Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Intelligente Systeme

– Definite-Klausel-Grammatiken –

Prof. Dr. Michael Neitzke

D1: Definite-Klausel-Grammatiken

- Wie wird in der symbolverarbeitenden KI natürliche Sprache verarbeitet?
- Konzept der Differenzlisten
- Definite-Klausel-Grammatiken

D1: Lernziele

- V1: Differenzlisten-Mechanismus zum Erzeugen und Analysieren von natürlichsprachlichen Sätzen erklären können
- V2: Erklären können, was Definite-Klausel-Grammatiken sind
  - Literaturtipp: Grune, Jacobs: „Parsing Techniques - A Practical Guide“, Kap. 6.7

Verarbeitung natürlicher Sprache: Aufgaben

- Natürliche Sprache überprüfen
  - Rechtschreibprüfung
- Natürliche Sprache erzeugen
  - Natürlichsprachliche Ausgabe
- Sprachübersetzung
- Klassifikation natürlicher Sprache
  - Spamfilter
  - Opinion Mining / Sentiment Analysis
- Information Extraction

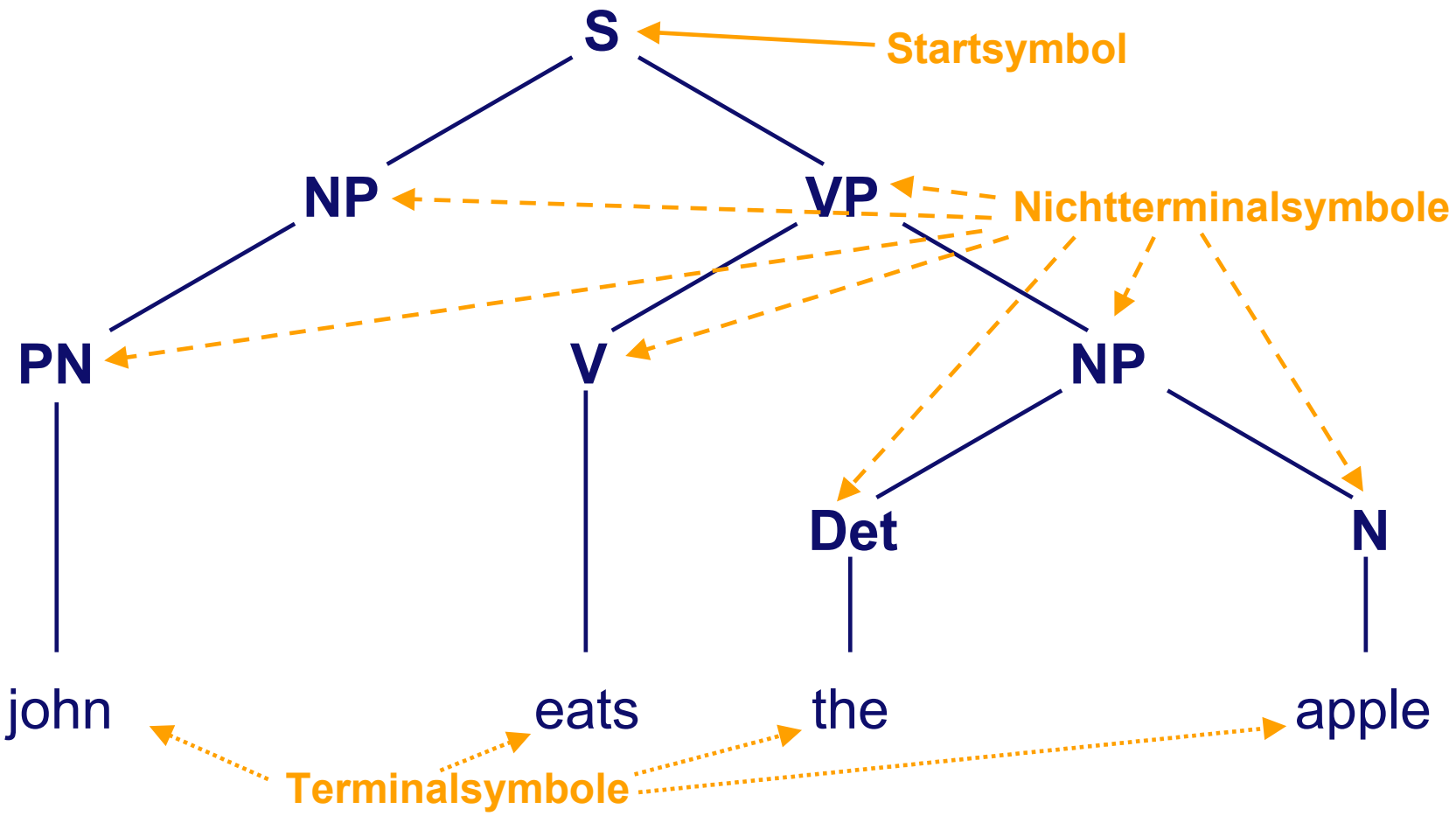
Unterschiedliche Ansätze

- Probabilistische Ansätze
  - Wie wahrscheinlich ist es, dass ein Satz mit „Ich“ beginnt und dass danach „lerne“ kommt und dass danach „gern“ kommt.
  - ... im Vergleich zu „Lerne“, „gern“, „ich“.
- Logische Ansätze (symbolverarbeitende)
  - Zerlegung eines Satzes in einzelne Satzbestandteile
  - Komposition eines Satzes aus Satzbestandteilen
  - → Grammatiken, Parsing
- Hybride Ansätze
  - Wie wahrscheinlich ist es, dass nach einem Eigennamen und anschließend einem Verb der Stammform „geben“ zwei Nominalphrasen folgen?

Erläuterungen

- Grundsätzlich gibt es zwei Aufgaben:
  - Ein natürlichsprachlicher Satz wird vorgegeben, stellt also eine Eingabe dar und muss ausgewertet werden.
  - Ein natürlichsprachlicher Satz muss erzeugt werden, stellt also eine Ausgabe dar.
- Die symbolverarbeitende KI arbeitet mit Modellen, versucht, Sätze zu parsen, grammatikalische Konstrukte zu erkennen, evt. die Bedeutung zu repräsentieren.
- Es gibt in der KI auch statistische Ansätze für natürlichsprachliche Anwendungen. So gibt es z. B. sehr leistungsfähige Übersetzungssysteme, die lediglich Tausende von vorgegebenen Übersetzungen auswerten und so unter Verwendung von Ähnlichkeitsmaßen auch neue Texte übersetzen können.

Parsebaum für englischsprachigen Satz



Kontextfreie Grammatik für Beispielsatz

S → NP VP

NP → PN

NP → Det N

VP → V

VP → V NP

PN → john | mary…

N → man | woman | dog | bird | apple…

V → sings | eats | bites | loves…

Det → the

Erläuterungen

- NP: Nominalphrase (noun phrase)
- VP: Verbalphrase (verb phrase)
- Det: Artikel (determiner)
- V: Verb (verb)
- N: Nomen (noun)
- PN: Eigenname (proper noun / proper name)

Diskutieren Sie!

- Wie würden Sie in Prolog natürlichsprachliche Sätze repräsentieren?

Lösung

- Benötigt werden dynamische Strukturen
- Also Listen:

```
[john, eats, the, apple]

[the, man, bites, the, dog]
```

Diskutieren Sie!

- Wie würden Sie die Grammatik in Prolog umsetzen?

S → NP VP

NP → PN

NP → Det N

VP → V

VP → V NP

PN → john | mary…

N → man | woman | dog | bird | apple…

V → sings | eats | bites | loves…

Det → the

Diskutieren Sie!

- Was halten Sie von diesem Lösungsversuch? Was sind die Nachteile?

```
s(S):- np(NP), vp(VP), append(NP,VP,S).
np(NP) :- pn(NP).
np(NP) :- det(D), n(N), append(D,N,NP).
vp(VP) :- v(VP).
vp(VP) :- v(V), np(NP), append(V,NP,VP).
v([eats]).
...
n([apple]).
...
```

Lösung

- Es werden blind Sätze erzeugt. Der zu überprüfende Satz wird erst am Ende mit dem erzeugten Satz verglichen. Der Inhalt des zu überprüfenden Satzes wird nicht für eine gezielte Überprüfung genutzt.
- Also extrem ineffizient, nicht praktikabel bei großem Sprachumfang.

Diskutieren Sie!

- Was ist an dieser Lösung besser?
- Was ist nach wie vor ungünstig?
- Wie würde eine ideale Vorgehensweise aussehen? Stellen Sie sich grammatikalisch falsche Sätze vor und überlegen Sie, wann und wie Sie die Fehler feststellen.

```
s(S):- append(NP,VP,S), np(NP), vp(VP).
```

usw.

Lösung

- Besser:
  - Jetzt wird der vorgegebene Satz benutzt.
- Ungünstig:
  - Der Satz wird beliebig in NP und VP zerlegt.
  - Bereits auf der zweiten Hierarchie-Ebene (z. B. NP) liegt die Gesamt-Information über den Satz nicht mehr vor.
    - Es wird nur ein willkürlich erzeugter Teilsatz nach unten durchgereicht.
- Ideales Vorgehen
  - Man könnte ja statt dessen prüfen, ob es die Grammatik zulässt, dass ein korrekter Satz mit „the" beginnt und wenn das erlaubt ist, ob danach „man" folgen darf, usw.

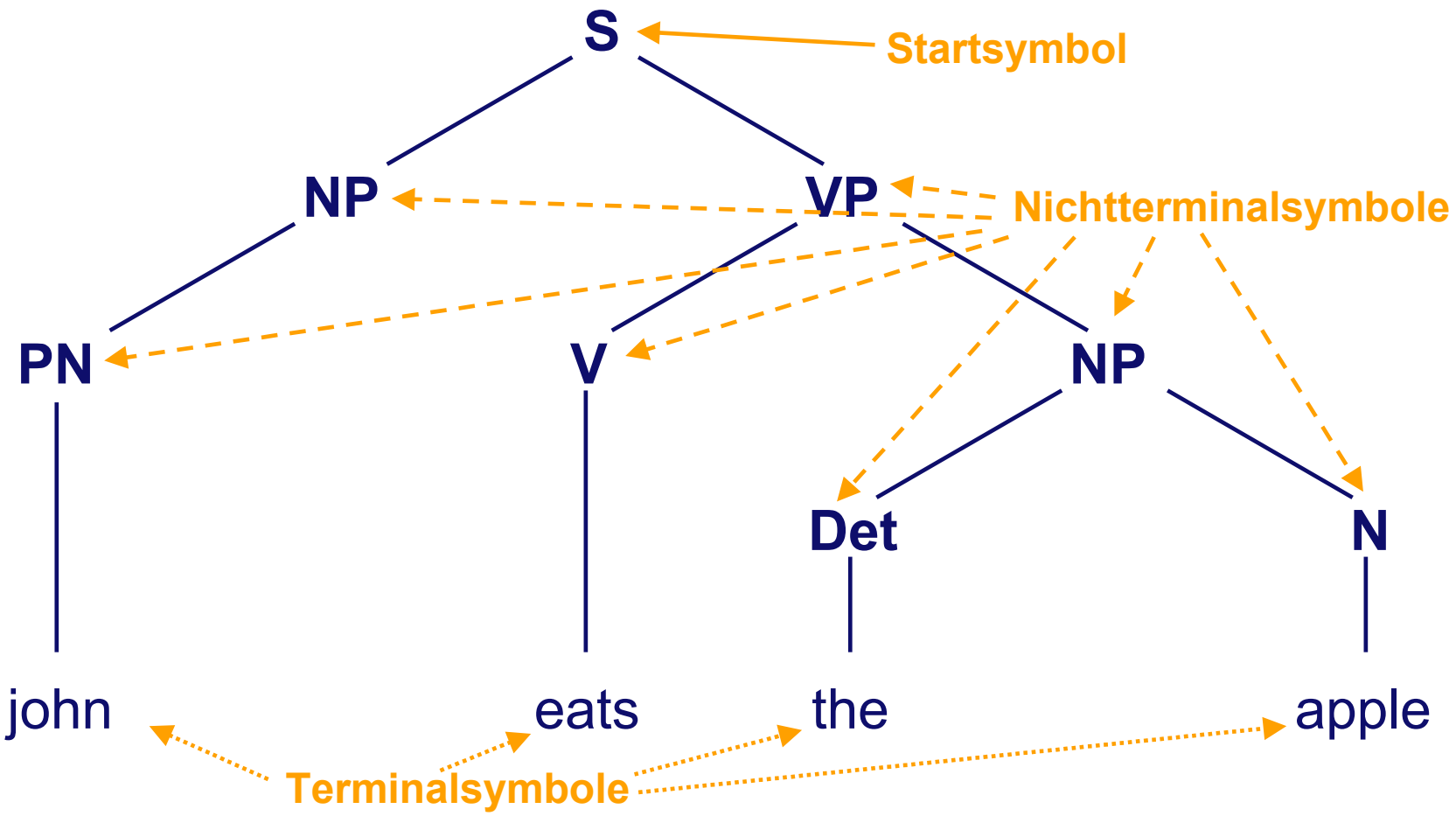
Lösung mit Differenzlisten

- Lösung: Differenzlisten (difference lists)
- Satzbestandteile werden durch die Differenz zweier Listen repräsentiert, z. B.
  - Der Artikel "the" durch
    `[the,man,bites,the,dog], [man,bites,the,dog]`
  - Oder allgemeiner durch `[the | Restliste], Restliste`
- Das Verb "bites" durch `[bites,the,dog], [the,dog]`
  - Oder allgemeiner durch `[bites | Restliste], Restliste`

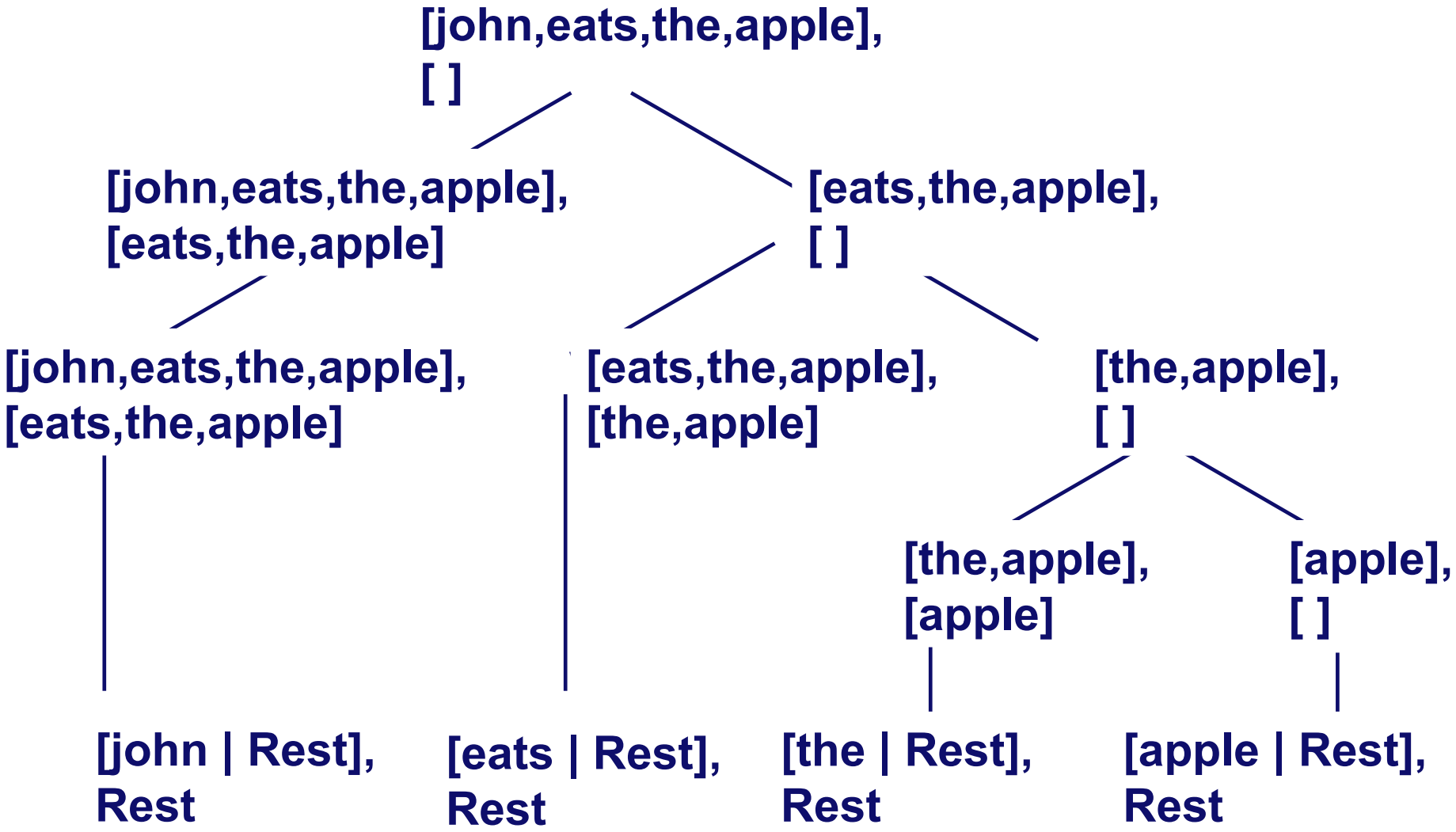
Gruppenarbeit

- Sie wissen jetzt, wie man einzelne Wörter durch Differenzlisten beschreibt.
- Aber wie werden Regeln der Grammatik in Differenzlisten ausgedrückt?
  - Z. B. VP → V NP
  - Also: `Verbalphrase` ist `Verb` gefolgt von `Nominalphrase`
- **Versuchen Sie, mit Hilfe der folgenden sieben Seiten die Modellierung und Abarbeitung mit Differenzlisten zu verstehen !**
- **Achtung: Der Programmaufruf (die Prolog-Anfrage) benötigt jetzt zwei Parameter: s([john,eats,the,apple],[]).**

Parsebaum für englischsprachigen Satz



Differenzlisten eingetragen im Parsebaum



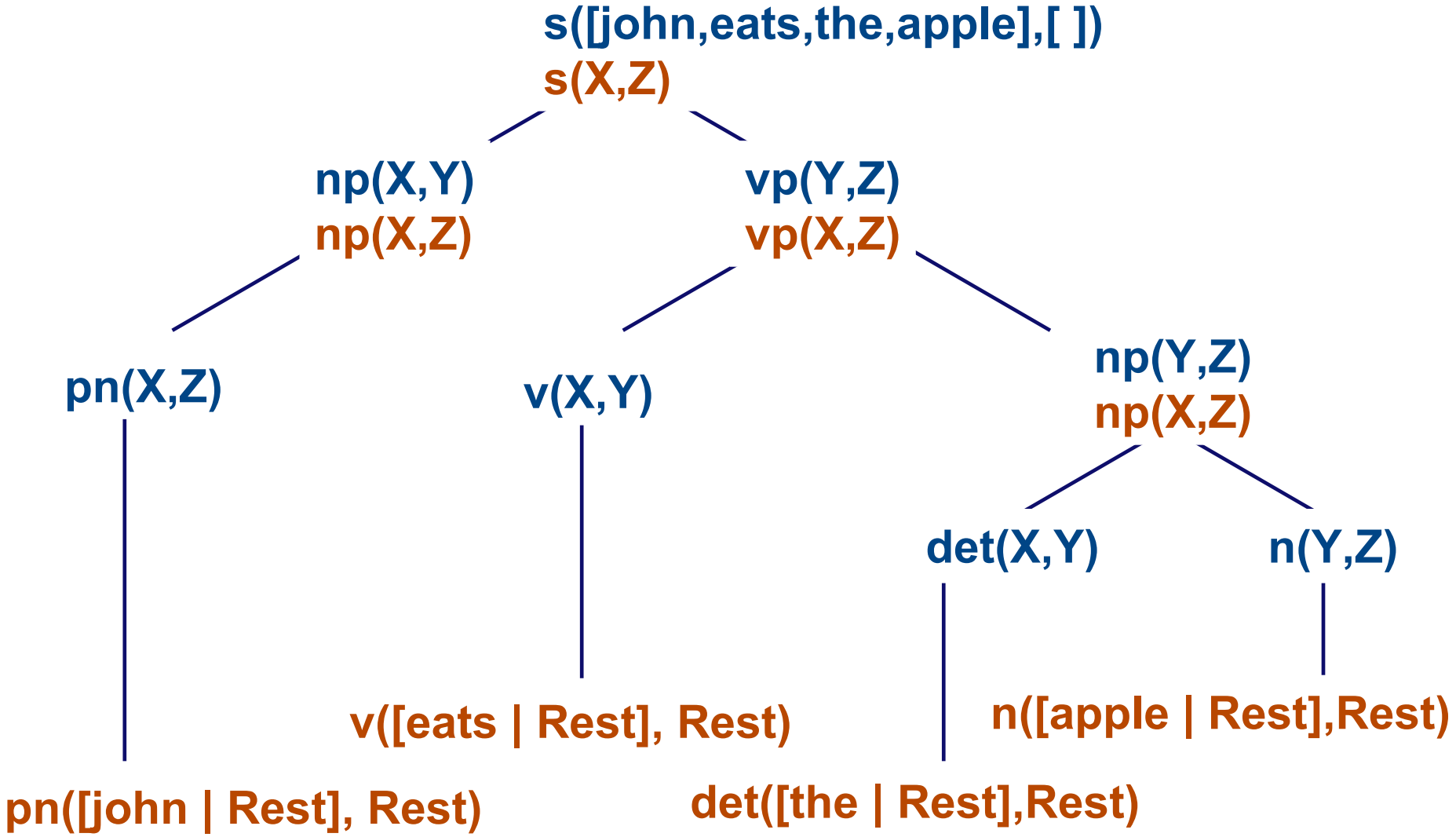
Grammatik in Differenzlisten

```
s(X,Z):-  np(X,Y), vp(Y,Z).
np(X,Z):- pn(X,Z).
np(X,Z):- det(X,Y), n(Y,Z).
vp(X,Z):- v(X,Z).
vp(X,Z):- v(X,Y), np(Y,Z).

pn([john|Rest], Rest).
pn([mary|Rest], Rest).
```

```
n([man|Rest], Rest).
n([woman|Rest], Rest).
n([dog|Rest], Rest).
n([bird|Rest], Rest).
n([apple|Rest], Rest).
v([sings|Rest], Rest).
v([eats|Rest], Rest).
v([bites|Rest], Rest).
v([loves|Rest], Rest).
det([the|Rest], Rest).
```

Grammatik-Bestandteile eingetragen im Parsebaum



Erläuterungen

- In dieser Darstellung ist der Regelkopf in **rot** dargestellt, Literale aus dem Regelrumpf hingegen in **blau**.

Abarbeitung mit Differenzlisten (1)

```
?- s([john,eats,the,apple],[]).
   Call: (6) s([john, eats, the, apple], [])
     Call: (7) np([john, eats, the, apple], _G659)
           Call: (8) pn([john, eats, the, apple], _G659)
           Exit: (8) pn([john, eats, the, apple], [eats,
                 the, apple])
     Exit: (7) np([john, eats, the, apple], [eats, the,
           apple])
     Call: (7) vp([eats, the, apple], [])
           Call: (8) v([eats, the, apple], [])
           Fail: (8) v([eats, the, apple], [])
     Redo: (7) vp([eats, the, apple], [])
           Call: (8) v([eats, the, apple], _G659)
```

Abarbeitung mit Differenzlisten (2)

```
            Exit: (8) v([eats, the, apple], [the, apple])
            Call: (8) np([the, apple], [])
                  Call: (9) pn([the, apple], [])
                  Fail: (9) pn([the, apple], [])
            Redo: (8) np([the, apple], [])
                  Call: (9) det([the, apple], _G659)
                  Exit: (9) det([the, apple], [apple])
                  Call: (9) n([apple], [])
                  Exit: (9) n([apple], [])
            Exit: (8) np([the, apple], [])
      Exit: (7) vp([eats, the, apple], [])
   Exit: (6) s([john, eats, the, apple], [])

true .
```

Definite-Klausel-Grammatik (Definite Clause Grammar)

```
s --> np, vp.

np --> pn.
np --> det, n.

vp --> v.
vp --> v, np.

pn --> [john].
pn --> [mary].
```

```
n --> [man].
n --> [woman].
n --> [dog].
n --> [bird].
n --> [apple].
v --> [sings].
v --> [eats].
v --> [bites].
v --> [loves].
det --> [the].
```

Definite-Klausel-Grammatiken in Prolog

- Vereinfachte Schreibweise (Syntactic Sugar)
- Übertragung in Differenzlisten durch Präprozessor
- [ ] kennzeichnet Terminale
  - Unterscheidung gegenüber Regeln wie `np --> pn.`

Erläuterungen

- DCGs stellen gegenüber den Differenzlisten eine kompaktere Notation dar, werden in Prolog intern aber in Differenzlisten übersetzt.

D1: Lernziele

- V1: Differenzlisten-Mechanismus zum Erzeugen und Analysieren von natürlichsprachlichen Sätzen erklären können
- V2: Erklären können, was Definite-Klausel-Grammatiken sind
  - Literaturtipp: Grune, Jacobs: „Parsing Techniques - A Practical Guide“, Kap. 6.7

D2: Konzepte zur Verarbeitung natürl. Sprache

- Lexikon
- Umgang mit morphologischen Eigenschaften
- Semantik

D2: Lernziele

- V1: Aufgabe und Funktion eines Lexikons erklären können
- V2: Aufgabe und Funktion von Attributen erklären können
- V3: Begriff der kompositionellen Semantik erklären können

Diskutieren Sie!

- Was ist an derartigen Regeln schlecht?
- Stellen Sie sich vor, dass Sie eine natürliche Sprache komplett mit allen Aspekten modelliert haben. Danach möchten Sie die Modellierung auf eine andere, grammatikalisch gleiche Sprache übertragen.

```
n --> [man].
n --> [woman].
n --> [dog].
n --> [bird].
n --> [apple].
v --> [sings].
v --> [eats].
v --> [bites].
v --> [loves].
det --> [the].
```

Lösung

- Sie vermischen Konzepte einer allgemeinen Grammatik mit den speziellen Wörtern einer Sprache.

Diskutieren Sie!

- Worin liegt der Vorteil, wenn man die Grammatik von den speziellen Wörtern einer Sprache trennt?

Lösung

- Konzeptionelle Klarheit
- Wörterbuch (Lexikon) und Grammatik können unabhängig voneinander gepflegt werden.
  - Wenige, allgemeine Regeln und viele Fakten

Lexikon: Umsetzung

- Realisierung der Trennung durch Lexikon
- Statt

```
n --> [dog].
n --> [man].
n --> [apple].
…
```

- Jetzt

```
n --> [X], {lex(X,n)}
lex(dog,n).
lex(man,n).
lex(apple,n).
…
```

- (Geschweifte Klammern: Standard PROLOG Anfragen)

Weitere Einträge in realen Lexika

- Phonologische Eigenschaften, z. B. zur Sprachausgabe
- Semantische Eigenschaften
  - Versuch, die Bedeutung zu formalisieren
- Morphologische Eigenschaften
  - Genus
  - Casus
  - Singular/Plural
  - Eigenschaften konjugierter Verben

Singular/Plural-Problematik: Diskutieren Sie!

- Wie kann man verhindern, dass folgende Sätze gebildet / akzeptiert werden?

```
The man bite.

The men bites.
```

Diskutieren Sie folgenden Lösungsansatz!

- Schlechter Ansatz:

```
s --> np_sg, vp_sg.

s --> np_pl, vp_pl.
```

Usw.

- Besser Merkmale mitführen: Attribute!

```
s --> np(Numerus), vp(Numerus)

…

lex(man,n,sg).

lex(men,n,pl).
```

Semantik: Diskutieren Sie!

- Betrachten Sie folgende natürlichsprachliche Varianten ein und derselben Aussage:
  - Der Mann beißt den Hund.
  - Der Hund wird von dem Mann gebissen.
  - Da beißt der Mann den Hund.
  - Der Hund erleidet einen Biss des Mannes.
  - Der Mann schlägt seine Zähne in den Körper des Hundes.
  - ...
- Wie würde Ihre Modellierung in Prädikatenlogik aussehen?

Lösung

- Zum Beispiel so:

  beißen(mann,hund)

Semantik

- Die prädikatenlogische Modellierung bildet die Kerninformation ab
- Kerninformation kann als Bedeutung des Satzes betrachtet werden.
- Semantik wird dann also in formaler Sprache ausgedrückt
  - für PROLOG ist Prädikatenlogik günstig
  - Beispiel: Semantik von `[the, man, bites, the, dog]`: `bite(man,dog)`
- Semantik eines Satzes wird aus Semantik der Satzbestandteile gewonnen:

  Kompositionelle Semantik

- Analog: Semantik aussagenlogischer und prädikatenlogischer Sätze wird aus Semantik der Bestandteile berechnet.

Erläuterungen

- Lassen Sie sich an dieser Stelle nicht verwirren:
  - Die Semantik prädikatenlogischer Formeln ist ein anderes Thema. Dabei wird prädikatenlogischen Formeln ein Wahrheitswert zugeordnet.
  - Hier geht es aber um die Semantik natürlichsprachlicher Sätze. Die prädikatenlogische Modellierung (oder auch jede andere Darstellung in einer formalen Sprache) ist ein Versuch, die Bedeutung natürlichsprachlicher Sätze einheitlich und klar festzulegen. Wenn dies gelingt, sind komplexe Schlussfolgerungen in Verbindung mit der Auswertung natürlichsprachlicher Sätze möglich.

Repräsentation der Semantik im Lexikon

```
lex(john,john,pn,sg).

lex(man,man,n,sg).
lex(dog,dog,n,sg).
lex(pooch,dog,n,sg).
lex(men,man,n,pl).
lex(dogs,dog,n,pl).

lex(bites,bite,v,sg).
lex(loves,love,v,sg).
lex(bite,bite,v,pl).

lex(the,_,det,_).
```

(Anm.: pooch = Köter)

Erläuterungen

- Die Semantik befindet sich hier an der zweiten Position. Dabei fällt auf, dass nicht zwischen Singular und Plural unterschieden wird. Das ist tatsächlich für die Praxis in vielen Fällen eine zu starke Vereinfachung. Eine meist sinnvolle Vereinfachung besteht zum Beispiel darin, davon zu abstrahieren, ob es sich um die erste, zweite oder dritte Person Singular handelt. (I bite. You bite. He/She/It bites. Hier würde man die Semantik des Verbs einheitlich modellieren, z.B. durch „bite“.)

Gruppenarbeit

- Sehen Sie sich die folgenden Prolog-Konzepte zum Zugriff auf die Komponenten von Strukturen an. Hierbei geht es im Wesentlichen um das 'univ'-Prädikat. 'functor' und 'arg' werden nur der Vollständigkeit halber mit aufgeführt.
- Versuchen Sie dann zu verstehen, wie der Semantik-Ausdruck aus Einzelbestandteilen zusammen gesetzt wird.
  - Versuchen Sie, den Prolog-Code auf der Folie „Berechnung der Semantik“ bottom-up zu verstehen.
  - Besonders schwierig ist der letzte Ausdruck der ersten Klausel zu verstehen (Zeile 2). Dies sollten Sie erst am Schluss versuchen, wenn der Rest klar ist.
  - Sie können sich die Unifikationsprozesse auch an der Prolog-Konsole verdeutlichen.

Prolog: Zugriff auf Komponenten von Strukturen (1)

- `functor(?Term,?Functor,?Arity)`

```
?- functor(bite(man,dog),F,A).
F = bite
A = 2
Yes

?- functor(dog,F,A).
F = dog
A = 0
Yes

?- functor(3+4,F,A).
F = +
A = 2
Yes
```

Prolog: Zugriff auf Komponenten von Strukturen (2)

- `functor(?Term,?Functor,?Arity)`

```
?- functor(T,bite,2).
T = bite(_G342, _G343)
Yes
```

```
?- functor([a,b,c],F,A).
F = '.'
A = 2
Yes
```

Prolog: Zugriff auf Komponenten von Strukturen (3)

- `arg(?Int,+Term,?Value)`

```
?- arg(2,bite(man,dog),X).
X = dog
Yes

?- arg(N,bite(man,dog),X).
N = 1
X = man;

N = 2
X = dog;
No
```

Prolog: Zugriff auf Komponenten von Strukturen (4)

- **?Term =.. ?List** ("univ")

```
?- bite(man,dog) =.. L.
L = [bite,man,dog]
Yes

?- T =.. [bite,dog,brother(john)].
T = bite(dog,brother(john))
Yes

?- 3+4 =.. L.
L = [+,3,4]
Yes

?- [a,b,c,d] =.. L.
L = ['.',a,[b,c,d]]
Yes
```

Berechnung der Semantik

```
s(SemS)                    --> np(SemNP,N), vp(SemVP,N),
                               {SemVP = [_,SemNP|_], SemS =.. SemVP}.
np(SemN,N)                 --> pn(SemN,N).
np(SemN,N)                 --> det(N), n(SemN,N).
vp([SemV,_],N)             --> v(SemV,N).
vp([SemV,_,SemNP],N)       --> v(SemV,N), np(SemNP,_).
pn(SemN,N)                 --> [X], {lex(X,SemN,pn,N)}.
n(SemN,N)                  --> [X], {lex(X,SemN,n,N)}.
v(SemV,N)                  --> [X], {lex(X,SemV,v,N)}.
det(N)                     --> [X], {lex(X,_,det,N)}.
```

- Aufruf z.B. durch: `s(Sem,[the,man,bites,the,dog],[]).`

Definite-Klausel-Grammatiken: Bewertung

- Einerseits "Syntactic Sugar"
- Andererseits eigenständiger Formalismus mit bestimmten Eigenschaften
- Nachteile
  - Gefahr von Schleifen
    - Bei links-rekursiven Regeln wie z. B. s --> s conj s
    - (Top Down Parser geraten in Schleifen bei links-rekursiven Grammatiken)
    - Probleme nicht inhärent in DCGs, sondern in der Art, wie sie in PROLOG umgesetzt werden
  - Viele Argumente machen die Ausdrücke unhandlich
  - Ineffizient

D2: Lernziele

- V1: Aufgabe und Funktion eines Lexikons erklären können
- V2: Aufgabe und Funktion von Attributen erklären können
- V3: Begriff der kompositionellen Semantik erklären können