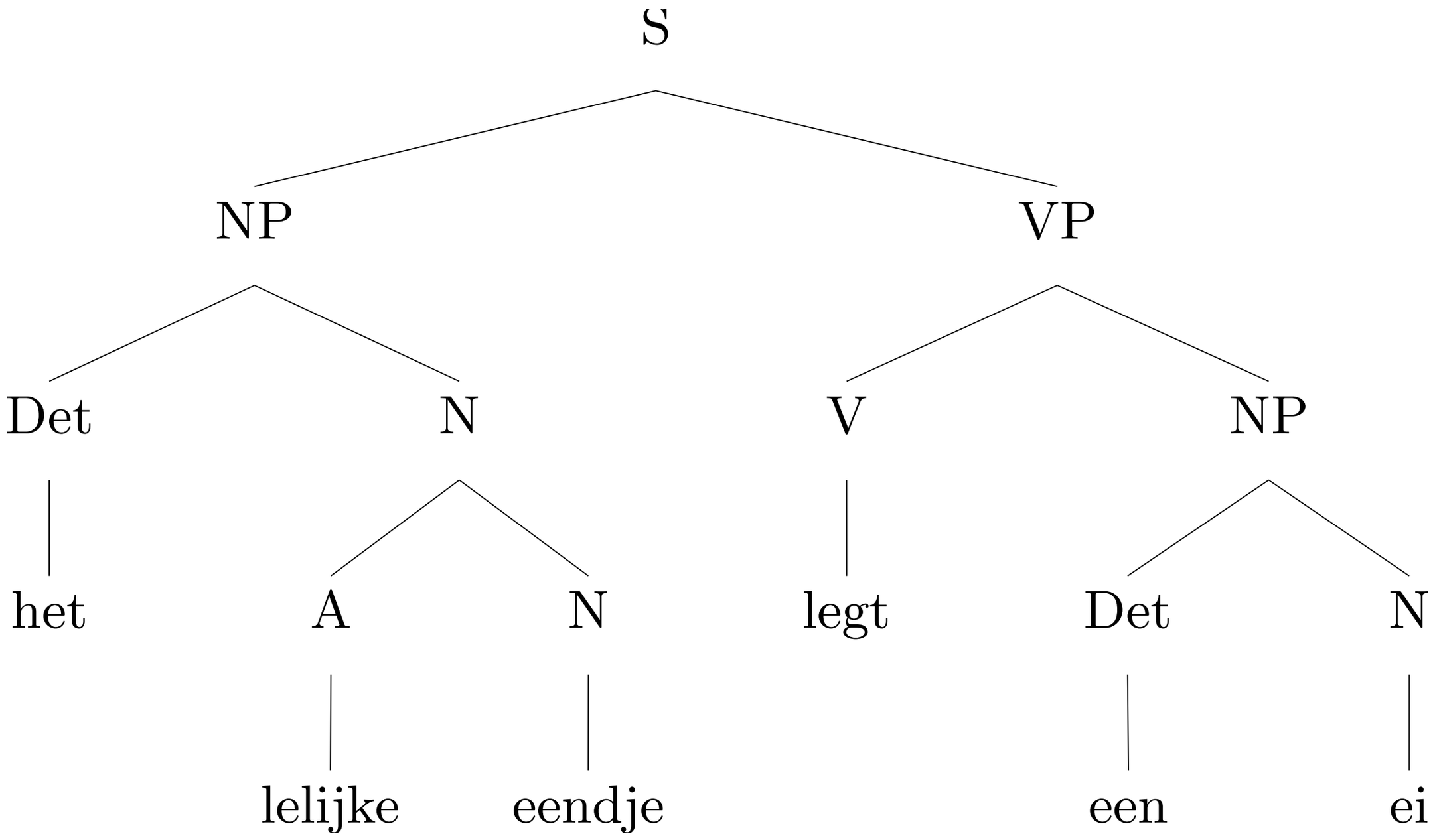
```
top([sign,head,agr]).

type(sign,[],[cat,head,comp,ds,sem]).
type(head,[],[agr,case,vform]).
type(agr,[],[per,num,det]).

type(sem,[nsem,psem,vsem],[pred,mod]).
type(nsem,[],[det]).
type(psem,[],[pobj]).
type(vsem,[],[subj,obj]).
```

REPRESENTATIES VOOR STRUCTUUR

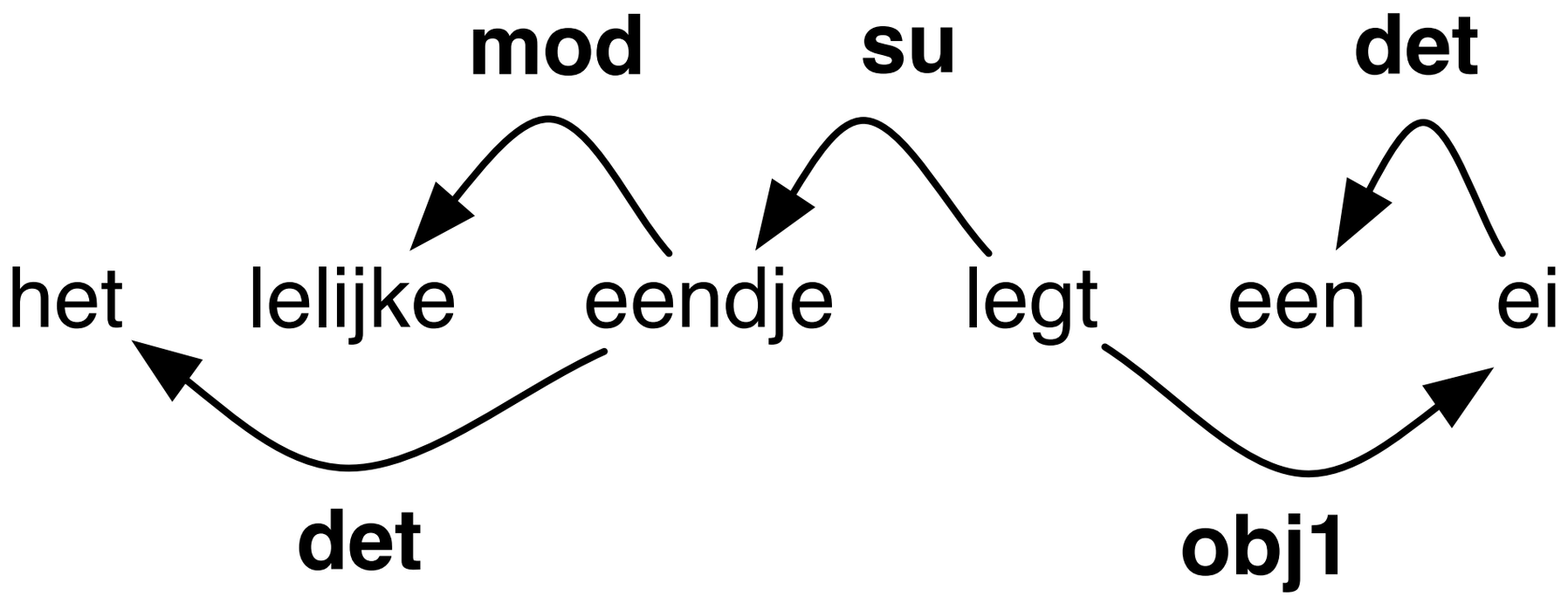
Doel van ontleden

- Bepalen van de syntactische structuur van een zin.
- Met welk doel, en welke informatie hebben we nodig om dit doel te bereiken?
- Verschillende analyses/representaties:
  - Phrase structure
  - Afhankelijkheidsrelaties tussen woorden (dependency relations)
  - Predikaten
- Verschillende opbouwmethoden:
  - Rechtstreeks
  - Bijeffect
  - Achteraf

Phrase structure trees



Dependency graph



Predikaten

eend(x), lelijk(x), dim(x), legt(x,y), ei(y)

DEPENDENCY STRUCTURES

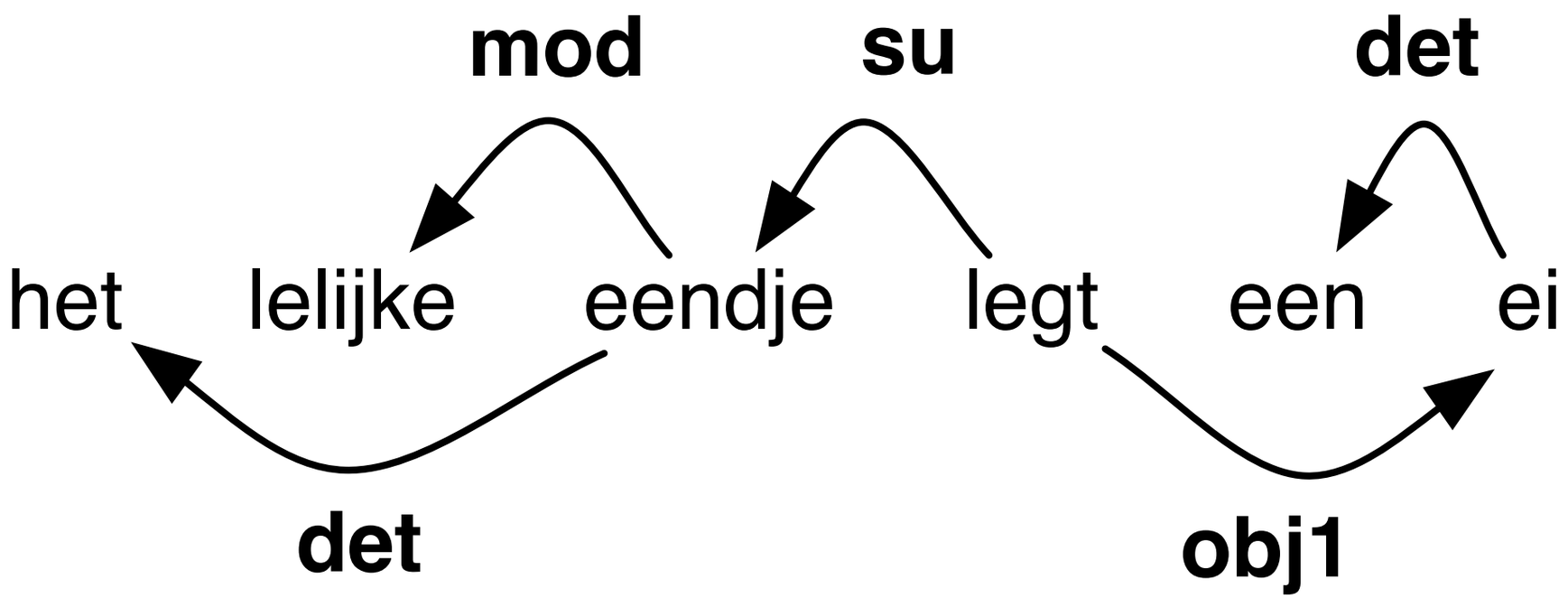
Dependentie relaties

- Dependentie relaties geven de grammaticale relaties tussen woorden weer.
- Lexicaal-gedreven: elk woord heeft een dependentie relatie met een ander woord.
- In een relatie is één woord het hoofd, het andere woord de dependant.
- Geeft veel kennis over de syntactische structuur van een zin.
- Krachtig genoeg om interessante informatie te extraheren uit een zin.

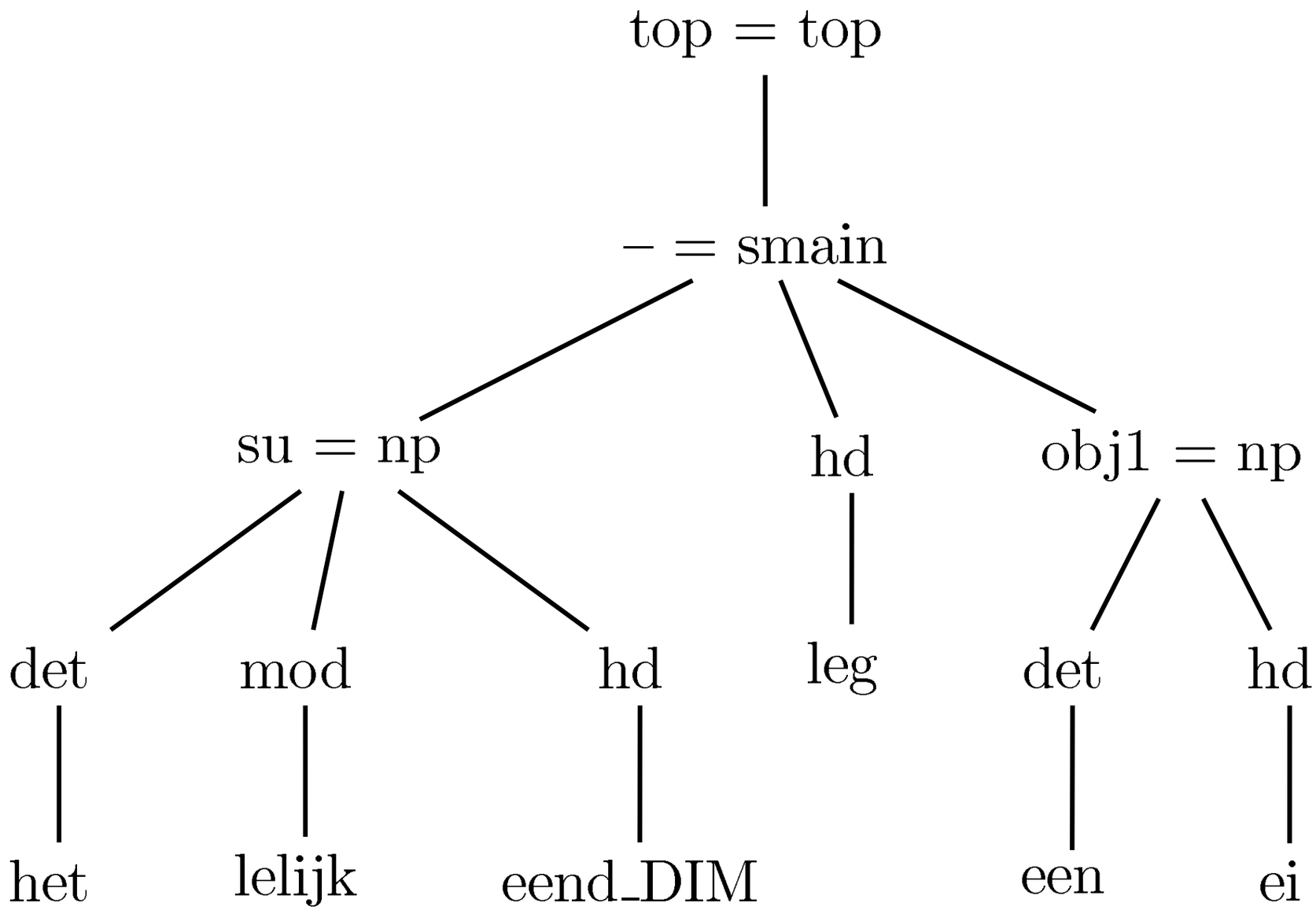
Toepassingen

- Informatie-extractie
- Machinevertaling
- Combineren van zinnen
- Vraag-antwoord systemen

Dependency graph



Dependency tree



Dependency triples

```
verb:leg/[3,4]|hd/su|noun:eend_DIM/[2,3]
noun:eend_DIM/[2,3]|hd/det|det:het/[0,1]
noun:eend_DIM/[2,3]|hd/mod|adj:lelijk/[1,2]
verb:leg/[3,4]|hd/obj1|noun:ei/[5,6]
noun:ei/[5,6]|hd/det|det:een/[4,5]
```

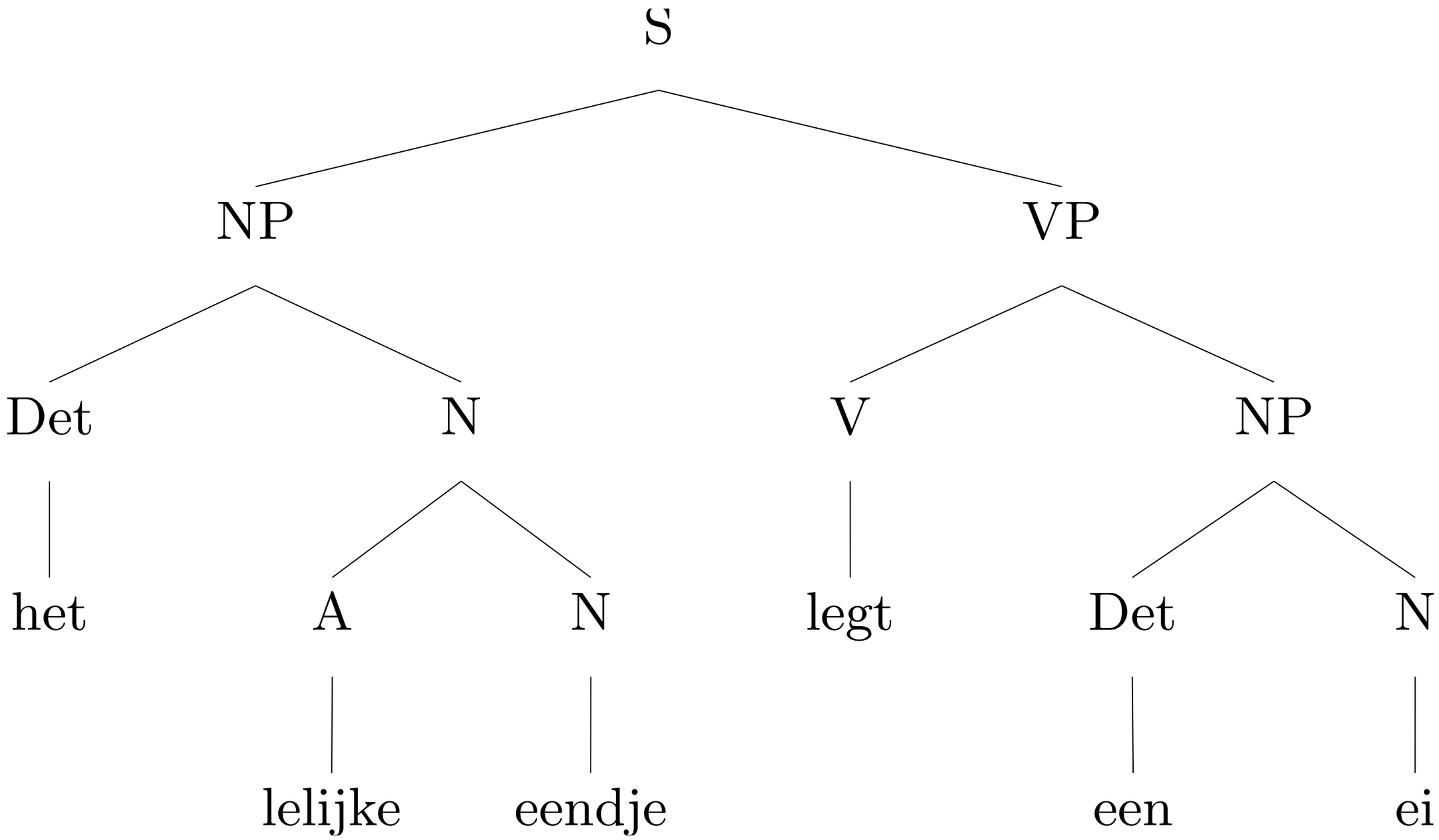
Dependentie structuur

$$
\begin{bmatrix}
\text{HWRD} & leg \\
\text{FRAME} & verb \\
\text{OBJ1} & \underset{dt}{\begin{bmatrix}
\text{HWRD} & ei \\
\text{FRAME} & noun \\
\text{DET} & \underset{dt}{\langle \begin{bmatrix} \text{HWRD} & een \\ \text{FRAME} & det \end{bmatrix} \rangle}
\end{bmatrix}} \\
\text{SU} & \underset{dt}{\begin{bmatrix}
\text{HWRD} & eend\_DIM \\
\text{FRAME} & noun \\
\text{DET} & \underset{dt}{\langle \begin{bmatrix} \text{HWRD} & het \\ \text{FRAME} & det \end{bmatrix} \rangle} \\
\text{MOD} & \underset{dt}{\langle \begin{bmatrix} \text{HWRD} & lelijk \\ \text{FRAME} & adj \end{bmatrix} \rangle}
\end{bmatrix}}
\end{bmatrix}_{dt}
$$

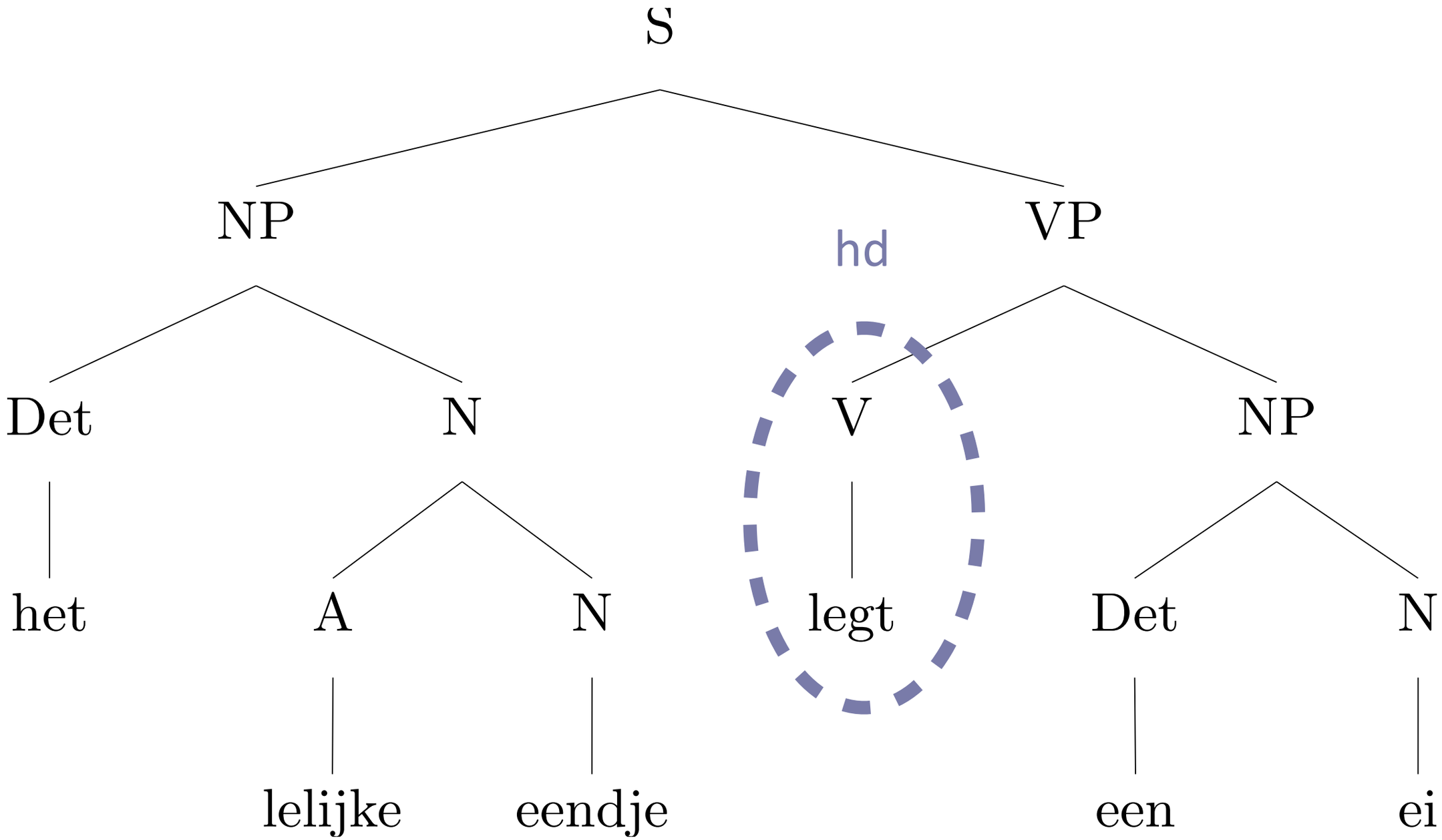
Bepalen dependencies

- Dependentiestructuren kunnen op verschillende manieren gebouwd worden:
  - Rechtstreeks
  - Als een bij-effect van een andere ontleedmethode
  - Achteraf (transformatie)
- Vandaag: bouwen van dependentiestructuren als bij-effect van ontleding met een unificatiegrammatica gebaseerd op phrase structure.

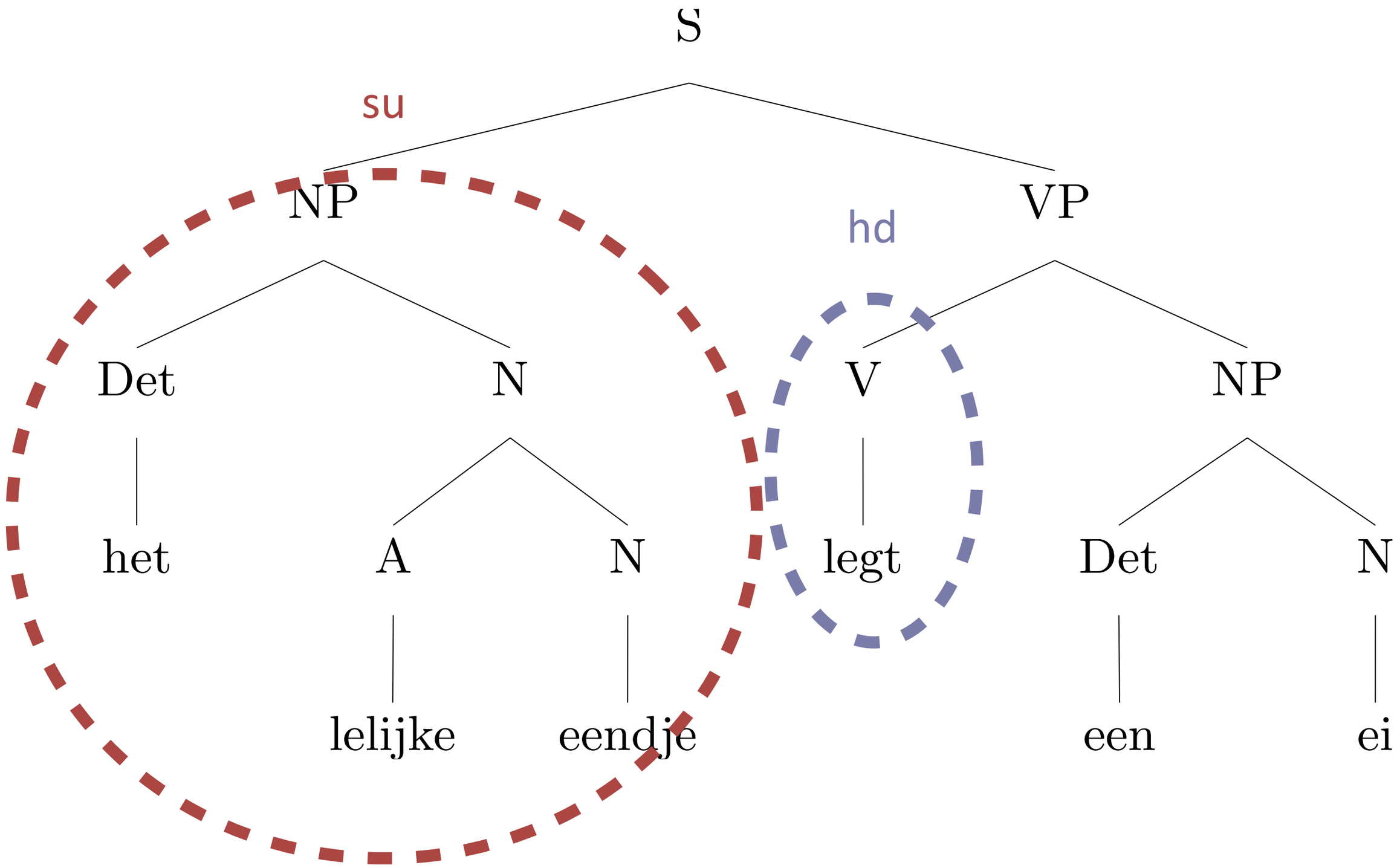
Bepalen dependencies



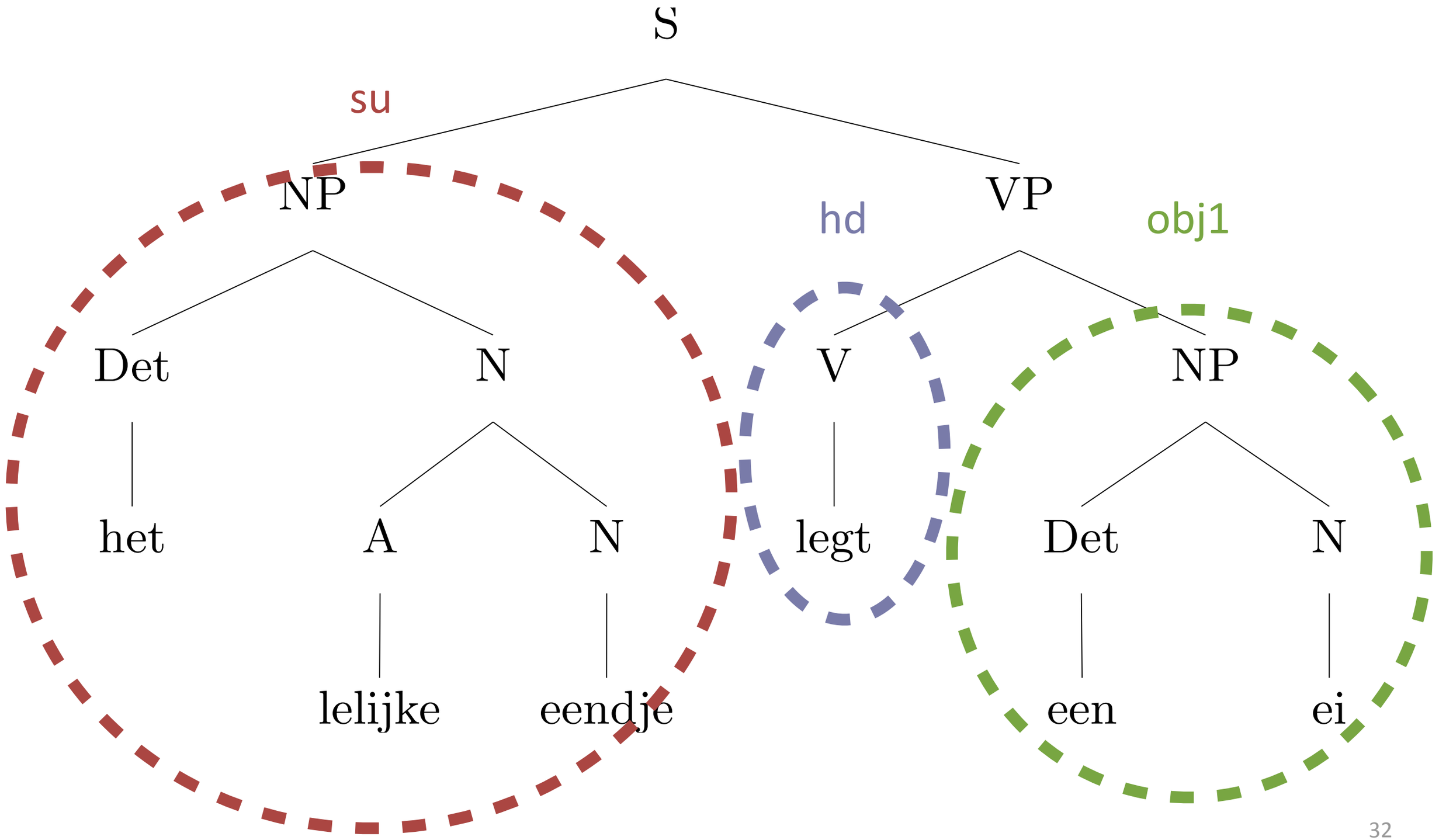
Bepalen dependencies



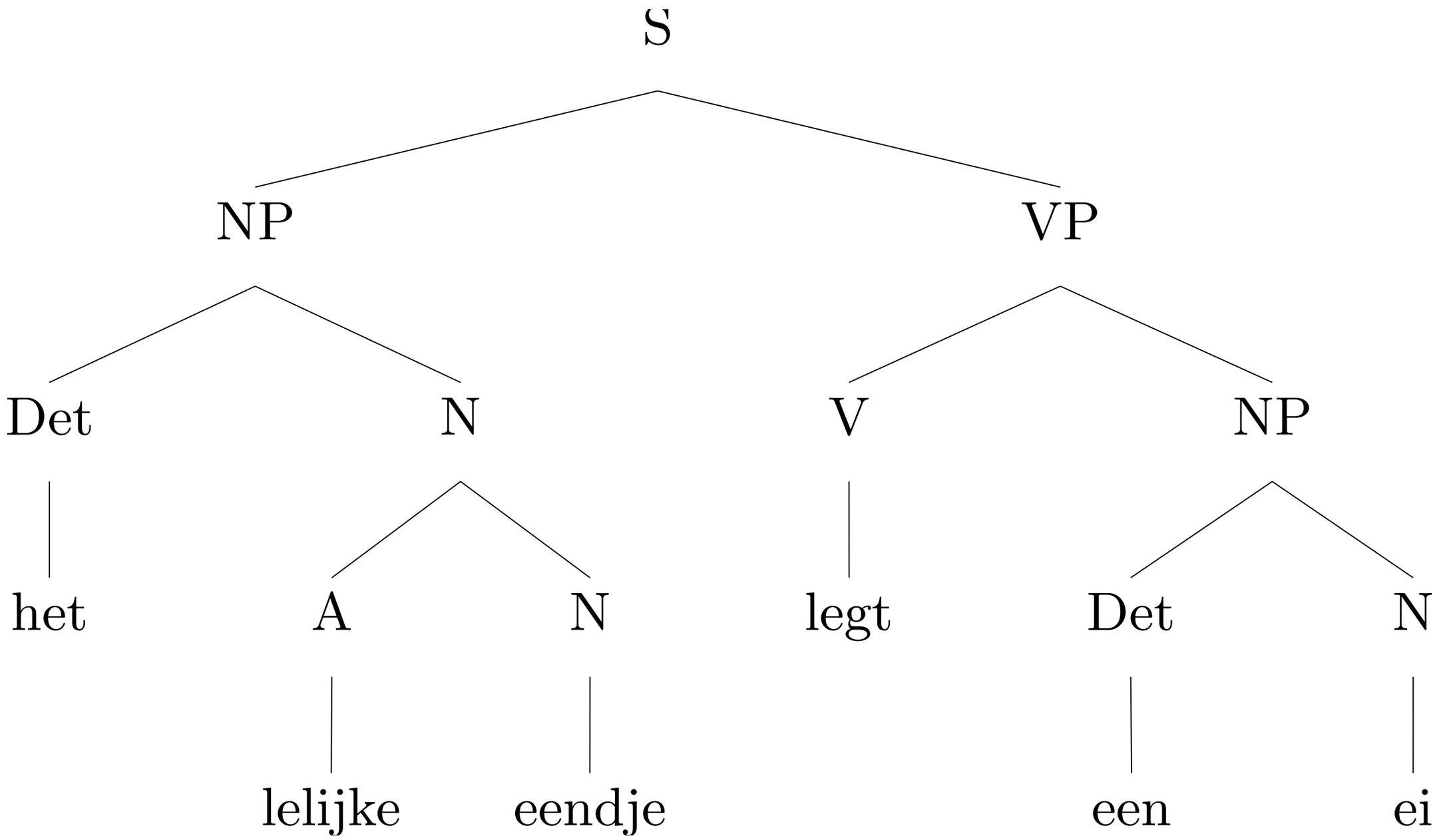
Bepalen dependencies



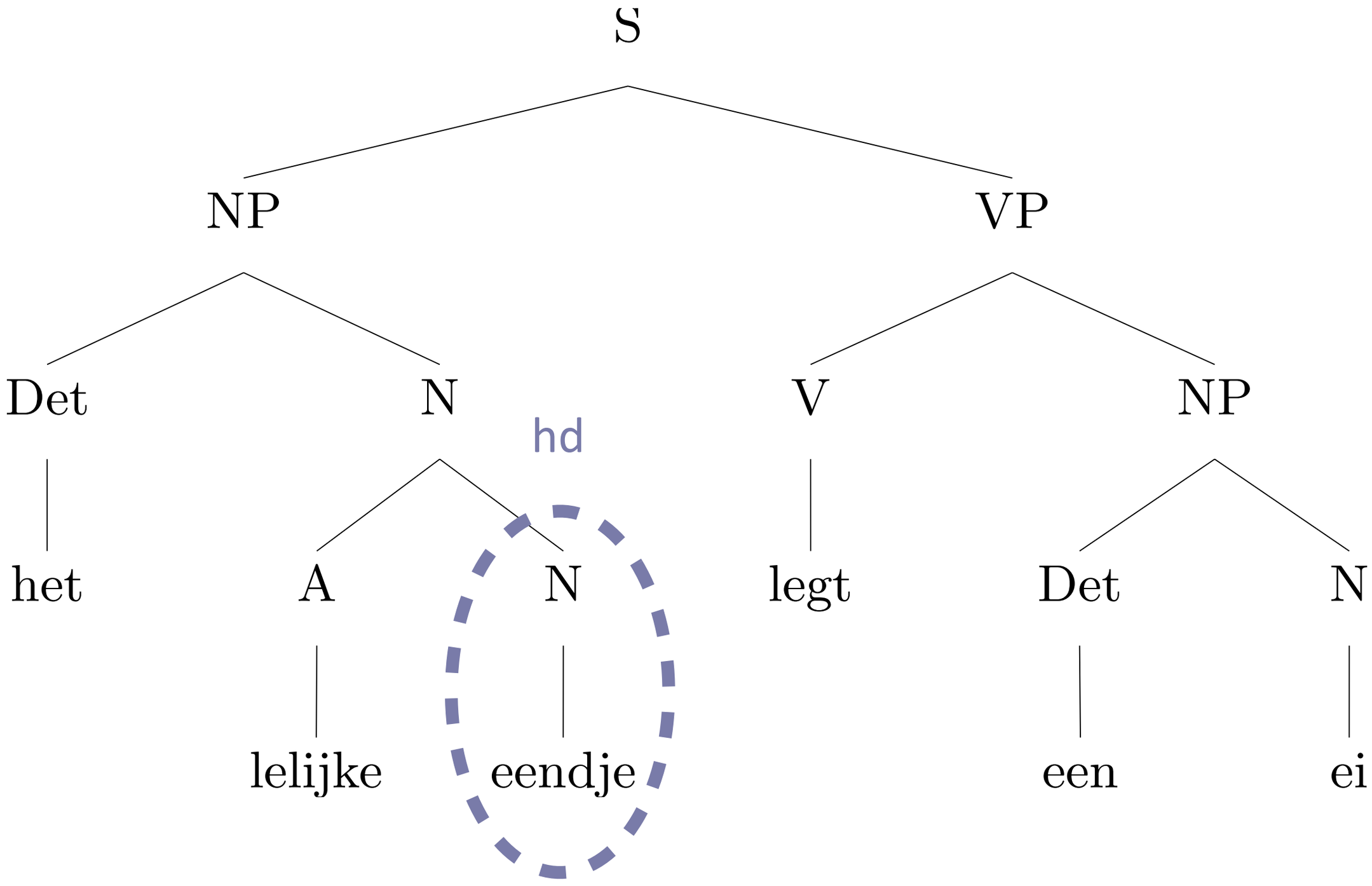
Bepalen dependencies



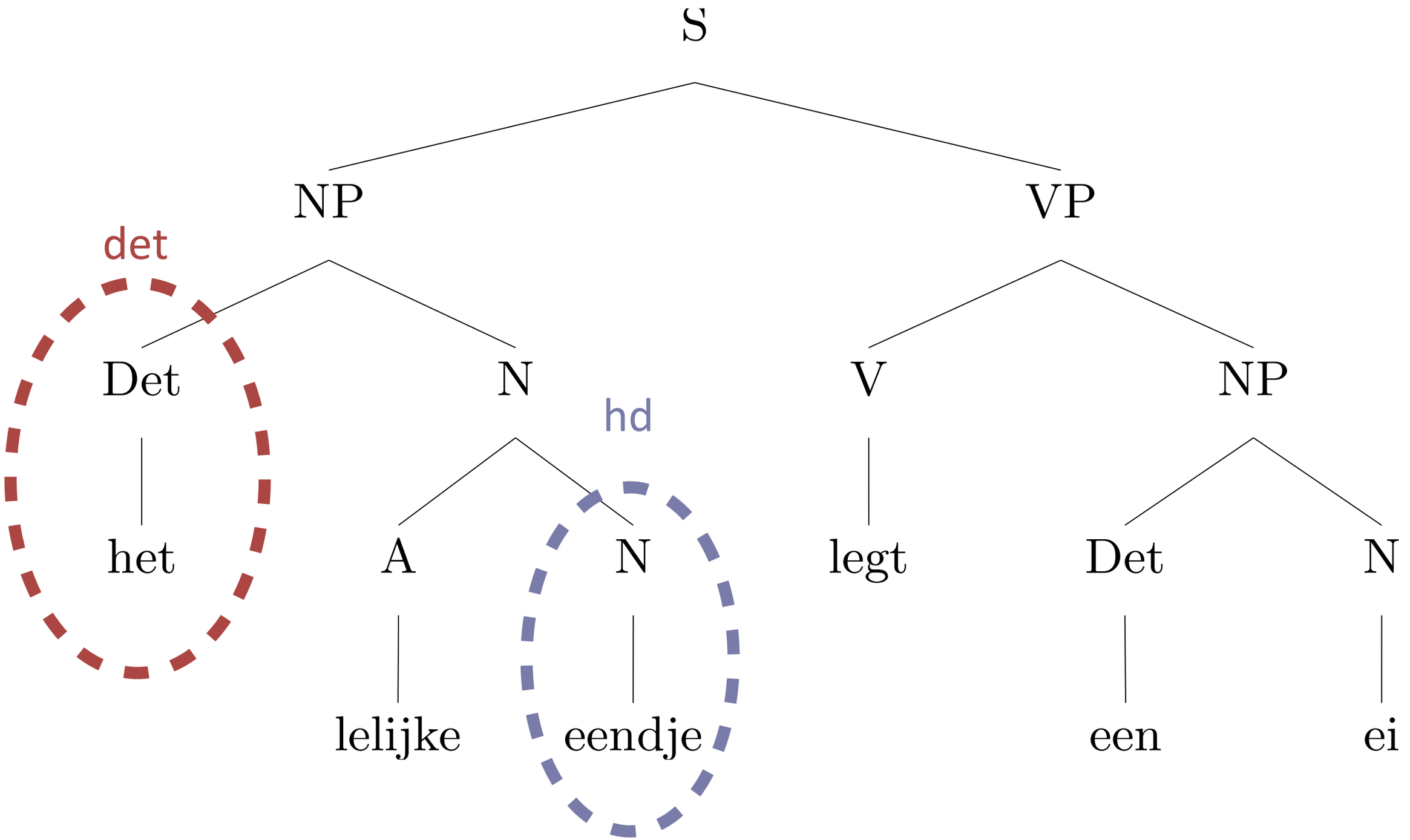
Bepalen dependencies (su)



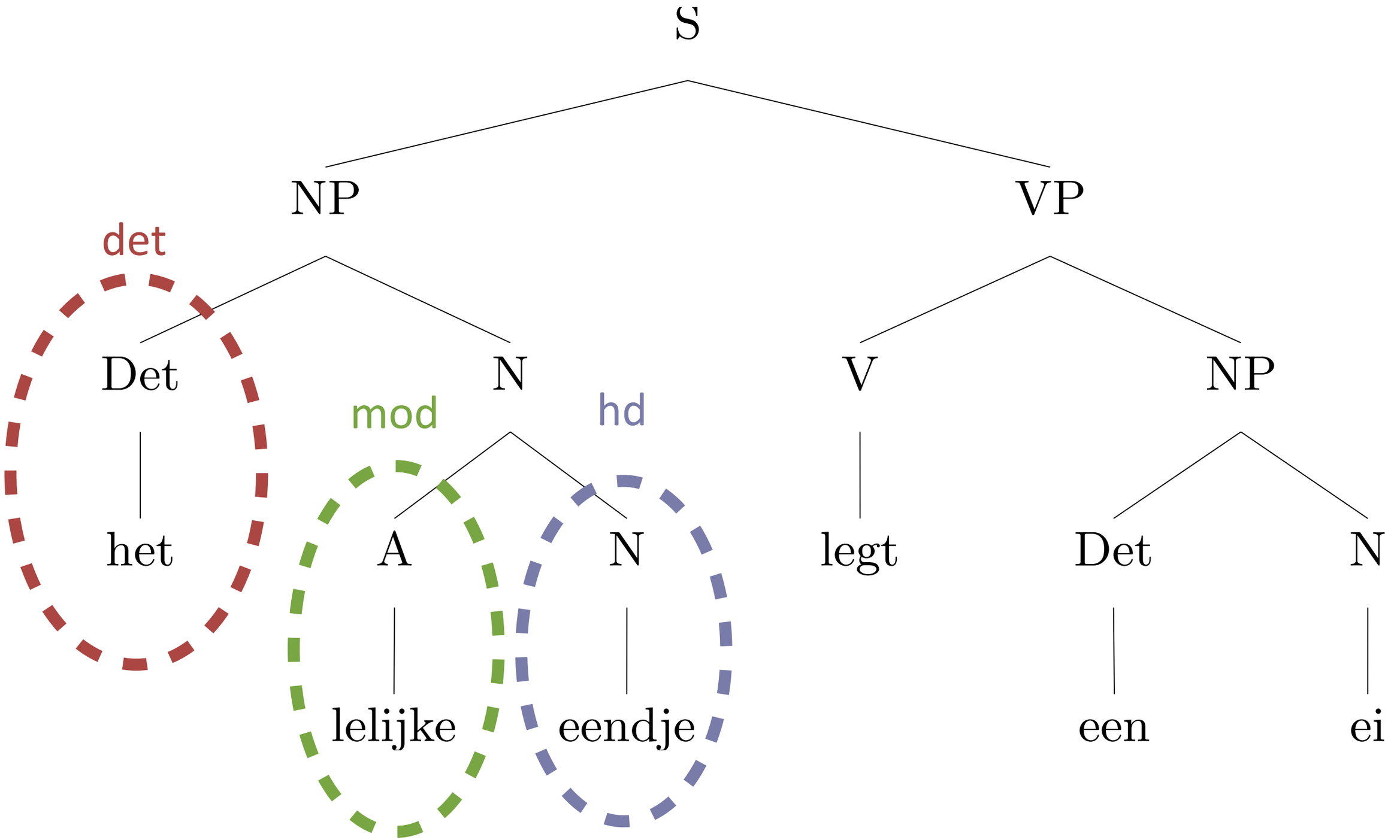
Bepalen dependencies (su)



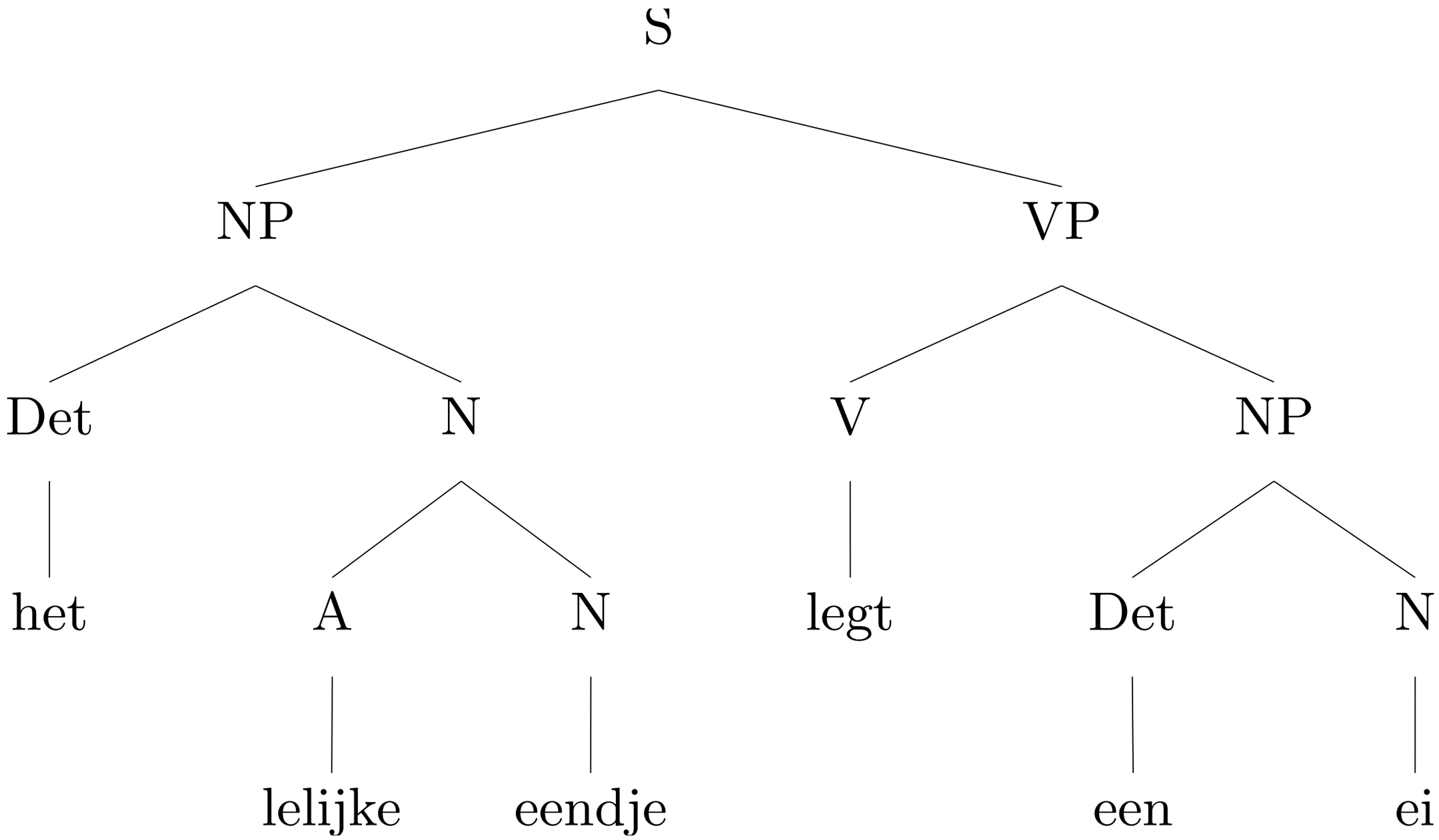
Bepalen dependencies (su)



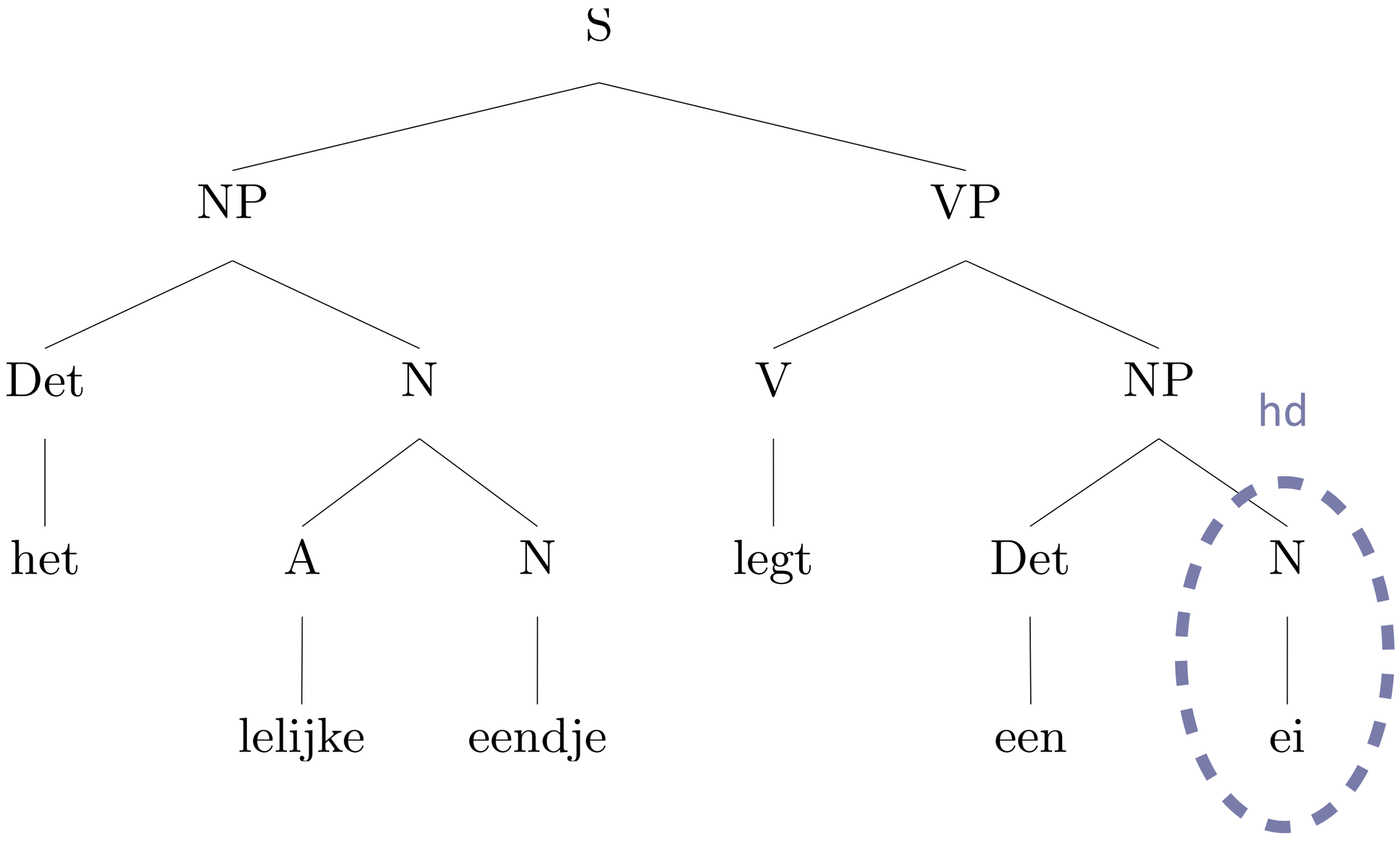
Bepalen dependencies (su)



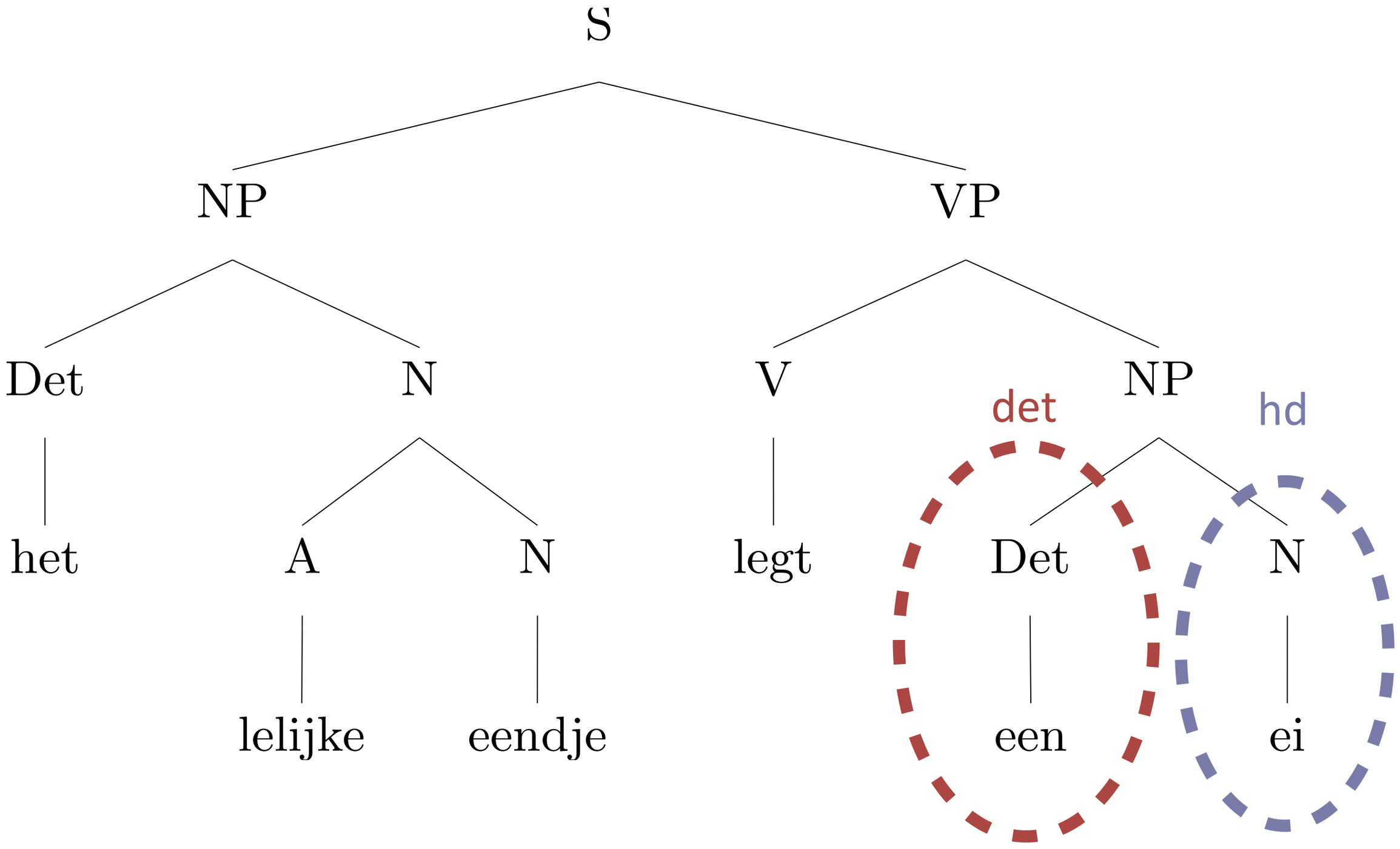
Bepalen dependencies (obj1)



Bepalen dependencies (obj1)



Bepalen dependencies (obj1)



Types voor dependentiestructuren

```
top([sign,head,agr,dt]).
type(sign,[],[cat,head,comp,cdet,cmod,dt,ds]).
[...]
type(dt,[],[hwrd,det,mod,su,obj1,obj2]).
```

Attribuut	Omschrijving
dt	Dependency structuur
hwrd	(Hoofd-)woord
det	Determiners
mod	Modifiers
su	Subject (onderwerp)
obj1	Direct object (bijv. lijdend voorwerp)
obj2	Indirect object (bijv. meew. voorwerp)

Dependencies voor S

$$\begin{bmatrix} \text{cat} & \text{s} \\ \text{dt} & \boxed{1}\begin{bmatrix} \text{su} & \boxed{2} \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \text{cat} & \text{np} \\ \text{dt} & \boxed{2}[\ldots] \end{bmatrix} \begin{bmatrix} \text{cat} & \text{vp} \\ \text{dt} & \boxed{1}[\ldots] \end{bmatrix}$$

```
regel(s_np_vp,S,[NP,VP]) :-
    S:cat <=> s,
    NP:cat <=> np,
    VP:cat <=> vp,
    ...
    S:dt <=> VP:dt,
    S:dt:su <=> NP:dt.
```

Dependencies voor NP

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

```
regel(np_det_n,NP,[Det,N]) :-
    NP:cat <=> np,
    Det:cat <=> det,
    N:cat <=> n,
    ...
    NP:dt <=> N:dt,
    NP:dt:det <=> Det:dt.
```

Dependencies voor NP

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

```
regel(np_det_n,NP,[Det,N]) :-
    NP:cat <=> np,
    Det:cat <=> det,
    N:cat <=> n,
    ...
    NP:dt <=> N:dt,
    NP:dt:det <=> Det:dt.
```

meerdere dets?

Dependencies voor VP

$$\begin{bmatrix} \text{cat} & \text{vp} & \\ \text{dt} & \boxed{1} & \begin{bmatrix} \text{obj1} & \boxed{2} \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \text{cat} & \text{v} & \\ \text{dt} & \boxed{1} & [\ldots] \end{bmatrix} \begin{bmatrix} \text{cat} & \text{np} & \\ \text{dt} & \boxed{2} & [\ldots] \end{bmatrix}$$

```
regel(vp_v_np,VP,[V,NP]) :-
    VP:cat <=> vp,
    V:cat <=> v,
    NP:cat <=> np,
    [...]
    VP:dt <=> V:dt,
    VP:dt:obj1 <=> NP:dt.
```

Lexical items

$$\text{lex\_dt}(\boxed{1},\boxed{2}) \text{ :- } \boxed{1}\left[\text{dt} \quad \left[\text{hwrd} \quad \boxed{2}\right]\right]$$

```
lex_dt(Fs,Head) :-
    Fs:dt <=> Dt,
    Dt => dt,
    Dt:hwrd <=> Head.

woord(duif,NP) :-
    N:cat <=> n,
    ...
    lex_dt(NP,duif).
```

Modifiers

- Adjectieven, bijwoorden, PPs, ...
- Als voorbeeld: PPs

PPs

```
regel(pp_p_np,PP,[P,NP]) :-
    PP:cat <=> pp,
    P:cat <=> p,
    NP:cat <=> np,
    [...]
    %% vul aan...
```

Modifiers

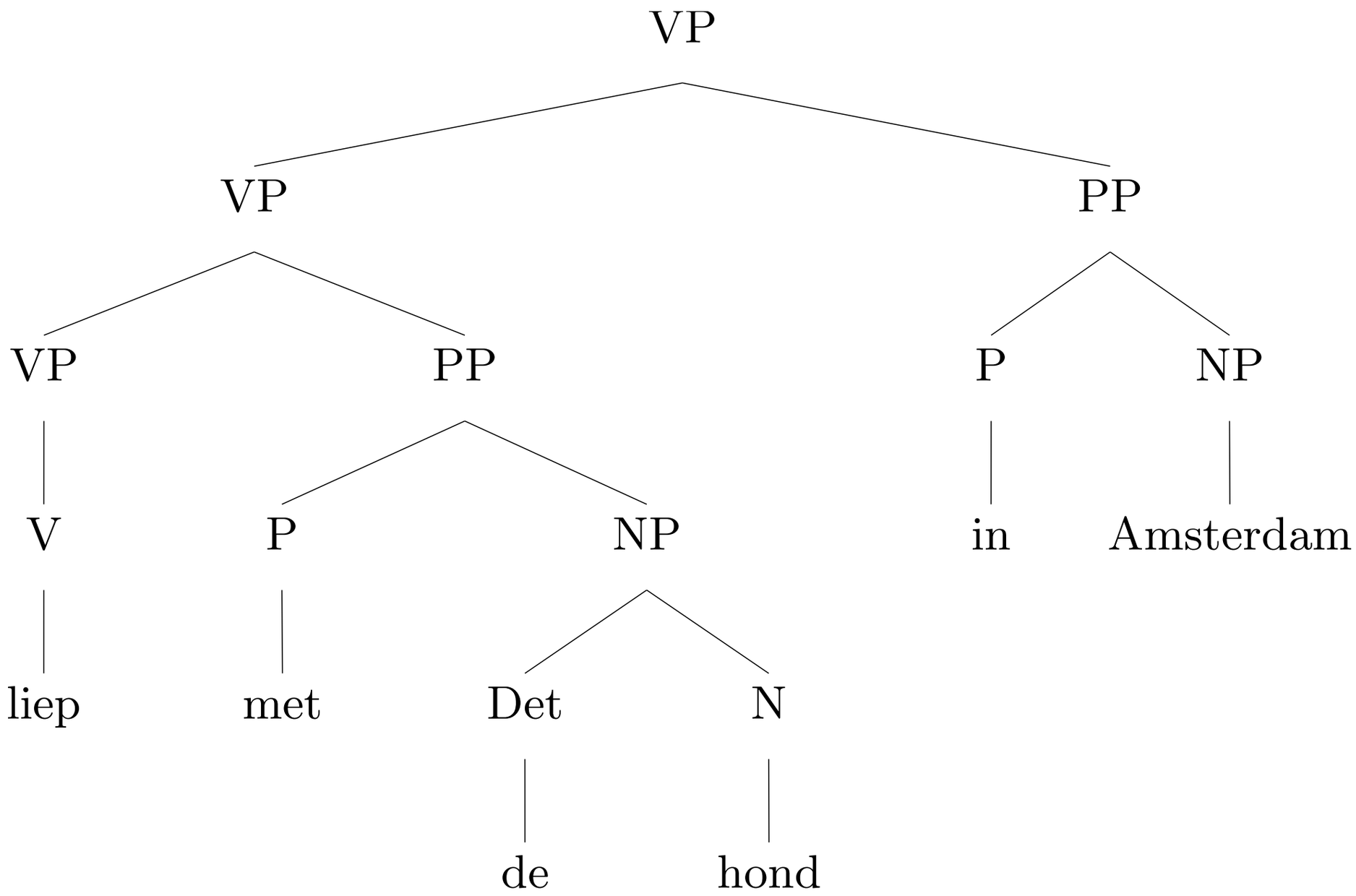
- Adjectieven, bijwoorden, PPs, ...
- Een hoofd kan meerdere modifiers hebben, dus representatie als een lijst.

$$\mathrm{VP} \quad \rightarrow \quad \mathrm{V}$$

$$\mathrm{VP} \quad \rightarrow \quad \mathrm{VP\ PP}$$

- *[[[liep] met de hond] in Amsterdam]*
- Het werkwoord is steeds het hoofd van de VP.
- Het hoofd krijgt steeds meer modifiers.
- Ergo: we moeten toestaan dat de lijst van modifiers uitgebreid kan worden.

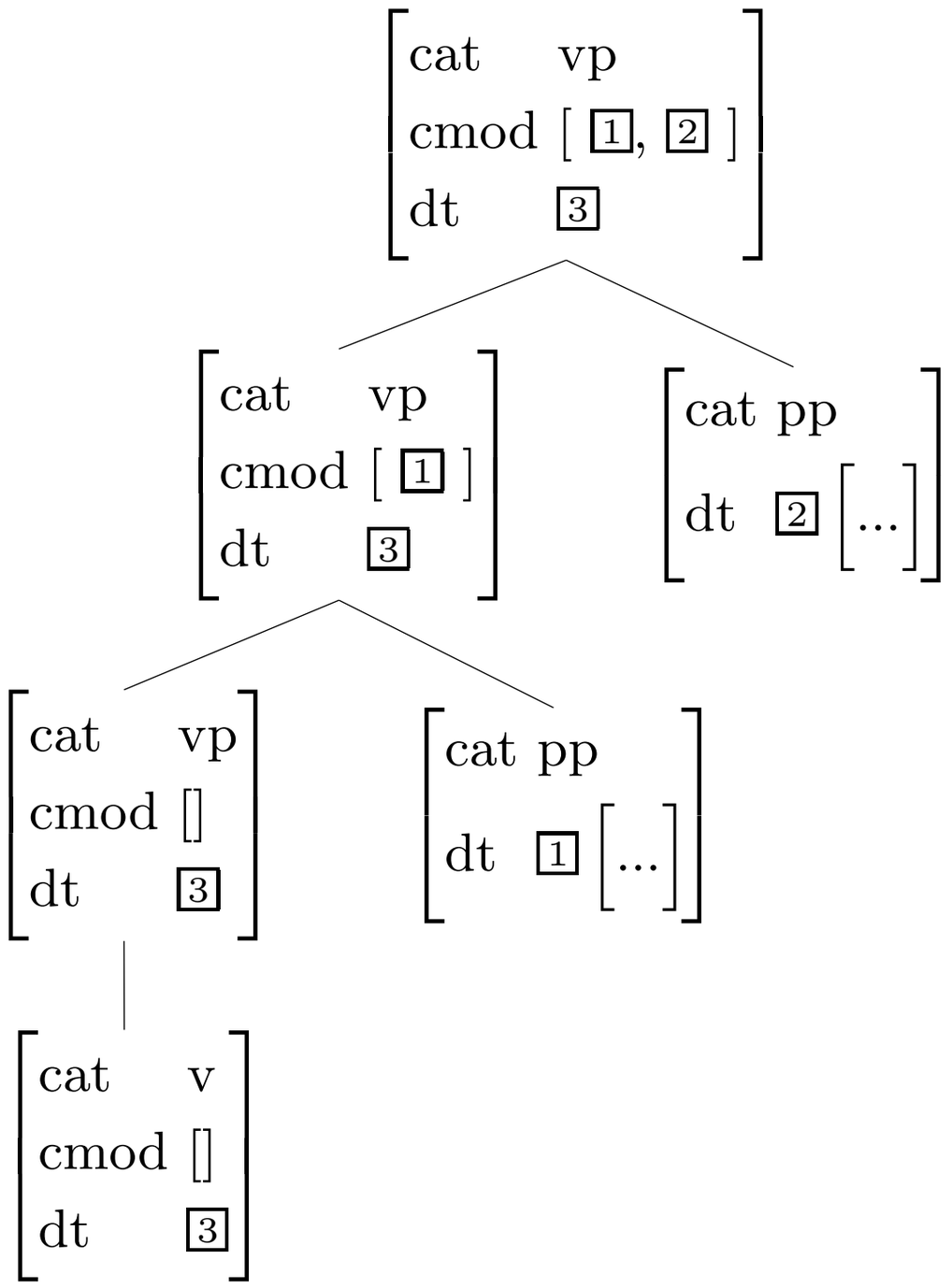
liep met de hond in Amsterdam



Methode

- Bouw gaandeweg een lijst van modifiers.
- Zodra er geen modifiers bij kunnen komen, unificeren we de lijst met het *mod* attribuut van de dependency structure.

Opbouwen lijst modifiers



Bouwen van lijst van modifiers

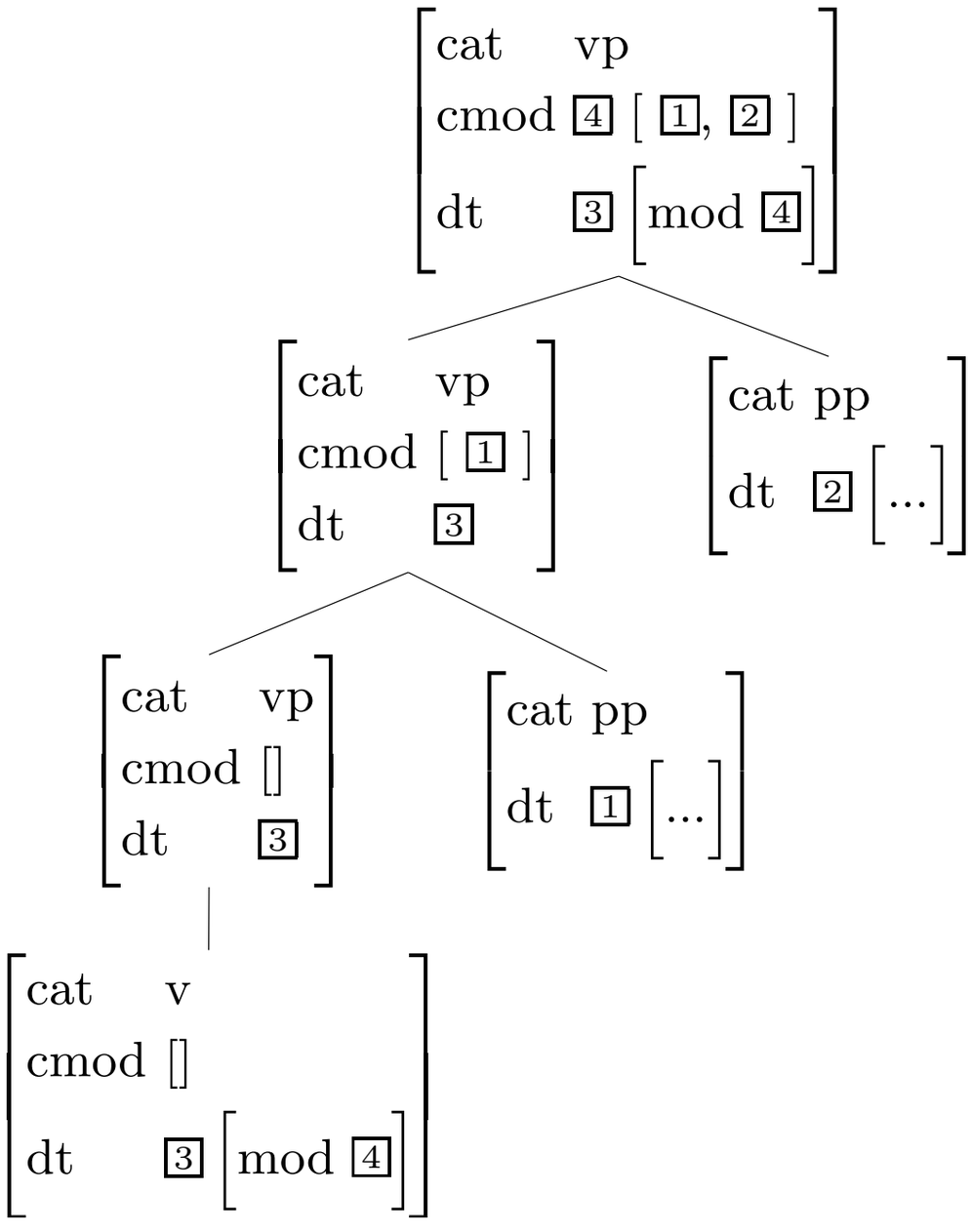
```
regel(vp_vp_pp,VP,[VP1,PP]) :-
  [...]
  VP:dt <=> VP1:dt,
  percolate_mods_principle(VP1,[PP],VP).
```

```
percolate_mods_principle(Head,Mods,Mother) :-
     Head:cmod <=> HeadMods,
     Mother:cmod <=> MotherMods,
     dts(Mods,ModsDts), %% dt van elke modifier
     append(ModsDts,HeadMods,MotherMods).
```

Modifiers in dependentie structuur

```
regel(s_np_vp,S,[NP,VP]) :-
    [...]
    unify_mods(NP),
    unify_mods(VP).

unify_mods(Cat) :-
    Cat:cmod <=> Cat:dt:mod.
```

VP na unificatie modifiers



INFORMATIE-EXTRACTIE

Informatie-extractie

- In veel toepassingen willen we uiteindelijk informatie extraheren uit zinnen die we ontleden:
  - Vraag/antwoordsystemen
  - Ontologieën
  - Taalkundige analyse
- Dependencystructuren bieden de mogelijkheid om informatie te extraheren.
- Redeneren over dependency structuren.
- Prolog ideaal voor het bewijzen dat bepaalde informatie in een dependency-structuur voorkomt.

Een voorbeeld

$$
\begin{bmatrix}
\text{HWRD} & \textit{ben} \\
\text{FRAME} & \textit{verb} \\
\text{PREDC} & \begin{bmatrix}
\text{HWRD} & \textit{hoofdstad} \\
\text{FRAME} & \textit{noun} \\
\text{DET} & \langle \begin{bmatrix} \text{HWRD} & \textit{de} \\ \text{FRAME} & \textit{det} \end{bmatrix} \rangle_{dt} \\
\text{MOD} & \langle \begin{bmatrix}
\text{HWRD} & \textit{van} \\
\text{FRAME} & \textit{prep} \\
\text{OBJ1} & \begin{bmatrix} \text{HWRD} & \textit{Nederland} \\ \text{FRAME} & \textit{name} \end{bmatrix}_{dt}
\end{bmatrix} \rangle_{dt}
\end{bmatrix}_{dt} \\
\text{SU} & \begin{bmatrix} \text{HWRD} & \textit{Amsterdam} \\ \text{FRAME} & \textit{name} \end{bmatrix}_{dt}
\end{bmatrix}_{dt}
$$

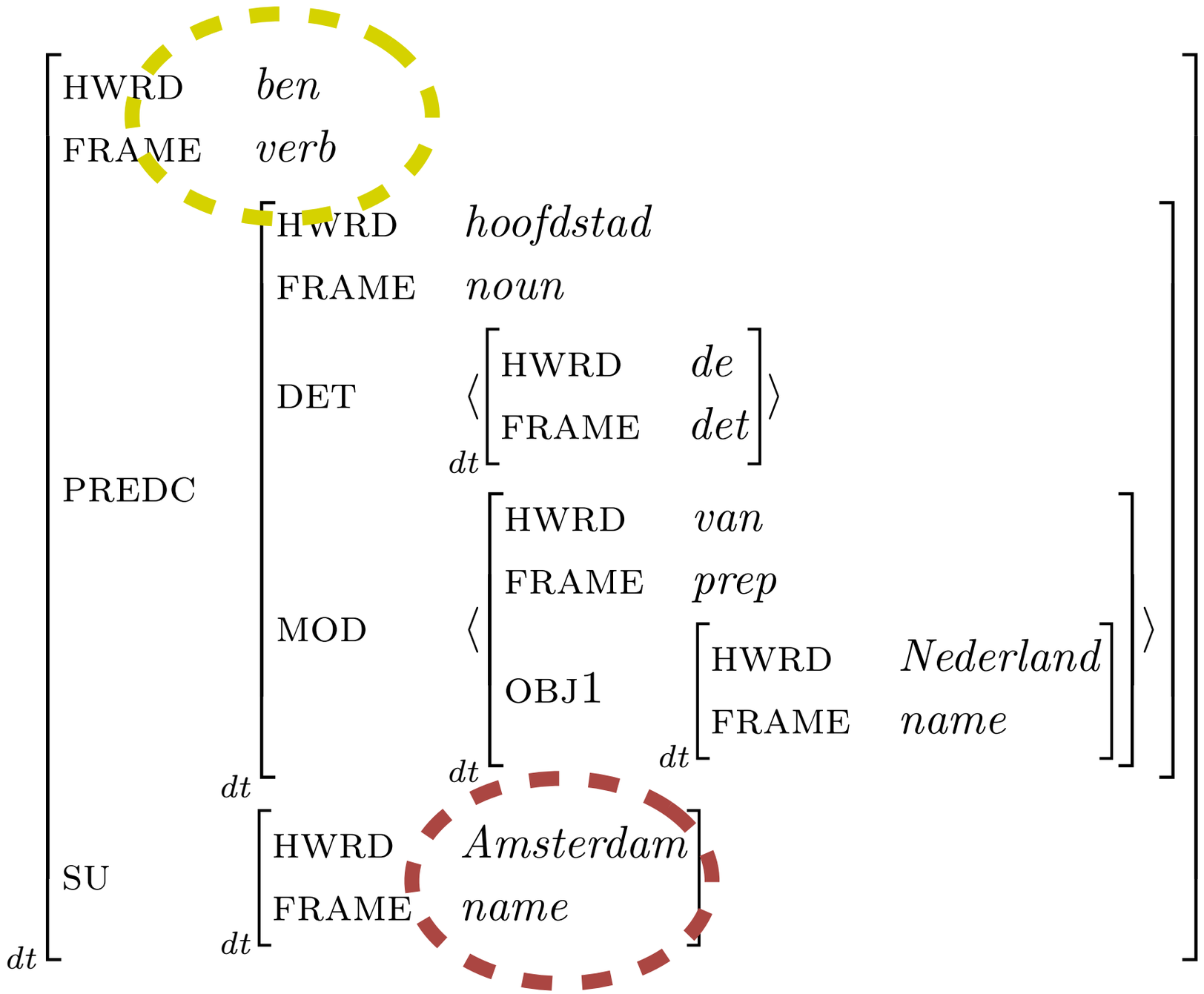
Amsterdam is de hoofdstad van Nederland -> capital(Nederland,Amsterdam)

Een voorbeeld

$$
\begin{bmatrix}
\text{HWRD} & \textit{ben} \\
\text{FRAME} & \textit{verb} \\
\text{PREDC} & \begin{bmatrix}
\text{HWRD} & \textit{hoofdstad} \\
\text{FRAME} & \textit{noun} \\
\text{DET} & \langle \begin{bmatrix} \text{HWRD} & \textit{de} \\ \text{FRAME} & \textit{det} \end{bmatrix} \rangle_{dt} \\
\text{MOD} & \langle \begin{bmatrix}
\text{HWRD} & \textit{van} \\
\text{FRAME} & \textit{prep} \\
\text{OBJ1} & \begin{bmatrix} \text{HWRD} & \textit{Nederland} \\ \text{FRAME} & \textit{name} \end{bmatrix}_{dt}
\end{bmatrix} \rangle_{dt}
\end{bmatrix}_{dt} \\
\text{SU} & \begin{bmatrix} \text{HWRD} & \textit{Amsterdam} \\ \text{FRAME} & \textit{name} \end{bmatrix}_{dt}
\end{bmatrix}_{dt}
$$

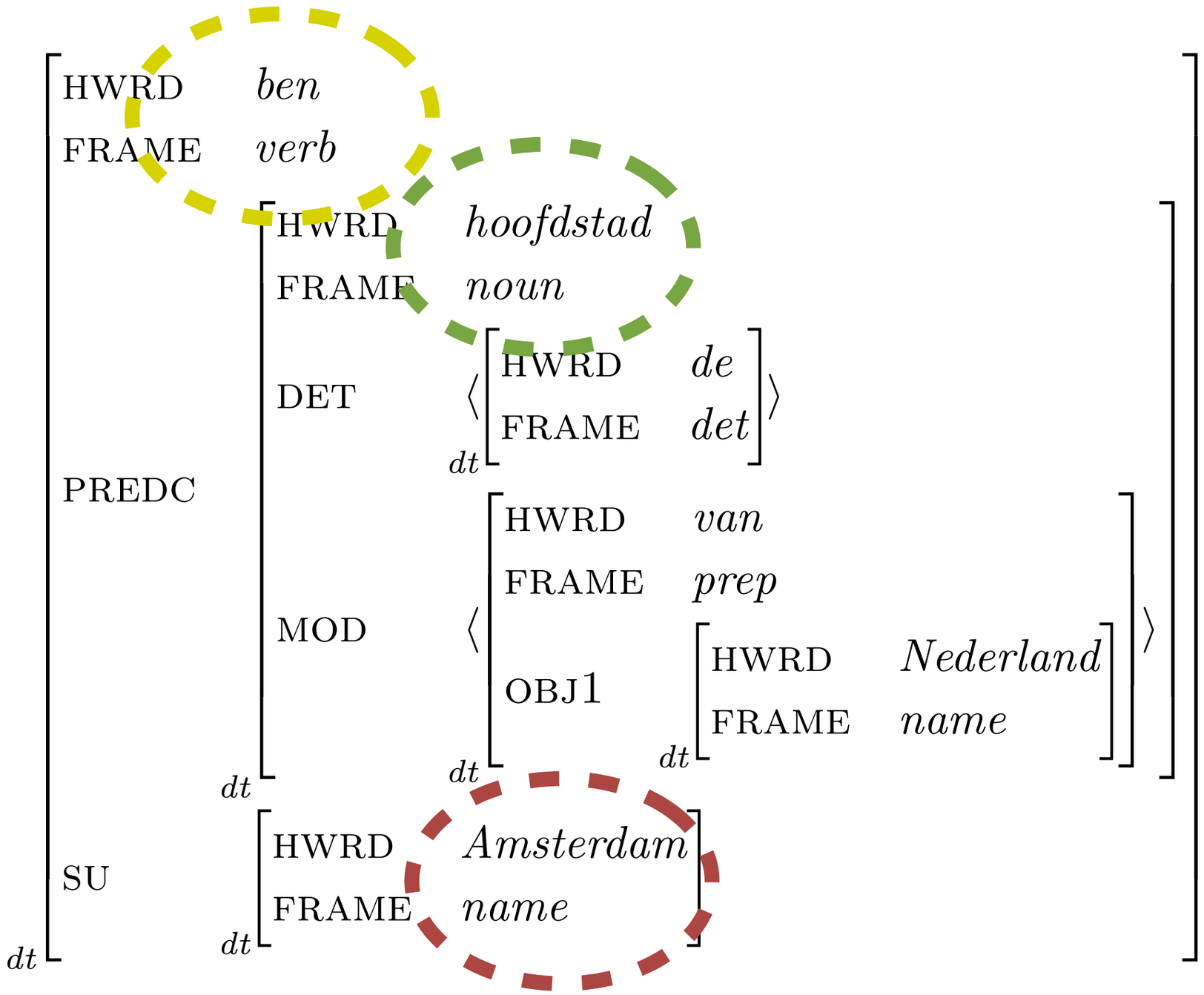
Amsterdam is de hoofdstad van Nederland -> capital(Nederland,Amsterdam)

Een voorbeeld



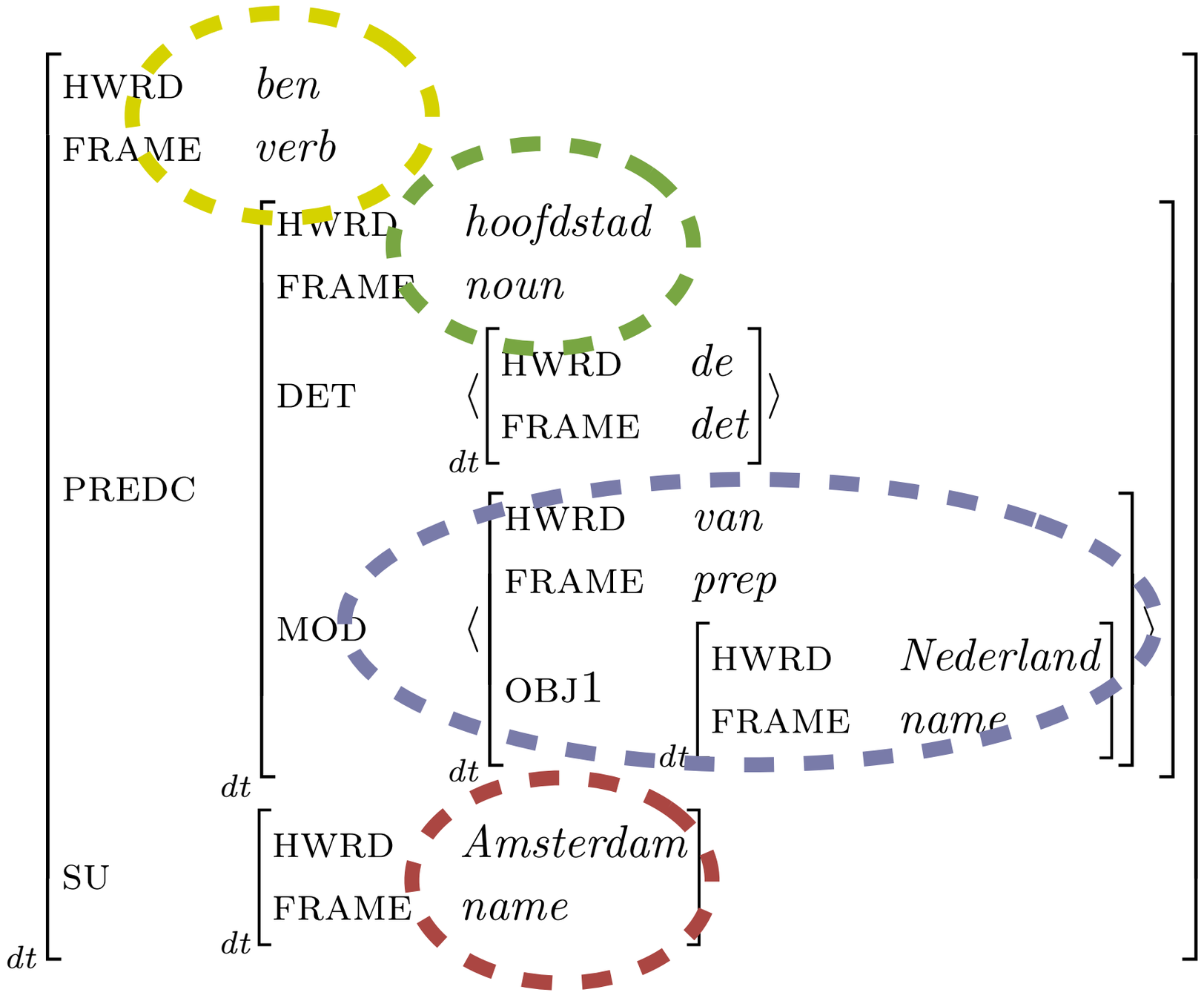
Amsterdam is de hoofdstad van Nederland -> capital(Nederland,Amsterdam)

Een voorbeeld



Amsterdam is de hoofdstad van Nederland -> capital(Nederland,Amsterdam)

Een voorbeeld



Amsterdam is de hoofdstad van Nederland -> capital(Nederland,Amsterdam)

In Prolog...

```
fact(Dt,Fact) :-
    Dt:hwrd <=> ben,
    su_pred_fact(Dt,Fact).
```

$$
\begin{bmatrix}
\text{HWRD} & \textit{ben} & & \\
\text{FRAME} & \textit{verb} & & \\
\text{PREDC} & \underset{dt}{\begin{bmatrix}
\text{DET} & \langle \underset{dt}{\begin{bmatrix} \text{HWRD} & \textit{de} \\ \text{FRAME} & \textit{det} \end{bmatrix}} \rangle \\
\text{MOD} & \langle \underset{dt}{\begin{bmatrix} \text{HWRD} & \textit{van} \\ \text{FRAME} & \textit{prep} \\ \text{OBJ1} & \underset{dt}{\begin{bmatrix} \text{HWRD} & \textit{Nederland} \\ \text{FRAME} & \textit{name} \end{bmatrix}} \end{bmatrix}} \rangle
\end{bmatrix}} \\
\text{SU} & \underset{dt}{\begin{bmatrix} \text{HWRD} & \textit{Amsterdam} \\ \text{FRAME} & \textit{name} \end{bmatrix}}
\end{bmatrix}_{dt}
$$

Onderwerp

```
%% Woord dat het onderwerp is.
subject_word(Dt,Word) :-
    Dt:su:hwrd <=> Word.
```

$$
\begin{bmatrix}
\text{HWRD} & \textit{ben} & & \\
\text{FRAME} & \textit{verb} & & \\
\text{PREDC} & \underset{dt}{\begin{bmatrix}
\text{DET} & \langle \underset{dt}{\begin{bmatrix} \text{HWRD} & \textit{de} \\ \text{FRAME} & \textit{det} \end{bmatrix}} \rangle \\
\text{MOD} & \langle \underset{dt}{\begin{bmatrix} \text{HWRD} & \textit{van} \\ \text{FRAME} & \textit{prep} \\ \text{OBJ1} & \underset{dt}{\begin{bmatrix} \text{HWRD} & \textit{Nederland} \\ \text{FRAME} & \textit{name} \end{bmatrix}} \end{bmatrix}} \rangle
\end{bmatrix}} \\
\text{SU} & \underset{dt}{\begin{bmatrix} \text{HWRD} & \textit{Amsterdam} \\ \text{FRAME} & \textit{name} \end{bmatrix}}
\end{bmatrix}_{dt}
$$

In Prolog (predikaat)

```
predicate_word(Dt,Word) :-
Dt:predc:hwrd <=> Word.
```

$$
\begin{bmatrix}
\text{HWRD} & \textit{ben} & & \\
\text{FRAME} & \textit{verb} & & \\
\text{PREDC} & \underset{dt}{\begin{bmatrix}
\text{DET} & \langle \underset{dt}{\begin{bmatrix} \text{HWRD} & \textit{de} \\ \text{FRAME} & \textit{det} \end{bmatrix}} \rangle \\
\text{MOD} & \langle \underset{dt}{\begin{bmatrix} \text{HWRD} & \textit{van} \\ \text{FRAME} & \textit{prep} \\ \text{OBJ1} & \underset{dt}{\begin{bmatrix} \text{HWRD} & \textit{Nederland} \\ \text{FRAME} & \textit{name} \end{bmatrix}} \end{bmatrix}} \rangle
\end{bmatrix}} \\
\text{SU} & \underset{dt}{\begin{bmatrix} \text{HWRD} & \textit{Amsterdam} \\ \text{FRAME} & \textit{name} \end{bmatrix}}
\end{bmatrix}_{dt}
$$

In Prolog (predikaat)

```
predicate_modifier_dt(Dt,Word,MDt) :-
    Dt:predc:mod <=> Mods,
    lists:member(MDt,Mods),
    MDt:hwrd <=> Word.
```

Alles samen...

```
fact(Dt,Fact) :-
    Dt:hwrd <=> ben,
    su_pred_fact(Dt,Fact).

su_pred_fact(Dt,capital(Country,City)) :-
    subject_word(Dt,City),
    predicate_word(Dt,hoofdstad),
    predicate_modifier_dt(Dt,van,ModDt),
    ModDt:obj1:hwrd <=> Country.
```

ALPINO

Introductie

- Ontleder voor het Nederlands op basis van:
  - Een handgeschreven (lexicaal-gedreven) unificatiegrammatica
  - Uitgebreid woordenboek
  - Hdrug omgeving
- Uitgebreide analyses in de vorm van dependency-structuren.
- Wide-coverage: 91-96% voor gangbare Nederlandse corpora.
- Goede accuratesse.
- Ontleding grote corpora (Wikipedia, Twente Nieuwscorpus, etc.)
- Gebruikt voor verschillende toepassingen (QA, sentence fusion, informatie-extractie).

Disambiguatie

- Disambiguatie op basis van een maximum entropy model.
- Leren van gewichten voor features die goede en slechte lezingen beschrijven.
- Meer in het college over disambiguatie...

Snelheidsoptimalisatie

- Uitsluiten van categorieën van woorden met behulp van een part-of-speech tagger.
- Uitsluiten van ontledingen die niet tot bruikbare analyses leiden.

Opdracht

- Ontleden van zinnen met Alpino.
- Extraheren van feiten op basis van dependency structuren.
- Over twee weken: gebruiken van geëxtraheerde feiten voor question-answering.

DEMO