Seminar Satzgrammatik
WS 2003/2004

Teilgrammatik des Deutschen
(Anfang: Kategorien und Merkmale)

Hans Leiß, CIS

29.10.2003

Zusammenfassung

Dieses Skriptum ist der Anfang einer Dokumentation der zu entwickelnden Satzgrammatik. Es werden lexikalische Kategorien, ihre Merkmale und Teile des Vollformenlexikons `word/2` und gewisser Stammlexika `wrd/2` angegeben.

Darüber hinaus werden Grammatikregeln für Nominal- und Präpositionalphrasen angegeben und gewisse Rektions- und Kongruenzprädikate angegeben.

Bisheriger Inhalt:

(i) Wortarten und ihre Merkmale

(ii) Grammatikregeln für Nominalphrasen

(iii) Grammatikregeln für Präpositionalphrasen

(iv) Beispieltext

(v) Seminarplan

1 Grammatik des Deutschen

1.1 Grammatische Kategorien und grammatische Relationen

Eine sorgfältige Klassifizierung der sprachlichen Ausdrücke führt zu einer großen Anzahl von Klassen. Diese Klassen lassen sich im Wesentlichen als eine Hierarchie darstellen, wobei die Unterklassen durch *Klassenparameter* angegeben werden. Durch die Wahl gleicher Parameter lassen sich Beziehungen zwischen Unterklassen verschiedener Klassen andeuten.

Klassenparameter beschreiben Eigenschaften, die bei der Zusammensetzung von Ausdrücken restringierend wirken, aber nicht variabel sind. Zum Beispiel ist das Genus bei Nominalphrasen ein Klassenparameter.

Jeder Ausdruck einer Unterklasse der Kategorie kann in verschiedenen *Formen* auftreten. Was die möglichen Formen sind, wird –abhängig von der jeweiligen Ausdrucksklasse– durch *Formparameter* beschrieben.

In der Implementierung werden die Wort- und Ausdruckskategorien als zweistellige Prädikate

```
Kategorie([Klassenparameter,...],[Formparameter,...])
```

angegeben.

Zum Beispiel ist der Kasus ein Formparameter von Nominalphrasen, der Modus ein Formparameter von Verben und Aussagesätzen, die Verbstellung ein Formparameter von Sätzen.

1.1.1 Kongruenz

Die *Kongruenz* zweier oder mehrer Teilausdrücke eines Ausdrucks ist eine Beziehung zwischen den Formen der Teilausdrücke, die von der Konstruktion des Ausdrucks abhängt. Sie wird als Beziehung zwischen den Formparametern der Kategorien einer Regel formuliert:

```
A ---> A_1(Klasse_1,Form_1), ... , A_k(Klasse_k,Form_k)
   :- kongruenz(Form_1,...,Form_k).
```

Die Kongruenzbedingung hängt von der Kategorie `A` und der jeweiligen Regel ab. Sie legt die Formen normalerweise nicht eindeutig fest, sondern beschränkt die Unabhängigkeit der Formen der Teilausdrücke.

Die Kongruenz von Teilausdrücken (in einer Formdimension) ist normalerweise an der Gleichheit der Formwerte (dieser Formdimension) zu erkennen, z.B. demselben Numerus oder Kasus. Dann kann man die Kongruenzbedingung durch gleiche Variable ausdrücken:

```
np(Klasse,[N,K]) ---> det(KlasseD,[N,K]), n(KlasseN,[N,K])
   :- numerus(N),kasus(K), bedingungen(Klasse,KlasseD,KlasseN).
```

Zur Kongruenz des finiten Verbs mit dem Satzsubjekt gehört aber z.B. auch die Tatsache, daß Subjektsätze nur mit finitem Verb in der dritten Person im Singular einen Satz bilden; trotzdem gibt man den Subjektsätzen keine Merkmale Numerus und Person. (Eine *Merkmalskongruenz* ist also eine n-stellige Relation K zwischen (Tupeln von) Formeigenschaften von n Ausdrücken, deren Formen $(f_1,\ldots,f_n)$ nur gemeinsam variieren, d.h. wenn eine Komponente sich ändert, müssen alle anderen es auch tun. Beipiel: Dominos $[n,m]$ und $[x,y]$, die nur dann als $[n,m][x,y]$ zueinander *passen*, wenn $n=1, x=a$, oder $n=2, x=b$ usw.)

1.1.2 Rektion

Ein Teilausdruck *regiert* die Form eines anderen Teilausdrucks in einem Ausdruck, wenn die Form des zweiten von der Unterklasse des ersten Teilausdrucks abhängt. Die Rektion wird in einer Grammatikregel als (funktionale) Beziehung zwischen der Unterklasse der Kategorie des regierenden und der Form der Kategorie des regierten Teilausdrucks formuliert:

```
A ---> A1(Klasse_1,Form_1), ... , A_k(Klasse_k,Form_k)
:- rektion(j,i,Klasse_j,Form_i).
```

Oft ist die Rektion als Gleichheit des Wertes eines Formparameters des abhängigen mit dem Wert eines Klassenparameters des unabhängigen Ausdrucks einer Konstruktion ausgedrückt:

```
np ---> det(KlasseD,[G,..]), a(KlasseA,[G,...]), n([G],Form)
:- genus(G).
```

In einer Nominalphrase haben sich die Form von Artikel und Adjektiv dem Genus des Nomens anzupassen, das (Genus des) Nomen(s) regiert also die Form (-dimension Genus) von Artikel und Adjektiv.

Problem: Man kann Rektion nur dann durch Verwendung derselben Variable beschreiben, wenn (i) die regierte Formeigenschaft einer Konstituente durch *dieselben* Merkmalwerte dargestellt wird wie die regierende (inhärente) Eigenschaft, und wenn (ii) die regierte Eigenschaft nur bei *obligatorischen* Konstituenten auftritt. Z.B. wird die Adjektivdeklination `Adekl` in einer Nominalphrase vom Artikel regiert, aber das adjektivische Attribut einer `n1` ist optional. Falls man das Merkmal `Adekl` einer `n1` ohne Adjektiv auf den Defaultwert `-` setzt, blockiert das die Verbindung mit einem Artikel, der die für die `Adekl` einen der Werte `st,sw,ge` verlangt.

Für eine deklarative Beschreibung der Syntax scheint es am besten, bei der Synthese des Merkmals einer Phrase für die Eigenschaften nicht vorhandener Konstituenten den Wert `-` anzunehmen, und für eine Alternative von Eigenschaften einer Konstituenten, die an der Konstituente nicht erkennbar ist, den Wert `unsp`. Die Synthetisierung wäre dann in der Syntakregel durch eine Instruktion `{synth(Konst.Eigenschaften+,Attr-)}` anzugeben. (Allerdings ist `unsp` im wesentlichen ein disjunktives Constraint.)

In einem eigenen Unifikationsalgorithmus könnte man natürlich eine Fallunterscheidung wie

⟨*Synthese eines Attributs a la SML*⟩≡

```
Attr := if (!KonstAttr = Instantiated{Value}
        then f(Value) else NewValue
```

einbauen.

⟨*Synthese eines Attributs in Prolog*⟩≡

```
<Phrase>(Attr) --> <ConstA>(ValueA),...,<ConstB>(ValueB),
                   {synth(ValueA,...,ValueB,Attr)}.

synth(A,...,B,Attr) :-  % falls alle A,...,B belegt sind!
        (A,...,B) = (-,...,-)
        -> Attr = NewValue
        ; Attr = f(A,...,B)
```

Man muß abwägen zwischen

(i) einer kompakten Sprachbeschreibung mit optionalen Konstituenten wie in

```
np --> det[Adekl], ?ap(Adekl'), n, {synth(Adekl',Adekl)}
```

und dem Rechenaufwand für `synth`, und

(ii) einer weniger übersichtlichen Sprachbeschreibung mit doppelten Regeln, wie in

`np --> det n` und `np --> det[Adekl], ap(Adekl), n`

aber dafür der effizienten Berechnung der synthetisierten Merkmale durch die Unifikation.

Die folgenden Regeln machen das noch nicht, da es bisher nur für den Parser `Parser/earley.doerre.dcg.pl` funktioniert. Für den `.epl`-Parser muß noch die Vorberechnung durchdacht werden.

1.1.3 Merkmalvererbung

Merkmale einer zusammengesetzten Phrase hängen oft von den Merkmalen ihrer Teilphrasen ab; umgekehrt hängen auch die Merkmale der Teilphrasen von denen der gesamten Phrase ab. Diese Abhängigkeiten wird in einer Grammatikregel durch Bedingungen an die Merkmale formuliert:

```
A(Klasse,Form) ---> A_1(Klasse_1,Form_1), ..., A_k(Klasse_k,Form_k)
:- vererbung(Klasse_1,Form_1,...,Klasse_k,Form_k,Klasse,Form).
```

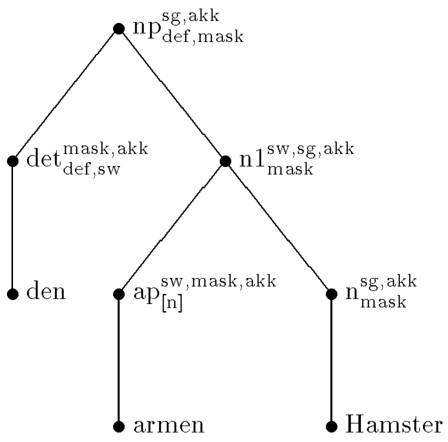
Die Klassenparameter sollten hier nicht von den Formparametern abhängen, und oft hängen auch die Formparameter nur von Formparametern ab. In vielen Fällen ist die Abhängigkeit einfach eine Gleichheit in einem Merkmal der Kopf- und Konstituentenkategeorien, z.B. beim Klassenmerkmal Genus und dem Formmerkmal Kasus der Nominalphrase:

```
np([G,...],[K,...]) ---> det([G,...],[K,...]), n([G],[K,...])
:- genus(G), kasus(K), ... .
```

Die Klasse der zusammengesetzten Phrase hängt im allgemeinen auch von der jeweiligen Grammatikregel ab. Merkmale, die nur zur Formulierung von Kongruenz und Rektion zwischen den Konstituenten dienen, sollten nicht an die gesamte Phrase vererbt werden, wie z.B. die Adjektivdeklination innerhalb der Nominalphrase.

Graphische Darstellung von Komplementrahmen, Rektion und Kongruenz

In der Baumdarstellung einer Analyse stehen dieFormparameter als obere, die Klassenparameter als untere Indizes an den Kategoriennamen. Komplementrahmen sind als Listen angegeben:



Wenn man wie hier beim Adjektiv überhaupt ein Subjekt im Komplementrahmen vorsieht, sollte es für die attributive Verwendung ein einfaches Nomen vorsehen, `[n]`, für die prädikative Verwendung eine Nominalphrase in einem bestimmten Kasus, normalerweise `[nom]`.

1.1.4 Aufbau der Grammatik aus Teilgrammatiken

Obwohl die verschiedenen Ausdruckskategorien einer Sprache simultan rekursiv von einander abhängen, muß man die Grammatik aus Teilgrammatiken für die einzelnen Kategorien zusammensetzen, um eine übersichtliche Organisation und Entwicklung zu erreichen.

Wegen der gegenseitigen Abhängigkeiten kann man nicht einfach Teilgrammatiken in einzelnen Dateien angeben, die andere Teilgrammatiken laden; sonst würde das Laden der Dateien nicht abbrechen. Wir wollen die Einzelgrammatiken so schreiben, daß

(i) Phrasenkategorien der anderen Teilgrammatiken wie lexikalische Kategorien behandelt werden,

(ii) Die Vorberechnungen der Teilgrammatiken unabhängig von einander möglich sind und in die Vorberechnung der kombinierten Grammatik eingehen.

Dazu muß jede Teilgrammatiken so aufgeteilt werden, daß

(i) ihre Prädikate zur Definition der Merkmalbereiche und ihre anderen Hilfsprädikate ohne ihre Regeln auch zur Vorberechung und Benutzung anderer Grammatiken geladen werden können,

(ii) die Vorberechnung für die Teilgrammatik separat vorgenommen werden kann,

(iii) Kombination beliebiger Teilgrammatiken durch Kombinationen von Hilfsprädikaten, Vorberechnungen und Regeln möglich ist.

Die Gesamtgrammatik lädt die Definitionen der Kategorien und ihrer Merkmale sowie die Teilgrammatiken. Bei den Vorberechnungen der Hilfsprädikate ist eine modulare Berechnung wünschenswert, hier aber nicht verwirklicht.

Bemerkung 1.1 Die Vorberechnung muß die Kongruenz- und andere Prädikate einer Teilgrammatik in die vorberechnete (Gesamt-) Grammatik übertragen. In der Teilgrammatik wird durch `auxiliary_predicates/1` die Liste dieser Prädikate angegeben. Die Gesamtgrammatik muß am Anfang eine Deklaration `:- multifile auxiliary_predicates/1.` enthalten, damit nicht die in der letzten Teilgrammatik eingeführten Prädikate die der vorigen überschreiben.

Beim erneuten Laden der Grammatik oder einer Teilgrammatik werden solche in mehreren Dateien erklärten Prädikate nur teilweise ersetzt: die Anteile, die aus der erneut geladenen Grammatik stammen, werden gelöscht und ihre Neudefinition am Ende angehängt. (Kann diese Änderung der Reihenfolge der Klauseln hier einen unerwünschsten Effekt haben?)

Ein wesentlicher Aspekt der Grammatik ist die Wahl der Klassen- und Formparameter und ihrer Wertebereiche für die verschiedenen lexikalischen und phrasalen Kategorien.

1.2 Lexikalische Kategorien und ihre Merkmale

Merkmale, die die Zugehörigkeit zu Flexionsklassen, die Passivierbarkeit von Verben, das Hilfsverb im Perfekt u.ä. beschreiben, müssen in einem Stammformenlexikon enthalten sein; in einem Vollformenlexikon, von dem hier ausgegangen wird, braucht man sie nicht. Nur wenn man Ausdrücke in verschiedene Formen setzen will, braucht man diese Informationen; dafür genügt es, wenn die Vollform einen Verweis auf die Stammform enthält.

Da wir hier nur an der Analyse interessiert sind, können wir auch den Verweis auf die Stammform bei den Vollformen weglassen. Wir legen uns (solange es geht) auf folgende lexikalischen Kategorien und ihre Merkmale fest.[1]

⟨*Lexika/wortarten.cat.pl*⟩≡
```
% :- module(wortarten,
%           [wortkategorie/1,restrict/2,
%            verbklasse/1,finiteVerbform/1,finiteVerbformPraef/1,subjekt/1,objekt/1,
%            verbtempus/1,modus/1,numerus/1,person/1,verbstellung/1,infiniteVerbform/1,
%            genus/1,numerus/1,kasus/1,definitheit_pron/1,
%            adjektivdeklination/1,komparation/1,adverbialeFunktion/1]).
:- multifile restrict/2.

⟨Wortkategorien⟩
⟨Lexikalische Merkmale⟩
```

1.2.1 Verben

(i) finite Verben: wir unterscheiden zwei Fälle, normale Verben und solche, die bei Verberst- und Verbzweitstellung ein Präfix abspalten. Für die zweite Sorte ist die Verbstellung im Satz daher ein Formparameter, und das abgespaltene Präfix wird wie ein Komplement als Klassenparameter im Lexikoneintrag notiert:

⟨*Wortkategorien*⟩≡
```
wortkategorie(v(VerbKlasse,FiniteForm)) :-
        verbklasse(VerbKlasse),
        finiteVerbform(FiniteForm).
wortkategorie(v(VerbKlasse,FiniteForm)) :-
        verbklassePraef(VerbKlasse),
        finiteVerbformPraef(FiniteForm).
```

[1] Diese Festlegungen werden in den Grammatikregeln bisher nur zum Teil eingehalten, insbesondere *nicht* in den Grammatiken aus `Seminarbeitraege`. Nach der Vorberechnung werden die in den Grammatikregeln verwendeten Kategorien mit den hier deklarierten verglichen und ggf. Warnungen ausgegeben.

⟨*Lexikalische Merkmale*⟩≡

```
verbklasse([Rahmen]) :-
        komplemente(Rahmen).
finiteVerbform([Pers,Num,Temp,Mod]) :-
        person(Pers),
        numerus(Num),
        verbtempus(Temp),
        modus(Mod).

verbklassePraef([Rahmen,Praef]) :-
        verbklasse([Rahmen]),
        praefix(Praef).
finiteVerbformPraef([Pers,Num,Temp,Mod,Vst]) :-
        finiteVerbform([Pers,Num,Temp,Mod]),
        verbstellung(Vst).
```

Die abtrennbaren Verbpräfixe, `praef([Praef],[])`, muß man `Lexika/praepositionen.wrd` entnehmen.

⟨*Lexikalische Merkmale*⟩+≡

```
praefix(Praef) :-
        ensure_loaded('Lexika/praepositionen.wrd'),
        word(_, praef([Praef],[])).
```

Jeder Komplementrahmen ist eine Liste, die das Subjekt und die Objekte beschreibt:

⟨*Lexikalische Merkmale*⟩+≡

```
komplemente([C1]) :-
        subjekt(C1).
komplemente([C|Cs]) :-
        subjekt(C),
        objekte(Cs).
```

Dabei werden die Komplemente nur in der Reihenfolge angegeben, in der sie in Sätzen mit Verbletztstellung auftreten. (Die übrigen Stellungen werden in der Grammatik durch Regeln formuliert. Optionale Argumente werden vorläufig durch eigene Komplementrahmen, aber mittelfristig wie in `word(lese,v([ [nom,?(akk)] ],Form))` angegeben.)

Jedes Komplement wird durch eine Abkürzung seiner Ausdrucksform beschrieben, aus der seine Ausdrucksklasse hervorgeht:

⟨*Lexikalische Merkmale*⟩+≡

```
subjekt(nom).
subjekt(gen).
subjekt(dat).
subjekt(es).
subjekt(daß).
subjekt(inf).
subjekt(infzu).

objekte([]).
objekte([C]) :- objekt(C).
objekte([C,D]) :- objekt(C), objekt(D).

objekt(gen).
objekt(dat).
objekt(dat(refl)).
```

```
objekt(akk).
objekt(akk(refl)).
objekt(praep(P)).
objekt(daß).
objekt(qu).
objekt(inf).
objekt(infzu).
objekt(infzu(nom)).
objekt(infzu(dat)).
objekt(infzu(akk)).
objekt(infzu(praep(P))).
```

Bem.: Durch `objekt(P) :- praeposition(P).` werden die Komplementrahmen vom Lexikon abhängig und vor allem werden dann zu viele Grammatikregeln entstehen.

Die Unterklassen, die für die Syntaxanalyse unterschieden werden müssen, kann man an den Verbrahmen ablesen: intransitive Verben haben kein Objekt, Modalverben sind diejenigen mit einem `inf`-Objekt, reflexive Verben diejenigen mit einem `dat(refl)`- oder `akk(refl)`-Objekt. Durch `qu` werden interrogative Objekte bei Verben wie `fragen`, `nicht wissen` usw. angegeben.

Kontrollverben haben `infzu`-Objekte, die durch `infzu(Kasus)` dargestellt werden: der Kasus gibt an, welches Nominalobjekt des Kontrollverbs das implizite Subjekt des Infinitivs ist; falls im Infinitivobjekt Reflexivpronomina vorkommen, müssen diese mit dem impliziten Subjekt in Person und Numerus übereinstimmen: z.B. hat `raten` den Komplementrahmen `[nom,dat,infzu(dat)]` und `versprechen` den Rahmen `[nom,dat,infzu(nom)]`. Analog ist `infzu(praep(C))` für Kontrollverben mit Präpositionalobjektkontrolle gedacht, z.B. hat `appellieren` den Rahmen `[nom,praep(an),infzu(praep(an))]`.

Die Formmerkmale der finiten Verbformen haben folgende Wertebereiche:

⟨*Lexikalische Merkmale*⟩+≡
```
verbtempus(praes).
verbtempus(praet).

modus(ind).
modus(konj).

numerus(sg).
numerus(pl).

person(1).
person(2).
person(3).

verbstellung(erst).
verbstellung(zweit).
verbstellung(letzt).
```

Die Verbstellung ist eigentlich ein Formmerkmal von einfachen Sätzen, das aber eben auch die Abspaltung von Verbpräfixen regiert und deshalb hier auftritt.

(ii) infinite Verbformen: Bei der Partizipform eines Verbs, die zur Bildung von Säzten im Perfekt gebraucht wird, notieren wir auch das Hilfsverb, mit dem das Perfekt gebildet wird.

⟨*Wortkategorien*⟩+≡
```
wortkategorie(v(VerbKlasse,InfiniteForm)) :-
```

```
        infiniteVerbklasse(VerbKlasse),
        infiniteVerbform(InfiniteForm).
```

⟨*Lexikalische Merkmale*⟩+≡

```
infiniteVerbform([inf]).
infiniteVerbform([infzu]).
infiniteVerbform([part]).
infiniteVerbform([part2]).

infiniteVerbklasse([Rahmen]) :-
        % verbklasse([_Subj|Rahmen]).
        verbklasse(Rahmen).
infiniteVerbklasse([Rahmen,PerfAux]) :-
        % verbklasse([_Subj|Rahmen]),
        verbklasse(Rahmen),
        hilfsverb(PerfAux).

hilfsverb(sein).
hilfsverb(haben).
```

Die Hilfsverben sollten dem Lexikon entnommen werden.

1.2.2 Nomen

Vorläufig wird folgende Parametrisierung vorgeschlagen:

(i) Eigennamen: nur Vor- und Nachnamen sind Wortkategorien, zusammengesetzte Eigennamen werden in der Nominalphrasengrammatik als syntaktische Kategorie erklärt.

⟨*Wortkategorien*⟩+≡

```
wortkategorie(vn([Gen],[Num,Kas])) :-
        genus(Gen), numerus(Num), kasus(Kas).
wortkategorie(nn([],[Num,Kas])) :-
        numerus(Num), kasus(Kas).
```

Bei den Eigennamen `en` wird das Genus in der Regel `unsp` sein und nur bei den Vornamen `vn` ein echtes Genus sein; bei Nachnamen `nn` ist die Unterscheidung nach dem Genus unnötig. In beiden Fällen wird gelegentlich eine Pluralform verwendet, z.B. `alle Kläuse, die Meiers`, aber das wird hier ignoriert.

(ii) Personalpronomen:[2]

⟨*Wortkategorien*⟩+≡

```
% wortkategorie(pron([Def,Pers,Gen],[Num,Kas])) :-
wortkategorie(pron([Def,PronPos],[Gen,Num,Kas,Pers])) :-
        definitheit_pron(Def),
        \+ (Def = rel; Def = refl),
        numerus(Num),
        person(Pers),
        (((Pers = 1 ; Pers = 2), Genus = unsp)
        ; (Pers = 3, genus(Gen), Num = sg)),
        kasus(Kas),
        (PronPos = pers; PronPos = pos ; PronPos = neg).
```

[2] Das wird unten leider nicht eingehalten! Dort wird `pron([Def,PronPos],[Gen,Num,Kas,Pers]))` benutzt.

```
wortkategorie(pron([refl],[unsp,Num,Kas,Pers])) :-
        numerus(Num),
        person(Pers),
        kasus(Kas).
wortkategorie(pron([rel],[Gen,Num,Kas])) :-
        ((genus(Gen), Num = sg) ; (Gen = unsp, Num = pl)),
        kasus(Kas).
```

Die Definitheit wird zur Unterscheidung zwischen bestimmten (definiten) und unbestimmten, quantifizierenden, relativierenden, interrogativen Pronomina gebraucht.

⟨*Lexikalische Merkmale*⟩+≡

```
definitheit_pron(def).
definitheit_pron(indef).
definitheit_pron(rel).
definitheit_pron(qu).
```

(a) Definite Personalpronomen:

⟨*Lexika/pronomen.wrd*⟩≡

```
% Personalpronomen:
word(ich, pron([def,pers], [unsp, sg, nom, 1]), ppron).
word(meiner, pron([def,pers], [unsp, sg, gen, 1]), ppron).
word(mir, pron([def,pers], [unsp, sg, dat, 1]), ppron).
word(mich, pron([def,pers], [unsp, sg, akk, 1]), ppron).

word(du, pron([def,pers], [unsp, sg, nom, 2]), ppron).
word(deiner, pron([def,pers], [unsp, sg, gen, 2]), ppron).
word(dir, pron([def,pers], [unsp, sg, dat, 2]), ppron).
word(dich, pron([def,pers], [unsp, sg, akk, 2]), ppron).

word(er, pron([def,pers], [mask, sg, nom, 3]), ppron).
word(seiner, pron([def,pers], [mask, sg, gen, 3]), ppron).
word(ihm, pron([def,pers], [mask, sg, dat, 3]), ppron).
word(ihn, pron([def,pers], [mask, sg, akk, 3]), ppron).

word(sie, pron([def,pers], [fem, sg, nom, 3]), ppron).
word(ihrer, pron([def,pers], [fem, sg, gen, 3]), ppron).
word(ihr, pron([def,pers], [fem, sg, dat, 3]), ppron).
word(sie, pron([def,pers], [fem, sg, akk, 3]), ppron).

word(es, pron([def,pers], [neut, sg, nom, 3]), ppron).
word(seiner, pron([def,pers], [neut, sg, gen, 3]), ppron).
word(ihm, pron([def,pers], [neut, sg, dat, 3]), ppron).
word(es, pron([def,pers], [neut, sg, akk, 3]), ppron).

word(wir, pron([def,pers], [unsp, pl, nom, 1]), ppron).
word(unser, pron([def,pers], [unsp, pl, gen, 1]), ppron).
word(uns, pron([def,pers], [unsp, pl, dat, 1]), ppron).
word(uns, pron([def,pers], [unsp, pl, akk, 1]), ppron).

word(ihr, pron([def,pers], [unsp, pl, nom, 2]), ppron).
word(euer, pron([def,pers], [unsp, pl, gen, 2]), ppron). % ??
word(eurer, pron([def,pers], [unsp, pl, gen, 2]), ppron). % ??
word(euch, pron([def,pers], [unsp, pl, dat, 2]), ppron).
```

```
word(euch, pron([def,pers], [unsp, pl, akk, 2]), ppron).

word(sie, pron([def,pers], [unsp, pl, nom, 3]), ppron).
word(ihrer, pron([def,pers], [unsp, pl, gen, 3]), ppron).
word(ihnen, pron([def,pers], [unsp, pl, dat, 3]), ppron).
word(sie, pron([def,pers], [unsp, pl, akk, 3]), ppron).
```

(b) Indefinite Personalpronomen: (Da man zwischen positiver und negativer Version wählen kann, sollte das ein *Form*parameter sein.)

⟨*Lexika/pronomen.wrd*⟩+≡

```
% Indefinite Pronomen:
word(man, pron([indef,pers], [mask, sg, nom, 3])).  % mask!

word(einer, pron([indef,pos], [mask, sg, nom, 3]), ein).
word(eines, pron([indef,pos], [mask, sg, gen, 3]), ein). % kommt das vor?
word(einem, pron([indef,pos], [mask, sg, dat, 3]), ein).
word(einen, pron([indef,pos], [mask, sg, akk, 3]), ein).

word(eine,  pron([indef,pos], [fem, sg, nom, 3]), ein).
word(einer, pron([indef,pos], [fem, sg, gen, 3]), ein).
word(einer, pron([indef,pos], [fem, sg, dat, 3]), ein).
word(eine,  pron([indef,pos], [fem, sg, akk, 3]), ein).

word(eines, pron([indef,pos], [neut, sg, nom, 3]), ein).
word(eines, pron([indef,pos], [neut, sg, nom, 3]), ein).
word(einem, pron([indef,pos], [neut, sg, dat, 3]), ein).
word(eines, pron([indef,pos], [neut, sg, akk, 3]), ein).

word(keiner, pron([indef,neg], [mask, sg, nom, 3]), ein).
word(keines, pron([indef,neg], [mask, sg, gen, 3]), ein). % kommt das vor?
word(keinem, pron([indef,neg], [mask, sg, dat, 3]), ein).
word(keinen, pron([indef,neg], [mask, sg, akk, 3]), ein).

word(keine,  pron([indef,neg], [fem, sg, nom, 3]), ein).
word(keiner, pron([indef,neg], [fem, sg, gen, 3]), ein).
word(keiner, pron([indef,neg], [fem, sg, dat, 3]), ein).
word(keine,  pron([indef,neg], [fem, sg, akk, 3]), ein).

word(keines, pron([indef,neg], [neut, sg, nom, 3]), ein).
word(keines, pron([indef,neg], [neut, sg, nom, 3]), ein).
word(keinem, pron([indef,neg], [neut, sg, dat, 3]), ein).
word(keines, pron([indef,neg], [neut, sg, akk, 3]), ein).


word(etwas, pron([indef,pos], [unsp, sg, nom, 3]), etw). % neut
word(etwas, pron([indef,pos], [unsp, sg, akk, 3]), etw).
word(etwas, pron([indef,pos], [unsp, sg, dat, 3]), etw).
word(etwas, pron([indef,pos], [unsp, sg, gen, 3]), etw).

word(nichts, pron([indef,neg], [unsp, sg, nom, 3]), etw). % neut
word(nichts, pron([indef,neg], [unsp, sg, akk, 3]), etw).
word(nichts, pron([indef,neg], [unsp, sg, dat, 3]), etw).
word(nichts, pron([indef,neg], [unsp, sg, gen, 3]), etw).
```

```
word(jemand, pron([indef,pos], [mask, sg, akk, 3]), jmd).
word(jemand, pron([indef,pos], [mask, sg, nom, 3]), jmd).
word(jemand, pron([indef,pos], [mask, sg, dat, 3]), jmd).
word(jemandem, pron([indef,pos], [mask, sg, dat, 3]), jmd).
word(jemanden, pron([indef,pos], [mask, sg, akk, 3]), jmd).
word(jemandes, pron([indef,pos], [mask, sg, gen, 3]), jmd). % poss?

word(niemand, pron([indef,neg], [mask, sg, akk, 3]), jmd).
word(niemand, pron([indef,neg], [mask, sg, nom, 3]), jmd).
word(niemand, pron([indef,neg], [mask, sg, dat, 3]), jmd).
word(niemandem, pron([indef,neg], [mask, sg, dat, 3]), jmd).
word(niemanden, pron([indef,neg], [mask, sg, akk, 3]), jmd).
word(niemandes, pron([indef,neg], [mask, sg, gen, 3]), jmd). % poss?
```

(c) Reflexivpronomen:

⟨*Lexika/pronomen.wrd*⟩+≡

```
% Reflexivpronomen:
word(mir,  pron([refl],[unsp, sg, dat, 1]), rpron).
word(mich, pron([refl],[unsp, sg, akk, 1]), rpron).

word(dir,  pron([refl],[unsp, sg, dat, 2]), rpron).
word(dich, pron([refl],[unsp, sg, akk, 2]), rpron).

word(sich, pron([refl],[unsp, sg, dat, 3]), rpron).
word(sich, pron([refl],[unsp, sg, akk, 3]), rpron).

word(uns,  pron([refl],[unsp, pl, dat, 1]), rpron).
word(uns,  pron([refl],[unsp, pl, akk, 1]), rpron).

word(euch, pron([refl],[unsp, pl, dat, 2]), rpron).
word(euch, pron([refl],[unsp, pl, akk, 2]), rpron).

word(sich, pron([refl],[unsp, pl, dat, 3]), rpron).
word(sich, pron([refl],[unsp, pl, akk, 3]), rpron).

word(einander, pron([refl], [unsp, pl, nom, 3]), rpron).
word(einander, pron([refl], [unsp, pl, akk, 3]), rpron).
word(einander, pron([refl], [unsp, pl, dat, 3]), rpron).
word(einander, pron([refl], [unsp, pl, gen, 3]), rpron).
```

Man sollte vielleicht `einander` nicht als `refl`, sondern als `rezi` für *reziprok* klassifizieren; es sollte nur bei Verben verwendbar sein, deren Subjekt notwendigerweise eine Nominalphrase im Plural ist, die als kollektives Subjekt zu verstehen ist (keine distributive Bedeutung).

(d) Interrogativpronomen: (vs. interrogative Determinatoren)

⟨*Lexika/pronomen.wrd*⟩+≡

```
word(wer, pron([qu], [unsp, sg, nom, 3])).   % statt qu auch rel!
word(wessen, pron([qu], [unsp, sg, gen, 3])).
word(wem, pron([qu], [unsp, sg, dat, 3])).
word(wen, pron([qu], [unsp, sg, dat, 3])).

word(was, pron([qu], [neut, sg, nom, 3])).
word(was, pron([qu], [neut, sg, akk, 3])).
```

(e) Relativpronomen: Hier ist die Person kein Formmerkmal, da dieselbe Form für alle Personen verwendet wird (wobei in erster und zweiter Person Singular das grammatische Genus dem natürlichen angepaßt wird).

⟨*Lexika/pronomen.wrd*⟩+≡

```
word(der, pron([rel], [mask, sg, nom]), drel).
word(dessen, pron([rel], [mask, sg, gen]), drel).
word(dem, pron([rel], [mask, sg, dat]), drel).
word(den, pron([rel], [mask, sg, akk]), drel).

word(die, pron([rel], [fem, sg, nom]), drel).
word(deren, pron([rel], [fem, sg, gen]), drel).
word(der, pron([rel], [fem, sg, dat]), drel).
word(die, pron([rel], [fem, sg, akk]), drel).

word(das, pron([rel], [neut, sg, nom]), drel).
word(dessen, pron([rel], [neut, sg, gen]), drel).
word(dem, pron([rel], [neut, sg, dat]), drel).
word(das, pron([rel], [neut, sg, akk]), drel).

word(die, pron([rel], [unsp, pl, nom]), drel).
word(deren, pron([rel], [unsp, pl, gen]), drel).
word(denen, pron([rel], [unsp, pl, dat]), drel).
word(die, pron([rel], [unsp, pl, akk]), drel).
```

Bem. Die Possessivpronomen stehen unter den Determinatoren/Adjektiven, da sie als Determintatoren und als Prädikative (`mein Buch` vs. `das Buch ist meines`) verwendet werden.

Frage: Ist es ein Kriterium für Determinatoren, daß sie in diesen beiden Funktionen verwendet werden können?

(iii) Einfache Nomen:

⟨*Wortkategorien*⟩+≡

```
wortkategorie(n([Gen],[Num,Kas])) :-
        genus(Gen), numerus(Num), kasus(Kas).
```

(iv) Nominalisierte Verben: (`nv` ??)

⟨*Wortkategorien*⟩+≡

```
wortkategorie(n([Gen,Rahmen],[Num,Kas])) :-
        genus(Gen), numerus(Num), kasus(Kas),
        komplemente(Rahmen).
```

(v) Nominalisierte Adjektive: bei diesen sind die Komparation und die Adjektivdeklination eine Formeigenschaft (`das Schwierigste`, `ein Beamter`, `der Beamte`):

⟨*Wortkategorien*⟩+≡

```
wortkategorie(na([Gen],[Komp,Adekl,Num,Kas])) :-
        genus(Gen), numerus(Num), kasus(Kas),
        komplemente(Rahmen),
        komparation(Komp),
        adjektivdeklination(Adekl).
```

Man muß vielleicht noch einen Komplementrahmen vorsehen, je nach den Adjektiven.

Fehler: Die aus dem CISLEX übernommenen Lexikoneinträge zu nominalisierten Adjektiven sind aber relativ ungenau, vgl. `wissenschaftsreligion.na.wrd`. Bei vielen nominalisierten Adjektiven sind eigentlich keine Pluralformen erlaubt, bei manchen keine Komparatationsformen, aber das wird von der Vollformenerstellung nicht (oder nicht ausreichend) berücksichtigt.

(vi) Titel: Bei Titeln wie `Dr.` wird das Genus auf `unsp` festgelegt:

⟨*Wortkategorien*⟩+≡
```
wortkategorie(titel([unsp],[Num])) :-
        numerus(Num).
```

(vii) Anrede: Hierzu zählen `Herr`, `Frau` usw., wobei folgende Parameter berücksichtigt werden:

⟨*Wortkategorien*⟩+≡
```
wortkategorie(anrede([Gen],[Num,Kas])) :-
        genus(Gen), numerus(Num), kasus(Kas).
```

Die Merkmalbereiche für Numerus und Person sind schon oben bei den Verben festgelegt; die für Numerus und Kasus sind:

⟨*Lexikalische Merkmale*⟩+≡
```
genus(mask).
genus(fem).
genus(neut).
genusErw(Gen) :-
        genus(Gen) ; Gen = unsp.

kasus(nom).
kasus(gen).
kasus(dat).
kasus(akk).
```

Bem: Die indefiniten Pronomina im Singular wie `man,jemand`, haben im CISLEX das Genus `unsp`, sollten aber `mask` haben: man soll *seine* Fehler nicht wiederholen!

1.2.3 Adjektive

Bei den Klassenmerkmalen der Adjektive unterscheiden wir nach dem Komplementrahmen und den syntaktischen Funktionen, die das Adjektiv übernehmen kann: welche Adjektive sind attributiv zum Nomen, welche adverbial und welche prädikativ verwendbar, welche graduierbar?

⟨*Wortkategorien*⟩+≡
```
wortkategorie(a([Rahmen,Funktionen],[Komp,Adekl,Gen,Num,Kas])) :-
        komparation(Komp), adjektivdeklination(Adekl),
        ((Num = sg, genusErw(Gen)) ; (Num = pl, Gen = unsp)),
        kasus(Kas),
        komplemente(Rahmen),
        verwendbarkeit(Funktionen).

verwendbarkeit(praedik).    % testhalber
verwendbarkeit(attrib).     % testhalber
verwendbarkeit(adverbial).  % testhalber
```

Warnung: im Lexikon steht statt `[Rahmen,Funktionen]` *nur eine leere Merkmalliste!* Man muß also Komplementrahmen noch ergänzen. (Flexionsprogramme?) Deswegen ergänzen wir:

⟨*Wortkategorien*⟩+≡
```
wortkategorie(a([Rahmen],[Komp,Adekl,Gen,Num,Kas])) :-
        komparation(Komp), adjektivdeklination(Adekl),
        genus(Gen), numerus(Num), kasus(Kas),
        komplemente(Rahmen).
wortkategorie(a([],[Komp,Adekl,Gen,Num,Kas])) :-
        komparation(Komp), adjektivdeklination(Adekl),
        genusErw(Gen), numerus(Num), kasus(Kas).
```

Frage: Sollen Ordinal- und Kardinalzahlen als Unterklassen markiert werden? Das machen wir erst einmal nicht, sondern setzen:

⟨*Wortkategorien*⟩+≡
```
wortkategorie(ordinalzahl).
wortkategorie(kardinalzahl).
```

Bei den Formmerkmalen der deklinierten Adjektive unterscheiden wir neben Genus (erweitert um den Wert `unsp` wie im CISLEX), Numerus und Kasus nach der Adjektivdeklination und der Komparation. Diese haben die Werte

⟨*Lexikalische Merkmale*⟩+≡
```
adjektivdeklination(st).
adjektivdeklination(sw).
adjektivdeklination(ge).

komparation(pos).
komparation(komp).
komparation(sup).
```

Unflektiert verwendete Adjektive werden durch andere Lexikoneinträge dargestellt, von denen es bisher noch nicht viele gibt:

⟨*Wortkategorien*⟩+≡
```
wortkategorie(a([],[Komp,praedik]))  :-
        komparation(Komp).
```

(Vgl. `Seminarbeitraege/LenkLeonardi/adjektive.wrd`). Das Format kann sich ändern; es sollte z.B. zumindest noch ein Komplementrahmen erlaubt sein. Vorläufig behandeln wir nur einstellige Adjektive:

⟨*Wortkategorien*⟩+≡
```
wortkategorie(a([Kas],[Komp,praedik]))  :-
        kasus(Kas),
        komparation(Komp).
wortkategorie(a([Praep],[Komp,praedik]))  :-
        % praeposition(Praep), % oder a([praep(Praep)],Form) ??
        komparation(Komp).
```

Frage: Welche Partikel sind Adjektivmodifikatoren (Gradpartikel)?

1.2.4 Präpositionen

Prä-, Post- und Circumpositionen werden im Stammformenlexikon `wrd/2` nach ihrem Komplement und der adverbialen Funktion, die sie anzeigen können, unterklassifiziert:

⟨*Wortkategorien (unbenutzt)*⟩≡
```
wortkategorie(pr([[Kas],AdvF],[Praep])) :-
        kasus(Kas),
        adverbialeFunktion(AdvF).
wortkategorie(po([[Kas],AdvF],[Praep])) :-
        kasus(Kas),
        adverbialeFunktion(AdvF).
wortkategorie(pc([[Kas],AdvF],[Praep])) :-
        kasus(Kas),
        adverbialeFunktion(AdvF).
```

Damit ließen sich die Lexika überprüfen (wenn man in den Lexika die Klammern um das Komplement tatsächlich schreiben würde.). Das wird bisher nicht gemacht.

Solange die adverbiale Funktion in der lexikalischen Analyse weggeblendet wird, benutzen wir (zur Überprüfung von `lexword/2`) nur:

⟨*Wortkategorien*⟩+≡
```
wortkategorie(pr([[Kas]],[Praep])) :-
        kasus(Kas).
wortkategorie(po([[Kas]],[Praep])) :-
        kasus(Kas).
wortkategorie(pc1([[Kas]],[Praep])) :-
        kasus(Kas).
wortkategorie(pc2([[Kas]],[Praep])) :-
        kasus(Kas).

wortkategorie(proPP([Def],[Praep])) :-
        definitheit_pp(Def).
```

Die adverbialen Funktionen werden bei der Wortart der Adverbiale angegeben.

Als Komplemente von Präpositionen sind erlaubt: durch einen Kasus `Kas` angegebene Nominalphrasen, durch eine adverbiale Funktion `AdvF` angegebene Adverbiale (darunter Adverbien, wie in `bis später`, und Präpositionalphrasen), und durch eine Präposition `Praep` angegebene Präpositionalphrasen:

```
word(bis, pr([dir(hin)],[bis])).
word(bis, pr([temp],[bis])).
word(bis, pr([zu(dat)],[bis])).
```

Die Präposition selbst wird in der Kategorie als Formmerkmal gespeichert, um sie an die Präpositionalphrase vererben zu können. Dort wird der von der Präposition regierte Kasus mit angegeben, damit bei den adverbial verwendeten Präpositionalphrasen z.B. direktionale und lokale Adverbiale `in,an,auf` unterschieden werden können.

Die semantische Unterklassifizierung der Präpositionen erhöht die (scheinbare) syntaktische Mehrdeutigkeit sehr stark. Um zunächst nicht zwischen der möglichen Verwendung einer Präpositionalphrase als Adverbial oder als Komplement unterscheiden zu müssen, projizieren wir das Merkmal `AdvF` bei der lexikalischen Analyse weg: wir leiten Einträge ins Vollformenlexikon `word/2` aus Einträgen eines

Stammformenlexikons `wrd/2` ab, die semantische Information enthalten (und das Komplement nicht als Liste verpacken):

⟨*Lexika/praepositionen.wrd*⟩≡
```
word(Praep,Cat) :-
        wrd(Praep, pr([Kas,_AdvF],Form)),
        Cat = pr([[Kas]],Form).
word(Praep,Cat) :-
        wrd(Praep, po([Kas,_AdvF],Form)),
        Cat = po([[Kas]],Form).
```

Erst beim Einbau einer Präpositionalphrase in einen Satz wird überprüft, ob sie eine bestimmte adverbiale Funktion erfüllen kann. Da die Präposition (mit dem regierten Kasus) im Formmerkmal der Präpositionalphrase steht, kann die semantische Information (mit Hilfe von `wrd(für,...)`) rekonstruiert werden.[3]

(i) Alphabetische Liste der Präpositionen, klassifiziert nach dem regierenden Kasus und den adverbialen Funktionen (nach Helbig/Buscha: Deutsche Grammatik), die eine mit der Präposition eingeleitete Phrase haben kann:

⟨*Lexika/praepositionen.wrd*⟩+≡
```
wrd(ab,        pr([dat,lok],[ab])).
wrd(ab,        pr([dat,temp],[ab])).
wrd(abseits,   pr([gen,lok],[abseits])).
wrd(abzüglich, pr([gen,modl],[abzüglich])).
wrd(als,       pr([Kas,modl],[als(Kas)])) :-    kasus(Kas). % du/dich als Spezialist/en
wrd(an,        pr([dat,lok],[an(dat)])).
wrd(an,        pr([dat,temp],[an(dat)])).
wrd(an,        pr([akk,dir(hin)],[an(akk)])).
wrd(angesichts,pr([gen,kaus],[angesichts])).
% anhand
% à
wrd(anläßlich, pr([gen,kaus],[anläßlich])).
wrd(anstatt,   pr([gen,modl],[anstatt])).
% anstelle
wrd(auf,       pr([dat,lok],[auf(dat)])).
wrd(auf,       pr([dat,temp],[auf(dat)])).  % auf dem Ereignis
wrd(auf,       pr([akk,dir(hin)],[auf(akk)])).
wrd(auf,       pr([akk,temp],[auf(akk)])).  % auf (die) Dauer
wrd(auf,       pr([akk,kaus],[auf(akk)])).  % auf Empfehlung (hin)
wrd(auf,       pr([akk,modl],[auf(akk)])).  % (bis) auf die Sekunde
% aufgrund
wrd(aus,       pr([dat,dir(her)],[aus])).
wrd(aus,       pr([dat,kaus],[aus])).
wrd(aus,       pr([dat,modl],[aus])).
wrd(ausschließlich, pr([gen,kop],[ausschließlich])).
wrd(außer,     pr([dat,modl],[außer])).
wrd(außer,     pr([dat,kop],[außer])).
wrd(außerhalb, pr([gen,lok],[außerhalb])).
wrd(außerhalb, pr([gen,temp],[außerhalb])).
```

[3] Für die Vorberechnungen müßte aber die adverbiale Funktion an den Merkmalen der Konstituentenkategorien ablesbar sein!

```
% behufs
wrd(bei,       pr([dat,lok],[bei])).
wrd(bei,       pr([dat,temp],[bei])).
wrd(bei,       pr([dat,kond],[bei])).
% betreffs
wrd(bezüglich, pr([gen,kaus],[bezüglich])).
wrd(binnen,    pr([dat,temp],[binnen])).
% wrd(bis,     pr([akk],[bis])).    % dir: mit 2.Präp s.u. ; bis[adv(temp)]

wrd(dank,      pr([dat,kaus],[dank])).  % inst ?
wrd(diesseits, pr([gen,lok],[diesseits])).
wrd(durch,     pr([akk,lok],[durch])).
wrd(durch,     pr([akk,dir(hin)],[durch])).

wrd(einschließlich, pr([gen,kop],[einschließlich])).
wrd(entlang,   pr([gen,lok],[entlang])).
wrd(entgegen,  pr([dat,modl],[entgegen])).
wrd(exklusive, pr([gen,kop],[exklusive])).

wrd(fern,      pr([gen,lok],[fern])).
wrd(für,       pr([akk,final],[für])).
wrd(für,       pr([akk,modl],[für])).

wrd(gegen,     pr([akk,dir(hin)],[gegen])).
wrd(gegen,     pr([akk,modl],[gegen])).
wrd(gegenüber, pr([dat,lok],[gegenüber])).
wrd(gegenüber, pr([dat,modl],[gegenüber])).
wrd(gemäß,     pr([dat,modl],[gemäß])).
wrd(gemäß,     pr([dat,kaus],[gemäß])).
% gen

wrd(hinter,    pr([dat,lok],[hinter(dat)])).
wrd(hinter,    pr([akk,dir(hin)],[hinter(akk)])).

wrd(in,        pr([dat,lok],[in(dat)])).
wrd(in,        pr([dat,temp],[in(dat)])).
wrd(in,        pr([dat,modl],[in(dat)])).
wrd(in,        pr([akk,dir(hin)],[in(akk)])).
wrd(infolge,   pr([gen,kaus],[infolge])).
wrd(inklusive, pr([gen,kop],[inklusive])).
wrd(inmitten,  pr([gen,lok],[inmitten])).
wrd(innerhalb, pr([gen,lok],[innerhalb])).
wrd(innerhalb, pr([gen,temp],[innerhalb])).
wrd(je,        pr([nom,dis],[je(nom)])).         % Nullartikel!
wrd(je,        pr([akk,dis],[je(akk)])).
wrd(jenseits,  pr([gen,lok],[jenseits])).

wrd(kraft,     pr([gen,kaus],[kraft])).
wrd(längs,     pr([gen,lok],[längs])).
wrd(laut,      pr([gen,modl],[laut(gen)])).
wrd(laut,      pr([dat,modl],[laut(dat)])).

wrd(mangels,   pr([gen,kaus],[mangels])).
```

```
wrd(mit,       pr([dat,inst],[mit])).
wrd(mit,       pr([dat,kop],[mit])).
wrd(mit,       pr([dat,modl],[mit])).
wrd(mit,       pr([dat,temp],[mit])).
wrd(mit,       pr([dat,kond],[mit])).
wrd(mitsamt,   pr([dat,kop],[mitsamt])).
wrd(mittels,   pr([gen,inst],[mittels])).

wrd(nahe,      pr([dat,lok],[nahe])).
wrd(nach,      pr([dat,dir(hin)],[nach])).
wrd(nach,      pr([dat,temp],[nach])).
wrd(nach,      pr([dat,modl],[nach])).
wrd(neben,     pr([dat,lok],[neben])).
wrd(neben,     pr([dat,kop],[neben(dat)])).
wrd(neben,     pr([akk,dir(hin)],[neben(akk)])).
% nebst
% nächst
% namens
wrd(nördlich,  pr([gen,lok],[nördlich])).

% ob
wrd(oberhalb,pr([gen,lok],[oberhalb])).
wrd(ohne,      pr([akk,inst],[ohne])).
wrd(ohne,      pr([akk,kop],[ohne])).
wrd(ohne,      pr([akk,modl],[ohne])).

wrd(per,       pr([akk,dis],[per])).    % Nullartikel, nur Singular!
wrd(pro,       pr([akk,dis],[pro])).    %    "

% qua

wrd(samt,      pr([dat,kop],[samt])).
wrd(seit,      pr([dat,temp],[seit])).
wrd(seitens,   pr([gen,kaus],[seitens])).  % aber: meiner-seits
wrd(seitlich,  pr([gen,lok],[seitlich])).
% seitwärts
% sonder
wrd(statt,     pr([gen,modl],[statt])).
wrd(südlich,   pr([gen,lok],[südlich])).

wrd(trotz,     pr([gen,konz],[trotz])).  % dieselbe Praep mit 2 Kausus
wrd(trotz,     pr([dat,konz],[trotz])).

wrd(über,      pr([dat,lok],[über(dat)])).
wrd(über,      pr([akk,dir(hin)],[über(akk)])).
wrd(um,        pr([akk,lok],[um])).
wrd(um,        pr([akk,temp],[um])).
% unbeschadet
wrd(unfern,    pr([gen,lok],[unfern])). % ??
wrd(ungeachtet,pr([gen,konz],[ungeachtet])).
wrd(unter,     pr([dat,lok],[unter(dat)])).
wrd(unter,     pr([akk,dir(hin)],[unter(akk)])).
wrd(unterhalb, pr([gen,lok],[unterhalb])).
```

```
wrd(unweit,    pr([gen,lok],[unweit])).

wrd(vermittels,pr([gen,inst],[vermittels])).
wrd(vermöge,   pr([gen,inst],[vermöge])).
% vis-a-vis
wrd(von,       pr([dat,dir(her)],[von])).
wrd(von,       pr([dat,modl],[von])).
wrd(vor,       pr([dat,lok],[vor(dat)])).
wrd(vor,       pr([dat,temp],[vor(dat)])).
wrd(vor,       pr([dat,kaus],[vor(dat)])).
wrd(vor,       pr([akk,dir(hin)],[vor(akk)])).

wrd(während,   pr([gen,temp],[während])).
wrd(wegen,     pr([gen,kaus],[wegen])).  % Aber: seinetwegen - wegen ihm (!)
wrd(westlich,  pr([gen,lok],[westlich])).
wrd(wider,     pr([akk,konz],[wider])).
wrd(wie,       pr([Kas,modl],[wie(Kas)])) :- kasus(Kas). % = (so)wie dich/dir ..

wrd(zeit,      pr([gen,temp],[zeit])).
wrd(zu,        pr([dat,dir(hin)],[zu])).
% wrd(zu,        pr([dat,final],[zu])).  % zum Andenken, zur Erinnerung
wrd(zu,        pr([dat,dist],[zu])).     % zu dritt
% wrd(zu,        pr([dat,modl],[zu])).   % mir zur Freude
wrd(zugunsten, pr([gen,final],[zugunsten])).
wrd(zuzüglich, pr([dat,modl],[zuzüglich])). % gen?
wrd(zwecks,    pr([gen,final],[zwecks])).
wrd(zwischen,  pr([dat,lok],[zwischen(dat)])).
wrd(zwischen,  pr([akk,dir(hin)],[zwischen(akk)])).
```

(ii) Pronominaladverbien (`da|hier|wo` + Präposition):[4] Sie werden als Pro-Präpositional*phrasen* kategorisiert und ins Vollformenlexikon der Präpositionen geschrieben. Im Formmerkmal ist nur dann auch der Kasus der Präposition angegeben, wenn er nicht eindeutig bestimmt ist.

⟨*Lexika/praepositionen.wrd*⟩+≡

```
word(darauf,  proPP([def(3,sg)],[auf(dat)])).
word(darauf,  proPP([def(3,sg)],[auf(akk)])).
word(hierauf, proPP([def(3,sg)],[auf(dat)])).
word(hierauf, proPP([def(3,sg)],[auf(akk)])).
word(worauf,  proPP([qu(unsp,sg)],[auf(dat)])).
word(worauf,  proPP([qu(unsp,sg)],[auf(akk)])).
word(worauf,  proPP([rel(neut,sg)],[auf(dat)])).  % nur neut?
word(worauf,  proPP([rel(neut,sg)],[auf(akk)])).

word(daran,   proPP([def(3,sg)],[an(dat)])).
word(daran,   proPP([def(3,sg)],[an(akk)])).
word(hieran,  proPP([def(3,sg)],[an(dat)])).
word(hieran,  proPP([def(3,sg)],[an(akk)])).
word(woran,   proPP([qu(unsp,sg)],[an(dat)])).
word(woran,   proPP([qu(unsp,sg)],[an(akk)])).
word(woran,   proPP([rel(neut,sg)],[an(dat)])).
word(woran,   proPP([rel(neut,sg)],[an(akk)])).
```

[4] Helbig/Buscha: `auf,an,aus,bei,für,mit,nach,vor`. Dazu: `durch,in(dat),(e)in(akk),über,unter,zu`.

```
word(daraus,  proPP([def(3,sg)],[aus])).
word(hieraus, proPP([def(3,sg)],[aus])).
word(woraus,  proPP([qu(unsp,sg)],[aus])).
word(woraus,  proPP([rel(neut,sg)],[aus])).

word(dabei,   proPP([def(3,sg)],[bei])).
word(hierbei, proPP([def(3,sg)],[bei])).
word(wobei,   proPP([qu(unsp,sg)],[bei])).
word(wobei,   proPP([rel(neut,sg)],[bei])).

word(dadurch, proPP([def(3,sg)],[durch])).
word(hierdurch, proPP([def(3,sg)],[durch])).
word(wodurch, proPP([qu(unsp,sg)],[durch])).
word(wodurch, proPP([rel(neut,sg)],[durch])).

word(dafür,   proPP([def(3,sg)],[für])).
word(hierfür, proPP([def(3,sg)],[für])).
word(wofür,   proPP([qu(unsp,sg)],[für])).
word(wofür,   proPP([rel(neut,sg)],[für])).

word(darin,   proPP([def(3,sg)],[in(dat)])).
word(hierin,  proPP([def(3,sg)],[in(dat)])).
word(worin,   proPP([qu(unsp,sg)],[in(dat)])).
word(worin,   proPP([rel(neut,sg)],[in(dat)])).

word(darein,  proPP([def(3,sg)],[in(akk)])).
word(hierein, proPP([def(3,sg)],[in(akk)])).
word(worein,  proPP([qu(unsp,sg)],[in(akk)])).
word(worein,  proPP([rel(neut,sg)],[in(akk)])).

word(damit,   proPP([def(3,sg)],[mit])).
word(hiermit, proPP([def(3,sg)],[mit])).
word(womit,   proPP([qu(unsp,sg)],[mit])).
word(womit,   proPP([rel(neut,sg)],[mit])).

word(danach,  proPP([def(3,sg)],[nach])).
word(wonach,  proPP([qu(unsp,sg)],[nach])).
word(wonach,  proPP([rel(neut,sg)],[nach])).

word(darüber, proPP([def(3,sg)],[über(dat)])).
word(darüber, proPP([def(3,sg)],[über(akk)])).
word(hierüber,proPP([def(3,sg)],[über(dat)])).
word(hierüber,proPP([def(3,sg)],[über(akk)])).
word(worüber, proPP([qu(unsp,sg)],[über(dat)])).
word(worüber, proPP([qu(unsp,sg)],[über(akk)])).
word(worüber, proPP([rel(neut)],[über(dat)])).
word(worüber, proPP([rel(neut)],[über(akk)])).

word(darunter, proPP([def(3,sg)],[unter(dat)])).
word(darunter, proPP([def(3,sg)],[unter(akk)])).  % dir
word(hierunter,proPP([def(3,sg)],[unter(dat)])).
word(worunter, proPP([qu(unsp,sg)],[unter(dat)])).
word(worunter, proPP([rel(neut)],[unter(dat)])).
```

```
word(davon,   proPP([def(3,sg)],[von])).
word(hiervon, proPP([def(3,sg)],[von])).
word(wovon,   proPP([qu(unsp,sg)],[von])).
word(wovon,   proPP([rel(neut)],[von])).

word(davor,   proPP([def(3,sg)],[vor(dat)])).
word(davor,   proPP([def(3,sg)],[vor(akk)])).     % stelle den Eimer davor
word(hiervor, proPP([def(3,sg)],[vor(dat)])).
word(hiervor, proPP([def(3,sg)],[vor(akk)])).
word(wovor,   proPP([qu(unsp,sg)],[vor(dat)])).
word(wovor,   proPP([qu(unsp,sg)],[vor(dat)])).
word(wovor,   proPP([qu(unsp,sg)],[vor(akk)])).
word(wovor,   proPP([rel(neut)],[vor(dat)])).

word(dazu,    proPP([def(3,sg)],[zu])).
word(hierzu,  proPP([def(3,sg)],[zu])).
word(wozu,    proPP([qu(unsp,sg)],[zu])).
word(wozu,    proPP([rel(neut)],[zu])).
```

Bem: Sollte man auch die adverbialen Funktionen, die diese Pronominaladverbien verwirklichen können, als Klassenmerkmal angeben? Dann wird die Analyse aller Präpositionalphrasen zu vieldeutig!

(iii) Konjunktionaladverbien: (Satzglied | satzkoordinierende Konjunktion): [5]

⟨*Lexika/praepositionen.wrd*⟩+≡

```
word(deswegen, proPP([def(3,sg)],[wegen(gen)])).  % auch als adv([def,kaus]). Unnötig.

word(dessentwegen, proPP([rel(mask,sg)],[wegen(gen)])).
word(deretwegen,   proPP([rel(neut,sg)],[wegen(gen)])).
word(dessentwegen, proPP([rel(neut,sg)],[wegen(gen)])).
word(deretwegen,   proPP([rel(unsp,pl)],[wegen(gen)])).

word(weswegen, proPP([qu(unsp,sg)],[wegen(gen)])).
word(weswegen, proPP([rel(kaus)],[wegen(gen)])).  % das liefert auch die rel-Lesart

word(dorthin, proPP([def(3,sg)],   [in(akk)])).         % oder als Direktionaladverbien ??
word(wohin,   proPP([qu(unsp,sg)],   [in(akk)])).
word(wohin,   proPP([rel(dir(hin))],[in(akk)])).  % der Ort, wohin ich fahre, ...

% gehen :v([nom,dir(akk)],[inf]) auf, zu,
% kommen:v([nom,dir(dat)],[inf]) aus,von:dat, in:akk
```

(iv) Abtrennbare Verbpräfixe (CISLEX: VPART), soweit sie auch Präpositionen sind:

⟨*Lexika/praepositionen.wrd*⟩+≡

```
word(ab,      praef([ab],[])).
word(an,      praef([an],[])).
word(auf,     praef([auf],[])).
word(aus,     praef([aus],[])).
word(dar,     praef([dar],[])).
word(ein,     praef([ein],[])).
```

[5] Helbig/Buscha: deshalb, daher, trotzdem, folglich, nämlich, insofern, deswegen, mithin, demnach, sonst, außerdem, allerdings.

```
word(hervor,  praef([hervor],[])).
word(nach,    praef([nach],[])).
word(statt,   praef([statt],[])).       % stattfinden
word(über,    praef([über],[])).
word(um,      praef([um],[])).
word(vor,     praef([vor],[])).
word(weg,     praef([weg],[])).
word(zu,      praef([zu],[])).
```

(v) Präpositionen, die auch satzkomplementierende Elemente sind: *Kategorie??*

⟨*Lexika/praepositionen.wrd*⟩+≡

```
word(ob,      ob).
word(um,      um).
word(zu,      zu).
```

(vi) Postpositionen:

⟨*Lexika/praepositionen.wrd*⟩+≡

```
wrd(entgegen,  po([dat,modl],[entgegen])).  % konz ?
wrd(entlang,   po([akk,dir(_)],[entlang])).
wrd(gegenüber, po([dat,lok],[gegenüber])).
wrd(gegenüber, po([dat,modl],[gegenüber])).
wrd(halber,    po([gen,kaus],[halber])).
wrd(nach,      po([dat,modl],[nach])).
wrd(über,      po([akk,temp],[über])).
% wrd(ungeachtet,po([gen,konz],[ungeachtet])).
wrd(wegen,     po([gen,kaus],[wegen])).     % meinet-wegen
wrd(zufolge,   po([dat,kaus],[zufolge])).
wrd(zufolge,   po([dat,modl],[zufolge])).
wrd(zuliebe,   po([dat,kaus],[zuliebe])).
% wrd(zuwider,   p([dat,modl],[zuwider])).  % adv, praef
```

(vii) Präpositionen mit einem Adverbialargument: *Sehr provisorisch, unfertig!* (Beachte: Bei diesen `word/2`-Einträgen wird in den Kategorien die adverbiale Funktion als Komplement*liste* angegeben.)

⟨*Lexika/praepositionen.wrd*⟩+≡

```
% bis - an|auf|gegen|hinter|in|nach|unter|über|vor|zu

word(bis,           pr([[dir(hin)]],[bis])).  % bis: an(akk) ... zu(dat)
word(bis,           pr([[temp]],[bis])).
word(bis,           pr([[kop]],[bis])).       % bis auf NP ?

word(von,           pr([[dir(her)]],[von])).  % von dort her; von X aus?
word(seit,          pr([[temp]],[seit])).

% je-nach, zu-je, unweit-von
```

Bem. Im jeweiligen `wrd/2`-Stammeintrag fehlt noch die adverbiale Funktion, und das Komplement ist einfacheitshalber nicht als Liste notiert.

(viii) Zweiteilige Präpositionen (Circumpositionen): Aus einem 'Stamm'-Eintrag einer zweiteiligen Präposition werden für die beiden Teile getrennte Einträge ins Vollformenlexikon erzeugt:

⟨*Lexika/praepositionen.wrd*⟩+≡

```
word(Prae,          pc1([[Kas]],[Praep])) :-
        Praep = (Prae,Post),
        wrd(Praep, pc([Kas,_AdvF],[Praep])).
word(Post,          pc2([[Kas]],[Praep])) :-
        Praep = (Prae,Post),
        wrd(Praep, pc([Kas,_AdvF],[Praep])).

wrd((um,'Willen'),  pc([gen,kaus],[(um,'Willen')])).
wrd((an,'Statt'),   pc([gen,modl],[(an,'Statt')])).

wrd((auf,hinauf),   pc([akk,dir],[(auf,hinauf)])).
wrd((in,hinein),    pc([akk,dir],[(in,hinein)])).
wrd((um,herum),     pc([akk,dir],[(um,herum)])).
wrd((über,hinaus),  pc([akk,dir],[(über,hinaus)])).

wrd((von,ab),       pc([dat,temp],[(von,an)])).
wrd((von,ab),       pc([dat,lok], [(von,ab)])).
wrd((von,an),       pc([dat,temp],[(von,an)])).
wrd((von,an),       pc([dat,lok],[(von,an)])).
wrd((von,aus),      pc([dat,lok],[(von,aus)])).
```

Bem. Wird die Adverbiale Funktion solcher PPs richtig berechnent??

(ix) Präpositionen, die mit einem bestimmten Artikel verschmolzen sind: als Klassenparameter braucht man neben dem Kasus auch das Genus der möglichen Objekte; die adverbiale Funktion, die die Phrase realisieren kann, tragen wir nur im 'Stamm'-Eintrag ein:

⟨*Lexika/praepositionen.wrd*⟩+≡

```
word(PraepDet, pdet([[Kas],Gen],[Praep])) :-
        wrd(PraepDet, pdet([[Kas],_AdvF,Gen],[Praep])).

wrd(am,   pdet([[dat],lok,mask],[an(dat)])).
wrd(am,   pdet([[dat],lok,neut],[an(dat)])).
wrd(ans,  pdet([[akk],lok,neut],[an(akk)])).
wrd(aufs, pdet([[akk],dir,neut],[auf(akk)])).
wrd(beim, pdet([[dat],lok,mask],[bei])).
wrd(beim, pdet([[dat],lok,neut],[bei])).
wrd(fürs, pdet([[akk],modl,neut],[für])). % modl?
wrd(im,   pdet([[dat],lok,mask],[in(dat)])).
wrd(im,   pdet([[dat],lok,neut],[in(dat)])).
wrd(im,   pdet([[dat],temp,mask],[in(dat)])).
wrd(ins,  pdet([[akk],dir,neut],[in(akk)])).
wrd(vom,  pdet([[dat],dir,mask],[von])).
wrd(vom,  pdet([[dat],dir,neut],[von])).
wrd(zum,  pdet([[dat],dir,mask],[zu])).
wrd(zum,  pdet([[dat],dir,neut],[zu])).
wrd(zur,  pdet([[dat],dir,fem ],[zu])).
```

Bem.: Die Abkürzung des Komplements durch den Kasus ist hier ungenau: die Grammatikregeln müssen dafür sorgen, daß nicht beliebige Nominalphrasen, sondern nur artikellose Nominalausdrücke im Singular als Komplemente verwendet werden können.

Ein Hilfslexikon für eingebettete Nominalphrasen:

(i) Als Artikel nehmen wir alle Formen des bestimmten Artikels im Maskulinum:

⟨*Lexika/praepositionen.hilfslexikon.wrd*⟩≡

```
word(der,det([def,sw],[mask,sg,nom])).
word(des,det([def,sw],[mask,sg,gen])).
word(dem,det([def,sw],[mask,sg,dat])).
word(den,det([def,sw],[mask,sg,akk])).

word(die,det([def,sw],[mask,pl,nom])).
word(der,det([def,sw],[mask,pl,gen])).
word(den,det([def,sw],[mask,pl,dat])).
word(die,det([def,sw],[mask,pl,akk])).
```

Zum Testen seien noch ein paar andere Determinatoren im Lexikon:

⟨*Lexika/praepositionen.hilfslexikon.wrd*⟩+≡

```
word(ein,det([indef,ge],[mask,sg,nom])).
word(eines,det([indef,ge],[mask,sg,gen])).
word(einem,det([indef,ge],[mask,sg,dat])).
word(einen,det([indef,ge],[mask,sg,akk])).

word(welcher,det([qu],[mask,sg,nom])).
word(welchem,det([qu],[mask,sg,dat])).

word(eine,det([indef,ge],[fem,sg,nom])).
word(einer,det([indef,ge],[fem,sg,gen])).
word(einer,det([indef,ge],[fem,sg,dat])).
word(eine,det([indef,ge],[fem,sg,akk])).
```

(ii) Als Nomen nehmen wir nur ein Wort in allen Fällen:

⟨*Lexika/praepositionen.hilfslexikon.wrd*⟩+≡

```
word('Hamster',n([mask],[sg,nom])).
word('Hamsters',n([mask],[sg,gen])).
word('Hamster',n([mask],[sg,dat])).
word('Hamster',n([mask],[sg,akk])).

word('Hamster',n([mask],[pl,nom])).
word('Hamster',n([mask],[pl,gen])).
word('Hamstern',n([mask],[pl,dat])).
word('Hamster',n([mask],[pl,akk])).
```

(iii) Um relativierende und interrogative Präpositionalphrasen testen zu können, geben wir zwei Pseudo-Nominalphrasen vor:

⟨*Lexika/praepositionen.hilfslexikon.wrd*⟩+≡

```
word(dem, np([rel,3,mask],[sg,dat])).
word(wem, np([qu,3,mask],[sg,dat])).
```

(iv) Um Päpositionalphrasen, die eine Adverbialphrase enthalten, testen zu können, seien noch zwei Adverbien angegeben:

⟨*Lexika/praepositionen.hilfslexikon.wrd*⟩+≡

```
word(dann, adv([def,temp],[])).
word(wann, adv([qu,temp],[])).
```

Aufgabe 1.1 Berechne aus einer Grammatik mit zweistelligen Prädikaten `Kategorie(Klasse,Form)` die in der Grammatik verwendeten lexikalischen Kategorien und ihre Merkmalwerte. Mache das gleiche für die in einem Vollformenlexikon `word(Vollform,Kategorie)` (oder in einem Teillexikon `lexword/2` verwendeten Kategorien.

Erzeuge damit ein Testprogramm, das überprüft, ob eine Grammatik und ein (Teil-) Lexikon zueinander passen, d.h. ob im Lexikon nur Wortkategorien verwendet werden, die auch in der Grammatik auftreten.

Ein Parser sollte mit einem solchen Programm prüfen, ob die verwendete Grammatik alle von der lexikalischen Analyse genannten lexikalischen Kategorien kennt; wenn nicht, sollte der Parser gar keine Analyse versuchen, sondern eine Fehlermeldung ausgeben.

1.2.5 Determinatoren

Die Determinatoren bilden eigentlich (wie die Adverbiale) eine Sammelkategorie, sind also nicht aus einer Wortart durch Erweiterungen gebildete Ausdrücke. Es sind vielmehr Ausdrücke verschiedener Wortarten, die „nur" deshalb zusammengefaßt werden, weil sie *auch* die syntaktische Funktion eines Determinators einer Nominalphrase übernehmen können.

Aus diesem Grund ist es nicht verwunderlich, daß nicht alle Wörter darin in genau denselben syntaktischen Funktionen auftreten. (In manchen Fällen erfolgt die Flexion eines Worts in Abhängigkeit von der Verwendung als Determinator oder Pronomen). Üblicherweise unterscheidet man z.B. (grob, mit definiten, interrogativen, relativen Versionen) die Wortarten:

(i) Demonstrativum: `der, dieser, jener, welcher, ...`

(ii) Possessivum: `mein, sein, eure, jemandes, niemandes, dessen, wessen, ..`

(iii) Quantoren: `viele, wenige, alle, wie viele, wieviel, jeder, kein,...`

Wir beschreiben die Unterklassen von Determinatoren aber wie bei den eigentlichen Wortklassen durch Klassenparameter, nicht durch eigene Kategoriennamen.

⟨*Wortkategorien*⟩+≡
```
wortkategorie(det([Def,Adekl],[Gen,Num,Kas])) :-
        definitheit_det(Def),
        adjektivdeklination(Adekl),
        genusErw(Gen),
        numerus(Num),
        kasus(Kas).
```

Als Unterklassen unterscheiden wir in definite, indefinite, possessive, quantifizierende, relativierende und interrogative Determinatoren, sowie danach, welche Adjektivdeklination der Determinator regiert:

⟨*Lexikalische Merkmale*⟩+≡
```
definitheit_det(def).
definitheit_det(indef).
definitheit_det(poss).
definitheit_det(quant).
definitheit_det(rel).
definitheit_det(qu).
```

Bei den Formmerkmalen sind Genus, Numerus und Kasus zu berücksichtigen.

Die folgende Datei von Determinatoren aus dem Beispieltext `Testbett/wissenschaftsreligion.txt` enthält Einträge der Form `word(Determiner,Kategorie,Stamm)`, worin der Stamm für die Flexion gedacht ist. (Die Datei enthält keine Lesarten als Relativpronomen.)

⟨*Lexika/determiner.wrd*⟩≡

```
word(alle, det([quant,sw], [unsp, pl, nom]), alle).
word(alle, det([quant,sw], [unsp, pl, akk]), alle).
word(alle, det([quant,sw], [fem, sg, akk]), alle).
word(alle, det([quant,sw], [fem, sg, nom]), alle).
word(allem, det([quant,sw], [mask, sg, dat]), alle).
word(allem, det([quant,sw], [neut, sg, dat]), alle).
word(allen, det([quant,sw], [unsp, pl, dat]), alle).
word(allen, det([quant,sw], [mask, sg, gen]), alle).
word(allen, det([quant,sw], [neut, sg, gen]), alle).
word(allen, det([quant,sw], [mask, sg, akk]), alle).
word(aller, det([quant,sw], [unsp, pl, gen]), alle).
word(aller, det([quant,sw], [mask, sg, nom]), alle).
word(aller, det([quant,sw], [fem, sg, gen]), alle).
word(aller, det([quant,sw], [fem, sg, dat]), alle).
word(alles, det([quant,sw], [neut, sg, nom]), alle).
word(alles, det([quant,sw], [neut, sg, akk]), alle).
word(beide, det([quant,st], [unsp, pl, akk]), beid).
word(beide, det([quant,st], [unsp, pl, nom]), beid).
word(beiden, det([quant,st], [unsp, pl, dat]), beid).
word(beider, det([quant,st], [unsp, pl, gen]), beid).
word(das, det([def,sw], [neut, sg, nom]), d).
word(das, det([def,sw], [neut, sg, akk]), d).
word(dasjenige, det([def,sw], [neut, sg, nom]), d_jenig).
word(dasjenige, det([def,sw], [neut, sg, akk]), d_jenig).
word(dasselbe, det([def,sw], [neut, sg, nom]), d_selb).
word(dasselbe, det([def,sw], [neut, sg, akk]), d_selb).
word(dem, det([def,sw], [mask, sg, dat]), d).
word(dem, det([def,sw], [neut, sg, dat]), d).
word(demjenigen, det([def,sw], [mask, sg, dat]), d_jenig).
word(demjenigen, det([def,sw], [neut, sg, dat]), d_jenig).
word(demselben, det([def,sw], [mask, sg, dat]), d_selb).
word(demselben, det([def,sw], [neut, sg, dat]), d_selb).
word(den, det([def,sw], [unsp, pl, dat]), d).
word(den, det([def,sw], [mask, sg, akk]), d).
word(denjenigen, det([def,sw], [unsp, pl, dat]), d_jenig).
word(denjenigen, det([def,sw], [mask, sg, akk]), d_jenig).
word(denselben, det([def,sw], [unsp, pl, dat]), d_selb).
word(denselben, det([def,sw], [mask, sg, akk]), d_selb).
word(der, det([def,sw], [unsp, pl, gen]), d).
word(der, det([def,sw], [mask, sg, nom]), d).
word(der, det([def,sw], [fem, sg, gen]), d).
word(der, det([def,sw], [fem, sg, dat]), d).
word(deren, det([poss,sw], [unsp, pl, gen]), d).
word(deren, det([poss,sw], [fem, sg, gen]), d).
word(derjenige, det([def,sw], [mask, sg, nom]), d_jenig).
word(derjenigen, det([def,sw], [unsp, pl, gen]), d_jenig).
word(derjenigen, det([def,sw], [fem, sg, gen]), d_jenig).
```

```
word(derjenigen, det([def,sw], [fem, sg, dat]), d_jenig).
word(derselbe, det([def,sw], [mask, sg, nom]), d_selb).
word(derselben, det([def,sw], [unsp, pl, gen]), d_selb).
word(derselben, det([def,sw], [fem, sg, gen]), d_selb).
word(derselben, det([def,sw], [fem, sg, dat]), d_selb).
word(des, det([def,sw], [mask, sg, gen]), d).
word(des, det([def,sw], [neut, sg, gen]), d).
word(desjenigen, det([def,sw], [mask, sg, gen]), d_jenig).
word(desjenigen, det([def,sw], [neut, sg, gen]), d_jenig).
word(desselben, det([def,sw], [mask, sg, gen]), d_selb).
word(desselben, det([def,sw], [neut, sg, gen]), d_selb).
word(dessen, det([rel,sw], [mask, sg, gen]), d).
word(dessen, det([rel,sw], [neut, sg, gen]), d).
word(dein, det([poss,ge], [neut, sg, nom]), dein).
word(dein, det([poss,ge], [neut, sg, akk]), dein).
word(dein, det([poss,ge], [mask, sg, nom]), dein).
word(deine, det([poss,ge], [unsp, pl, akk]), dein).
word(deine, det([poss,ge], [unsp, pl, nom]), dein).
word(deine, det([poss,ge], [fem, sg, akk]), dein).
word(deine, det([poss,ge], [fem, sg, nom]), dein).
word(deinem, det([poss,ge], [mask, sg, dat]), dein).
word(deinem, det([poss,ge], [neut, sg, dat]), dein).
word(deinen, det([poss,ge], [unsp, pl, dat]), dein).
word(deinen, det([poss,ge], [mask, sg, akk]), dein).
word(deiner, det([poss,ge], [unsp, pl, gen]), dein).
word(deiner, det([poss,ge], [fem, sg, gen]), dein).
word(deiner, det([poss,ge], [fem, sg, dat]), dein).
word(deines, det([poss,ge], [neut, sg, gen]), dein).
word(deines, det([poss,ge], [mask, sg, gen]), dein).
word(die, det([def,sw], [unsp, pl, nom]), d).
word(die, det([def,sw], [unsp, pl, akk]), d).
word(die, det([def,sw], [fem, sg, akk]), d).
word(die, det([def,sw], [fem, sg, nom]), d).
word(diejenige, det([def,sw], [fem, sg, akk]), d_jenig).
word(diejenige, det([def,sw], [fem, sg, nom]), d_jenig).
word(diejenigen, det([def,sw], [unsp, pl, nom]), d_jenig).
word(diejenigen, det([def,sw], [unsp, pl, akk]), d_jenig).
word(diejenigen, det([def,sw], [unsp, pl, dat]), d_jenig).
word(diese, det([def,sw], [unsp, pl, nom]), dies).
word(diese, det([def,sw], [unsp, pl, akk]), dies).
word(diese, det([def,sw], [fem, sg, akk]), dies).
word(diese, det([def,sw], [fem, sg, nom]), dies).
word(dieselbe, det([def,sw], [fem, sg, akk]), d_selb).
word(dieselbe, det([def,sw], [fem, sg, nom]), d_selb).
word(dieselben, det([def,sw], [unsp, pl, nom]), d_selb).
word(dieselben, det([def,sw], [unsp, pl, akk]), d_selb).
word(dieselben, det([def,sw], [unsp, pl, dat]), d_selb).
word(diesem, det([def,sw], [mask, sg, dat]), dies).
word(diesem, det([def,sw], [neut, sg, dat]), dies).
word(diesen, det([def,sw], [unsp, pl, dat]), dies).
word(diesen, det([def,sw], [mask, sg, akk]), dies).
word(dieser, det([def,sw], [unsp, pl, gen]), dies).
word(dieser, det([def,sw], [mask, sg, nom]), dies).
```

```
word(dieser, det([def,sw], [fem, sg, gen]), dies).
word(dieser, det([def,sw], [fem, sg, dat]), dies).
word(dieses, det([def,sw], [neut, sg, nom]), dies).
word(dieses, det([def,sw], [neut, sg, akk]), dies).
word(dieses, det([def,sw], [mask, sg, gen]), dies).
word(dieses, det([def,sw], [neut, sg, gen]), dies).
word(ein, det([indef,ge], [neut, sg, nom]), ein).
word(ein, det([indef,ge], [neut, sg, akk]), ein).
word(ein, det([indef,ge], [mask, sg, nom]), ein).
word(eine, det([indef,ge], [fem, sg, akk]), ein).
word(eine, det([indef,ge], [fem, sg, nom]), ein).
word(einem, det([indef,ge], [mask, sg, dat]), ein).
word(einem, det([indef,ge], [neut, sg, dat]), ein).
word(einen, det([indef,ge], [mask, sg, akk]), ein).
word(einer, det([indef,ge], [fem, sg, gen]), ein).
word(einer, det([indef,ge], [fem, sg, dat]), ein).
word(eines, det([indef,ge], [neut, sg, gen]), ein).
word(eines, det([indef,ge], [mask, sg, gen]), ein).
word(einige, det([indef,st], [unsp, pl, nom]), einig).
word(einige, det([indef,st], [unsp, pl, akk]), einig).
word(einige, det([indef,st], [fem, sg, akk]), einig).
word(einige, det([indef,st], [fem, sg, nom]), einig).
word(einigem, det([indef,st], [mask, sg, dat]), einig).
word(einigem, det([indef,st], [neut, sg, dat]), einig).
word(einigen, det([indef,st], [unsp, pl, dat]), einig).
word(einigen, det([indef,st], [mask, sg, gen]), einig).
word(einigen, det([indef,st], [neut, sg, gen]), einig).
word(einigen, det([indef,st], [mask, sg, akk]), einig).
word(einiger, det([indef,st], [unsp, pl, gen]), einig).
word(einiger, det([indef,st], [mask, sg, nom]), einig).
word(einiger, det([indef,st], [fem, sg, gen]), einig).
word(einiger, det([indef,st], [fem, sg, dat]), einig).
word(einiges, det([indef,st], [neut, sg, nom]), einig).
word(einiges, det([indef,st], [neut, sg, akk]), einig).
word(euer, det([poss,ge], [neut, sg, nom]), euer).
word(euer, det([poss,ge], [neut, sg, akk]), euer).
word(euer, det([poss,ge], [mask, sg, nom]), euer).
word(euere, det([poss,ge], [unsp, pl, akk]), euer).
word(euere, det([poss,ge], [unsp, pl, nom]), euer).
word(euere, det([poss,ge], [fem, sg, akk]), euer).
word(euere, det([poss,ge], [fem, sg, nom]), euer).
word(euerem, det([poss,ge], [mask, sg, dat]), euer).
word(euerem, det([poss,ge], [neut, sg, dat]), euer).
word(eueren, det([poss,ge], [unsp, pl, dat]), euer).
word(eueren, det([poss,ge], [mask, sg, akk]), euer).
word(euerer, det([poss,ge], [unsp, pl, gen]), euer).
word(euerer, det([poss,ge], [fem, sg, gen]), euer).
word(euerer, det([poss,ge], [fem, sg, dat]), euer).
word(eueres, det([poss,ge], [neut, sg, gen]), euer).
word(eueres, det([poss,ge], [mask, sg, gen]), euer).
word(euern, det([poss,ge], [unsp, pl, dat]), euer).
word(euern, det([poss,ge], [mask, sg, akk]), euer).
word(eure, det([poss,ge], [unsp, pl, akk]), euer).
```

```
word(eure, det([poss,ge], [unsp, pl, nom]), euer).
word(eure, det([poss,ge], [fem, sg, akk]), euer).
word(eure, det([poss,ge], [fem, sg, nom]), euer).
word(eurem, det([poss,ge], [mask, sg, dat]), euer).
word(eurem, det([poss,ge], [neut, sg, dat]), euer).
word(euren, det([poss,ge], [unsp, pl, dat]), euer).
word(euren, det([poss,ge], [mask, sg, akk]), euer).
word(eurer, det([poss,ge], [unsp, pl, gen]), euer).
word(eurer, det([poss,ge], [fem, sg, gen]), euer).
word(eurer, det([poss,ge], [fem, sg, dat]), euer).
word(eures, det([poss,ge], [neut, sg, gen]), euer).
word(eures, det([poss,ge], [mask, sg, gen]), euer).
word(ihr, det([poss,ge], [neut, sg, nom]), ihr).
word(ihr, det([poss,ge], [neut, sg, akk]), ihr).
word(ihr, det([poss,ge], [mask, sg, nom]), ihr).
word(ihre, det([poss,ge], [unsp, pl, akk]), ihr).
word(ihre, det([poss,ge], [unsp, pl, nom]), ihr).
word(ihre, det([poss,ge], [fem, sg, akk]), ihr).
word(ihre, det([poss,ge], [fem, sg, nom]), ihr).
word(ihrem, det([poss,ge], [mask, sg, dat]), ihr).
word(ihrem, det([poss,ge], [neut, sg, dat]), ihr).
word(ihren, det([poss,ge], [unsp, pl, dat]), ihr).
word(ihren, det([poss,ge], [mask, sg, akk]), ihr).
word(ihrer, det([poss,ge], [unsp, pl, gen]), ihr).
word(ihrer, det([poss,ge], [fem, sg, gen]), ihr).
word(ihrer, det([poss,ge], [fem, sg, dat]), ihr).
word(ihres, det([poss,ge], [neut, sg, gen]), ihr).
word(ihres, det([poss,ge], [mask, sg, gen]), ihr).
word(jede, det([quant,sw], [fem, sg, akk]), jed).
word(jede, det([quant,sw], [fem, sg, nom]), jed).
word(jedem, det([quant,sw], [mask, sg, dat]), jed).
word(jedem, det([quant,sw], [neut, sg, dat]), jed).
word(jeden, det([quant,sw], [mask, sg, akk]), jed).
word(jeder, det([quant,sw], [mask, sg, nom]), jed).
word(jeder, det([quant,sw], [fem, sg, gen]), jed).
word(jeder, det([quant,sw], [fem, sg, dat]), jed).
word(jedes, det([quant,sw], [neut, sg, nom]), jed).
word(jedes, det([quant,sw], [neut, sg, akk]), jed).
word(jedes, det([quant,sw], [mask, sg, gen]), jed).
word(jedes, det([quant,sw], [neut, sg, gen]), jed).
word(jene, det([def,sw], [unsp, pl, nom]), jen).
word(jene, det([def,sw], [unsp, pl, akk]), jen).
word(jene, det([def,sw], [fem, sg, akk]), jen).
word(jene, det([def,sw], [fem, sg, nom]), jen).
word(jenem, det([def,sw], [mask, sg, dat]), jen).
word(jenem, det([def,sw], [neut, sg, dat]), jen).
word(jenen, det([def,sw], [unsp, pl, dat]), jen).
word(jenen, det([def,sw], [mask, sg, akk]), jen).
word(jener, det([def,sw], [unsp, pl, gen]), jen).
word(jener, det([def,sw], [mask, sg, nom]), jen).
word(jener, det([def,sw], [fem, sg, gen]), jen).
word(jener, det([def,sw], [fem, sg, dat]), jen).
word(jenes, det([def,sw], [neut, sg, nom]), jen).
```

```
word(jenes, det([def,sw], [neut, sg, akk]), jen).
word(jenes, det([def,sw], [mask, sg, gen]), jen).
word(jenes, det([def,sw], [neut, sg, gen]), jen).
word(kein, det([quant,ge], [neut, sg, nom]), kein).
word(kein, det([quant,ge], [neut, sg, akk]), kein).
word(kein, det([quant,ge], [mask, sg, nom]), kein).
word(keine, det([quant,ge], [unsp, pl, akk]), kein).
word(keine, det([quant,ge], [unsp, pl, nom]), kein).
word(keine, det([quant,ge], [fem, sg, akk]), kein).
word(keine, det([quant,ge], [fem, sg, nom]), kein).
word(keinem, det([quant,ge], [mask, sg, dat]), kein).
word(keinem, det([quant,ge], [neut, sg, dat]), kein).
word(keinen, det([quant,ge], [unsp, pl, dat]), kein).
word(keinen, det([quant,ge], [mask, sg, akk]), kein).
word(keiner, det([quant,ge], [unsp, pl, gen]), kein).
word(keiner, det([quant,ge], [fem, sg, gen]), kein).
word(keiner, det([quant,ge], [fem, sg, dat]), kein).
word(keines, det([quant,ge], [neut, sg, gen]), kein).
word(keines, det([quant,ge], [mask, sg, gen]), kein).
word(mein, det([poss,ge], [neut, sg, nom]), mein).
word(mein, det([poss,ge], [neut, sg, akk]), mein).
word(mein, det([poss,ge], [mask, sg, nom]), mein).
word(meine, det([poss,ge], [unsp, pl, akk]), mein).
word(meine, det([poss,ge], [unsp, pl, nom]), mein).
word(meine, det([poss,ge], [fem, sg, akk]), mein).
word(meine, det([poss,ge], [fem, sg, nom]), mein).
word(meinem, det([poss,ge], [mask, sg, dat]), mein).
word(meinem, det([poss,ge], [neut, sg, dat]), mein).
word(meinen, det([poss,ge], [unsp, pl, dat]), mein).
word(meinen, det([poss,ge], [mask, sg, akk]), mein).
word(meiner, det([poss,ge], [unsp, pl, gen]), mein).
word(meiner, det([poss,ge], [fem, sg, gen]), mein).
word(meiner, det([poss,ge], [fem, sg, dat]), mein).
word(meines, det([poss,ge], [neut, sg, gen]), mein).
word(meines, det([poss,ge], [mask, sg, gen]), mein).
word(sein, det([poss,ge], [neut, sg, nom]), sein).
word(sein, det([poss,ge], [neut, sg, akk]), sein).
word(sein, det([poss,ge], [mask, sg, nom]), sein).
word(seine, det([poss,ge], [unsp, pl, akk]), sein).
word(seine, det([poss,ge], [unsp, pl, nom]), sein).
word(seine, det([poss,ge], [fem, sg, akk]), sein).
word(seine, det([poss,ge], [fem, sg, nom]), sein).
word(seinem, det([poss,ge], [mask, sg, dat]), sein).
word(seinem, det([poss,ge], [neut, sg, dat]), sein).
word(seinen, det([poss,ge], [unsp, pl, dat]), sein).
word(seinen, det([poss,ge], [mask, sg, akk]), sein).
word(seiner, det([poss,ge], [unsp, pl, gen]), sein).
word(seiner, det([poss,ge], [fem, sg, gen]), sein).
word(seiner, det([poss,ge], [fem, sg, dat]), sein).
word(seines, det([poss,ge], [neut, sg, gen]), sein).
word(seines, det([poss,ge], [mask, sg, gen]), sein).
word(solche, det([quant,sw], [unsp, pl, nom]), solch).
word(solche, det([quant,sw], [unsp, pl, akk]), solch).
```

```
word(solche, det([quant,sw], [fem, sg, akk]), solch).
word(solche, det([quant,sw], [fem, sg, dat]), solch).
word(solchem, det([quant,sw], [mask, sg, dat]), solch).
word(solchem, det([quant,sw], [neut, sg, dat]), solch).
word(solchen, det([quant,sw], [unsp, pl, dat]), solch).
word(solchen, det([quant,sw], [mask, sg, gen]), solch).
word(solchen, det([quant,sw], [neut, sg, gen]), solch).
word(solchen, det([quant,sw], [mask, sg, akk]), solch).
word(solcher, det([quant,sw], [unsp, pl, gen]), solch).
word(solcher, det([quant,sw], [mask, sg, nom]), solch).
word(solcher, det([quant,sw], [fem, sg, gen]), solch).
word(solcher, det([quant,sw], [fem, sg, dat]), solch).
word(solches, det([quant,sw], [neut, sg, nom]), solch).
word(solches, det([quant,sw], [neut, sg, akk]), solch).
word(unser, det([poss,ge], [neut, sg, nom]), unser).
word(unser, det([poss,ge], [neut, sg, akk]), unser).
word(unser, det([poss,ge], [mask, sg, nom]), unser).
word(unsere, det([poss,ge], [unsp, pl, akk]), unser).
word(unsere, det([poss,ge], [unsp, pl, nom]), unser).
word(unsere, det([poss,ge], [fem, sg, akk]), unser).
word(unsere, det([poss,ge], [fem, sg, nom]), unser).
word(unserem, det([poss,ge], [mask, sg, dat]), unser).
word(unserem, det([poss,ge], [neut, sg, dat]), unser).
word(unseren, det([poss,ge], [unsp, pl, dat]), unser).
word(unseren, det([poss,ge], [mask, sg, akk]), unser).
word(unserer, det([poss,ge], [unsp, pl, gen]), unser).
word(unserer, det([poss,ge], [fem, sg, gen]), unser).
word(unserer, det([poss,ge], [fem, sg, dat]), unser).
word(unseres, det([poss,ge], [neut, sg, gen]), unser).
word(unseres, det([poss,ge], [mask, sg, gen]), unser).
word(unsern, det([poss,ge], [unsp, pl, dat]), unser).
word(unsern, det([poss,ge], [mask, sg, akk]), unser).
word(unsre, det([poss,ge], [unsp, pl, akk]), unser).
word(unsre, det([poss,ge], [unsp, pl, nom]), unser).
word(unsre, det([poss,ge], [fem, sg, akk]), unser).
word(unsre, det([poss,ge], [fem, sg, nom]), unser).
word(unsrem, det([poss,ge], [mask, sg, dat]), unser).
word(unsrem, det([poss,ge], [neut, sg, dat]), unser).
word(unsren, det([poss,ge], [unsp, pl, dat]), unser).
word(unsren, det([poss,ge], [mask, sg, akk]), unser).
word(unsrer, det([poss,ge], [unsp, pl, gen]), unser).
word(unsrer, det([poss,ge], [fem, sg, gen]), unser).
word(unsrer, det([poss,ge], [fem, sg, dat]), unser).
word(unsres, det([poss,ge], [neut, sg, gen]), unser).
word(unsres, det([poss,ge], [mask, sg, gen]), unser).
word(welche, det([qu,sw], [unsp, pl, nom]), welch).
word(welche, det([qu,sw], [unsp, pl, akk]), welch).
word(welche, det([qu,sw], [fem, sg, akk]), welch).
word(welche, det([qu,sw], [fem, sg, nom]), welch).
word(welchem, det([qu,sw], [mask, sg, dat]), welch).
word(welchem, det([qu,sw], [neut, sg, dat]), welch).
word(welchen, det([qu,sw], [unsp, pl, dat]), welch).
word(welchen, det([qu,sw], [mask, sg, akk]), welch).
```

```
word(welcher, det([qu,sw], [unsp, pl, gen]), welch).
word(welcher, det([qu,sw], [mask, sg, nom]), welch).
word(welcher, det([qu,sw], [fem, sg, gen]), welch).
word(welcher, det([qu,sw], [fem, sg, dat]), welch).
word(welches, det([qu,sw], [neut, sg, nom]), welch).
word(welches, det([qu,sw], [neut, sg, akk]), welch).
word(welches, det([qu,sw], [mask, sg, gen]), welch).
word(welches, det([qu,sw], [neut, sg, gen]), welch).
```

Bem. Die aus dem CISLEX ermittelten Einträge weichen allerdings hiervon ab, z.B.:

⟨*CISLEX-Kodierung (unbenutzt)*⟩≡

```
word(alle, det([sw], [unsp, pl, nom])).
word(einige, det([st], [fem, sg, akk])).
word(viele, a([], [pos, ge, fem, sg, akk])).
```

Diese Einträge enthalten nicht alle Informationen, die wir für die Syntaxanalyse brauchen.

1.2.6 Junktoren

(i) Einstellige Satzjunktoren: Sie werden nach der adverbialen Funktion des eingebetteten Satzes klassifiziert. Daß sie alle die Verbletztstellung regieren, können wir in der Grammatikregel sichern und geben es deshalb nicht als Klassenparameter an.

⟨*Wortkategorien*⟩+≡

```
% wortkategorie(subj([Art])) :-
%       adverbialeFunktion(Art).
% restrict(subj([Art]),subj(_)).

wortkategorie(subj([Art],[Junktor])) :-
        adverbialeFunktion(Art).
restrict(subj([Art],[Junktor]),subj([Art],[Junktor])) :- !.
```

Als Formparameter geben wir den Junktor an, da er zur Behandlung von Adverbial- und Aussagekorrelaten sowie zur Rektion von `daß`- oder `ob`-Sätzen im Kategorienausdruck nötig ist.

Zwei Spezialfälle ohne adverbiale Funktion geben wir explizit an:

⟨*Lexika/konjunktionen.wrd*⟩≡

```
word(daß,  subj([-],[daß])).
word(dass, subj([-],[daß])).
word(ob,   subj([-],[ob])).
```

Bei den anderen Subjunktionen ergänzen wir das Formmerkmal wieder aus einfacheren Stammeinträgen:

⟨*Lexika/konjunktionen.wrd*⟩+≡

```
word(W, subj([AdvF],[W])) :-
    wrd(W, subj([AdvF])).
```

(a) kausale:

⟨*Lexika/konjunktionen.wrd*⟩+≡

```
wrd(da,      subj([kaus])).
wrd(weil,    subj([kaus])). % Korrelat: darum, deswegen, deshalb
```

(b) konsekutive:[6]

⟨*Lexika/konjunktionen.wrd*⟩+≡
```
wrd(sodaß,   subj([kons])).
wrd(sodass,  subj([kons])).

word(so,      subj([kons,[daß]],[(so,daß)])).
word(so,      subj([kons,[dass]],[(so,dass)])).
```

(c) konzessive:

⟨*Lexika/konjunktionen.wrd*⟩+≡
```
wrd(obgleich,   subj([konz])).
wrd(obwohl,     subj([konz])).
wrd(obschon,    subj([konz])).
wrd(wenngleich, subj([konz])).
```

(d) temporale: Man könnte sie feiner einteilen:
Temporale Junktoren, bei denen das Ereignis des untergeordneten Satzes *vor* dem des Matrixsatzes stattfindet:

⟨*Lexika/konjunktionen.wrd*⟩+≡
```
wrd(nachdem, subj([temp])).
wrd(seit,    subj([temp])).
wrd(seitdem, subj([temp])).
```

Temporale Junktoren, bei denen das Ereignis des untergeordneten Satzes *nach* dem des Matrixsatzes stattfindet:

⟨*Lexika/konjunktionen.wrd*⟩+≡
```
wrd(bis,     subj([temp])).  % Korrelat: solange (?)
wrd(bevor,   subj([temp])).
wrd(ehe,     subj([temp])).
```

Temporale Junktoren, bei denen das Ereignis des untergeordneten Satzes *gleichzeitig* mit dem des Matrixsatzes stattfindet:

⟨*Lexika/konjunktionen.wrd*⟩+≡
```
wrd(als,     subj([temp])).  % Korrelat: da
wrd(indem,   subj([temp])).
wrd(während, subj([temp])).  % Korrelat: da
wrd(wenn,    subj([temp])).
wrd(sobald,  subj([temp])).
wrd(sowie,   subj([temp])).
wrd(solange, subj([temp])).
wrd(sooft,   subj([temp])).
```

Bem. Ein Satz der Form `eher s1(_,[...,erst]), als daß s2(_,[...,letzt])` ist synonym zum entsprechenden Satz der Form `bevor s1(_,[...,letzt]), s2(_,[...,erst])`.

(e) konditionale:

⟨*Lexika/konjunktionen.wrd*⟩+≡
```
wrd(wenn,    subj([kond])).
wrd(falls,   subj([kond])).
```

(f) finale:

⟨*Lexika/konjunktionen.wrd*⟩+≡
```
wrd(damit,   subj([final])).
word(um,     subj([final,[infzu]],[um])).
```

[6] Diese adverbiale Funktion ist unter `adverbialeFunktion/1` (noch?) nicht erfaßt.

(g) instrumentale:

⟨*Lexika/konjunktionen.wrd*⟩+≡
```
wrd(indem,   subj([inst])).
```

(ii) Zweistellige Satzjunktoren, darunter *eingliedrige*, bei denen der Junktor als Formmerkmal (eigentlich mehr wegen der Baumanzeige) und die Verbstellungen der Teilsätze als Klassenmerkmal (Rahmen) notiert werden:

⟨*Wortkategorien*⟩+≡
```
wortkategorie(conj([Vst1,Vst2],[Junktor])) :-
        verbstellung(Vst1),
        verbstellung(Vst2).
```

Daneben gibt es *zweigliedrige* Satzjunktoren, bei denen wir die beiden Teile getrennt beschreiben. Der Zusammenhang geht nur aus dem (Lexikon-)Eintrag für den ersten Teil hervor:

⟨*Wortkategorien*⟩+≡
```
wortkategorie(conj1([Vst1,Junktor2],[Junktor1])) :-
        verbstellung(Vst1).
wortkategorie(conj2([Vst2],         [Junktor2])) :-
        verbstellung(Vst2).
```

Wieder steht der jeweilige Teil als Formmerkmal in der syntaktischen Kategorie, aber der zweite Teil wird im Rahmen des ersten festgeschrieben. (vgl. die Satzregeln in Abschnitt 1.2.6.)

Bem.: Diese Beschreibungen sind bisher nur sehr rudimentär eingehalten. Die Verwendung von Junktoren zur Koordination von Nominal-, Adjektiv-, und anderen Phrasen ist noch zu beschreiben.

1.2.7 Satzkonjunktionen

Die folgenden Einträge zur Bei- und Unterordnung von Sätzen sind für die Grammatik `vp.grm` erstellt und noch kaum überprüft worden:

⟨*Lexika/konjunktionen.wrd*⟩+≡
```
word(aber, conj([zweit], [aber])).
word(denn, conj([zweit], [denn])).
word(oder, conj([zweit], [oder])).
word(sondern, conj([zweit], [sondern])).
word(und, conj([zweit], [und])).
word(allein, conj([zweit], [allein])).
word(ob, conj([letzt], [ob])).
word(als, conj([letzt], [als])).
word(außer, conj([zweit], [außer])).
word(bis, conj([letzt], [bis])).
word(da, conj([letzt], [da])).
word(damit, conj([letzt], [damit])).
word(dass, conj([letzt], [dass])).
word(doch, conj([zweit], [doch])).
word(doch, conj([erst], [doch])).
word(indes, conj([erst], [indes])).
word(indes, conj([letzt], [indes])).
word(jedoch, conj([erst], [jedoch])).
word(jedoch, conj([zweit], [jedoch])).
word(so, conj([erst], [so])).
```

```
word(weil, conj([letzt], [weil])).
word(wenn, conj([letzt], [wenn])).
word(wie, conj([letzt], [wie])).
word(nun, conj([letzt], [nun])).
word(zu, conj([infzu], [zu])). % satztyp zu in vp.grm

word(je, conj([[je],letzt], [je])).
word(je, conj([[je],erst], [desto])).
word(je, conj([[je],erst], [umso])).
word(je, conj([letzt,[desto]], [je])).
word(je, conj([letzt,[umso]], [je])).
word(je, conj([letzt,[je]], [je])).

word(weder, conj([zweit, [noch]], [weder])).
word(noch, conj([[weder],erst], [noch])).
word(um, conj([[zu]], [um,zu])).
word(wo, conj([wo], [])). %letzt

% word(sobald, conj([letzt], [sobald])).
% H.L.

word(daß, conj([letzt],[daß])).
word(dass, conj([letzt],[dass])).

word(um,   conj([[infzu]],[um])).
word(ohne, conj([[infzu]],[ohne])).
word(ohne, conj([[daß]],[(ohne,daß)])).
word(so, conj([[daß]],[(so,daß)])).
word(so, conj([[wie]],[(so,wie)])).
word(ohne, conj([[dass]],[(ohne,dass)])).
word(so, conj([[dass]],[(so,dass)])).
word(so, conj([[wie]],[(so,wie)])).
```

Zur Koordinierung von anderen Phrasenarten führen wir noch zwei (vorläufige) Einträge anderer Form ein:

⟨*Wortkategorien*⟩+≡

```
wortkategorie(conj([],[und])).
wortkategorie(conj([],[oder])).
```

⟨*Lexika/konjunktionen.wrd*⟩+≡

```
word(und,  conj([], [und])).
word(oder, conj([], [oder])).
```

1.2.8 Partikeln

Bei den Partikeln bleibt eine Unterklassifizierung nach syntaktischen Kriterien noch vorzunehmen. Helbig/Buscha benutzt die Einteilung der Partiken in

(i) solche, die zu Nominalphrasen, Verben, Adjektiven und Adverbien treten können, dabei teilweise vor und teilweise nach der Nominalphrase:

⟨*Partikelgruppe 1*⟩≡

```
aber, auch, beinahe, bereits, bloß, eben, erst, etwa, gar, gerade,
geradezu, ja, nahezu, nicht einmal, noch, nur, schon
```

(ii) solche, die zu Nominalphrasen, Adjektiven und Adverbien treten können,

⟨*Partikelgruppe 2*⟩≡

```
besonders, fast, ganz
```

(iii) solche, die zu Verben, Adjektiven und Adverbien treten können:

⟨*Partikegruppe 3*⟩≡

```
doch, durchaus, immer, sehr, so, zu
```

(iv) solche, die zu Adjektiven und Adverbien treten können (Gradpartikeln):

⟨*Partikelgruppe 4*⟩≡

```
etwas, höchst, recht, überaus, viel, weit, weitaus, ziemlich
```

(v) solche, die zu Nominalphrasen und Adverbien treten können:

⟨*Partikelgruppe 5*⟩≡

```
allein, beispielsweise, zumal
```

(vi) solche, die zu Verben treten können:

⟨*Partikelgruppe 6*⟩≡

```
also, denn, nicht, nun, überhaupt
```

Wir benutzen die im Beispieltext `Testbett/wissenschaftsreligion.txt` vom CISLEX als Partikel klassifizierten Wörter. (Die Adjektivgrammatik hat eine Unterklassifikation nach der regierten Adjektivdeklination vorgenommen.)

⟨*Lexika/partikeln.wrd*⟩≡

```
word(allein, part([], [])).
word(allerdings, part([], [])).
word(also, part([], [])).
word(auch, part([], [])).
word(ausschließlich, part([], [])).
word(bereits, part([], [])).
word(bloß, part([], [])).
word(denn, part([], [])).
word(doch, part([], [])).
word(eigentlich, part([], [])).
word(einfach, part([], [])).
word(einmal, part([], [])).
word(etwas, part([], [])).
word(freilich, part([], [])).
word(ganz, part([], [])).
word(gar, part([], [])).
word(geradezu, part([], [])).
word(gleich, part([], [])).
word(immer, part([], [])).
word(ja, part([], [])).
word(man, part([], [])).
word(nicht, part([], [])).
word(noch, part([], [])).
word(nun, part([], [])).
word(nur, part([], [])).
word(nämlich, part([], [])).
word(schließlich, part([], [])).
word(schon, part([], [])).
```

```
word(sehr, part([], [])).
word(selbst, part([], [])).
word(sicher, part([], [])).
word(so, part([], [])).
word(sogar, part([], [])).
word(viel, part([], [])).
word(vielleicht, part([], [])).
word(vollständig, part([], [])).
word(zu, part([], [])).
word(zumindest, part([], [])).
```

⟨*Lexika/partikeln.wrd*⟩+≡

```
% ----------- Gradpartikel (nicht getestet) ----------

word(sehr, grad([pos])).  % sehr dumm

word(äußerst, grad([pos])). % äußerst dumm     % statt a([supl,grad])
word(höchst,  grad([pos])). % höchst unbefriedigend  % - " -

word(viel, grad([komp])). % viel dümmer
word(immer,grad([komp])). % immer dümmer
word(noch, grad([komp])). % noch dümmer

word(am,   grad([supl])). % am dümmsten  (So könnte man es zur Not machen)

word(zu,   grad([pos)).   % zu dumm, s([infzu])
word(so,   grad([pos])).  % so dumm, s([infzu])/s([dass,vl])
```

1.2.9 abtrennbare Verbpräfixe

⟨*Seminarbeitraege/HasselLiTonelli/praep.saetze.wrd*⟩≡

```
word(um, praef([um], [])).
word(ab, praef([ab], [])).
word(aus, praef([aus], [])).
word(auf, praef([auf], [])).
word(weg, praef([weg], [])).
word(vor, praef([vor], [])).
word(dar, praef([dar], [])).
word(über, praef([über], [])).
word(statt, praef([statt], [])).
word(an, praef([an], [])).
word(ein, praef([ein], [])).
word(hervor, praef([hervor], [])).
word(nach, praef([nach], [])).
```

Im CISLEX werden abtrennbare Verbpräfixe durch `VPART` angegeben,

1.2.10 Adverbien

Da die Adverbien keine Formmerkmale haben, geben wir aus Gründen der Einheitlichkeit[7] eine leere Liste von Formmerkmalen an:

[7] Sie wird –nach meiner Erinnerung– hauptsächlich zur Baumanzeige vorausgesetzt: bei einstelligen Wort- und Phrasenarten wäre unklar, ob die Merkmale als obere oder als untere Indizes darzustellen sind, d.h. ob sie Form-

⟨*wortarten.cat.pl*⟩≡

```
wortkategorie(adv([Def,AdvF],[])) :-
        definitheit_adv(Def),
        adverbialeFunktion(AdvF).
```

⟨*Lexikalische Merkmale*⟩+≡

```
definitheit_adv(def).
definitheit_adv(qu).
definitheit_adv(rel).

adverbialeFunktion(final).
adverbialeFunktion(inst).
adverbialeFunktion(kaus).
adverbialeFunktion(konz).
adverbialeFunktion(modl).
adverbialeFunktion(temp).
adverbialeFunktion(lok).
adverbialeFunktion(dir).
adverbialeFunktion(dir(hin)).
adverbialeFunktion(dir(her)).
```

Die leere Liste von Formmerkmalen ergänzen wir an „Stammeinträge“ in `wrd/2` ohne diese Liste:

⟨*Lexika/adverbien.wrd*⟩≡

```
word(W,adv(Art,[])) :-
        wrd(W,adv(Art)).
```

(i) Modaladverbien:

⟨*Lexika/adverbien.wrd*⟩+≡

```
wrd(anders,   adv([def,modl])).
wrd(gern,     adv([def,modl])).
wrd(irgendwie,adv([indef,modl])).
wrd(so,       adv([def,modl])).
wrd(wie,      adv([qu, modl])). % die Art, wie das gemacht wurde : adv([rel,modl])??
```

(ii) Adjektivadverbien (ableitbar??)

⟨*Lexika/adverbien.wrd*⟩+≡

```
wrd(gut,      adv([def,modl])).
```

(iii) Temporaladverbien[8]:

⟨*Lexika/adverbien.wrd*⟩+≡

```
% Zeitpunkt:
wrd(wann,        adv([qu, temp])).

wrd(damals,      adv([def,temp])).
wrd(früher,      adv([def,temp])). % ?? % früher, eher, später - als
wrd(seinerzeit,  adv([def,temp])).
wrd(neulich,     adv([def,temp])).
wrd(gerade,      adv([def,temp])).
```

oder Artmerkmale sind. Aber da es einige Wort- und Phrasenarten *ohne Formvarianten* gibt, könnte man einstellige Kategorienausdrücke zulassen und das Argument immer als Formargument darstellen. Nur Ausdrucksarten, die zwar Form- aber keine Artmerkmale haben, müßten dann eine leere Liste von Artmerkmalen bekommen.

[8] Helbig/Buscha unterteilen sie nach: Zeitpunkt,Zeitdauer,Wiederholung,relativer Zeit.

```
wrd(vorhin,      adv([def,temp])).
wrd(eben,        adv([def,temp])).
wrd(jetzt,       adv([def,temp])).
wrd(nun,         adv([def,temp])).
wrd(soeben,      adv([def,temp])).
wrd(sogleich,    adv([def,temp])).
wrd(heutzutage,  adv([def,temp])).
wrd(bald,        adv([def,temp])).
wrd(demnächst,   adv([def,temp])).
wrd(später,      adv([def,temp])). % ??
wrd(irgendwann,  adv([indef,temp])).

wrd(beizeiten,   adv([def,temp])).
wrd(rechtzeitig, adv([def,temp])).
wrd(früh,        adv([def,temp])).
wrd(spät,        adv([def,temp])).

wrd(vorerst,     adv([def,temp])).

% Reihenfolge
wrd(anfangs,     adv([def,temp])).
wrd(zuerst,      adv([def,temp])).
wrd(zunächst,    adv([def,temp])).
wrd(zugleich,    adv([def,temp])).
wrd(gleichzeitig,adv([def,temp])).
wrd(dann,        adv([def,temp])).
wrd(zuletzt,     adv([def,temp])).
wrd(endlich,     adv([def,temp])).
wrd(schließlich, adv([def,temp])).

% Relativer Tag
wrd(vorgestern,  adv([def,temp])).
wrd(gestern,     adv([def,temp])).
wrd(heute,       adv([def,temp])).
wrd(morgen,      adv([def,temp])).
wrd(übermorgen,  adv([def,temp])).

% Häufigkeit (wie oft?)
wrd(nie,         adv([def,temp])).
wrd(niemals,     adv([def,temp])).  % indef
wrd(selten,      adv([def,temp])).
wrd(bisweilen,   adv([def,temp])).
wrd(gelegentlich,adv([def,temp])).
wrd(manchmal,    adv([def,temp])).
wrd(mitunter,    adv([def,temp])).
wrd(zeitweise,   adv([def,temp])).
wrd(häufig,      adv([def,temp])).
wrd(oft,         adv([def,temp])).  % vielmals
wrd(immer,       adv([def,temp])).
wrd(jederzeit,   adv([def,temp])).
wrd(jemals,      adv([qu, temp])). % qu?

wrd(einmal,      adv([def,temp])).
```

```
wrd(zweimal,     adv([def,temp])).
wrd(dreimal,     adv([def,temp])).
wrd(viermal,     adv([def,temp])).
wrd(nochmals,    adv([def,temp])).
wrd(wiederum,    adv([def,temp])).
wrd(mehrmals,    adv([def,temp])).
wrd(oftmals,     adv([def,temp])).
wrd(jedesmal,    adv([def,temp])).

% Wiederholung
wrd(stündlich,   adv([def,temp])).
wrd(täglich,     adv([def,temp])).
wrd(wöchentlich, adv([def,temp])).
wrd(monatlich,   adv([def,temp])).
wrd(jährlich,    adv([def,temp])).

wrd(morgens,     adv([def,temp])).
wrd(vormittags,  adv([def,temp])).
wrd(mittags,     adv([def,temp])).
wrd(nachmittags, adv([def,temp])).
wrd(abends,      adv([def,temp])).
wrd(nachts,      adv([def,temp])).

wrd(montags,     adv([def,temp])).
wrd(dienstags,   adv([def,temp])).
wrd(mittwochs,   adv([def,temp])).
wrd(donnerstags, adv([def,temp])).
wrd(freitags,    adv([def,temp])).
wrd(samstags,    adv([def,temp])).
wrd(sonntags,    adv([def,temp])).

% immer wieder|öfter|seltener ??

% Dauer:
wrd(allezeit,    adv([def,temp])).
wrd(bislang,     adv([def,temp])).
wrd(immer,       adv([def,temp])).
wrd(längst,      adv([def,temp])).
wrd(nie,         adv([def,temp])).
wrd(noch,        adv([def,temp])).
wrd(seither,     adv([def,temp])).
wrd(stets,       adv([def,temp])).
wrd(zeitlebens,  adv([def,temp])).

% Relative Zeit:
wrd(indessen,    adv([def,temp])).
wrd(nachher,     adv([def,temp])).
wrd(seitdem,     adv([def,temp])).
wrd(vorher,      adv([def,temp])).
wrd(unterdessen, adv([def,temp])).
```

Bem. Die Definitheitswerte sind flüchtig gesetzt und gegebenenfalls zu korrigieren.

(iv) Finaladverbien:

⟨*Lexika/adverbien.wrd*⟩+≡

```
wrd(dazu,        adv([def,final])).
wrd(darum,       adv([def,final])).
wrd(deshalb,     adv([def,final])).
wrd(deswegen,    adv([def,final])).
wrd(hierfür,     adv([def,final])).
wrd(hierzu,      adv([def,final])).
wrd(warum,       adv([qu, final])).
wrd(wozu,        adv([qu, final])).
```

(v) Kausaladverbien:

⟨*Lexika/adverbien.wrd*⟩+≡

```
wrd(warum,       adv([qu ,kaus])).
wrd(darum,       adv([def,kaus])).

wrd(deswegen,    adv([def,kaus])).
wrd(weswegen,    adv([qu, kaus])).
wrd(deshalb,     adv([def,kaus])).
wrd(weshalb,     adv([qu, kaus])).

wrd(meinetwegen, adv([def,kaus])).
wrd(deinetwegen, adv([def,kaus])).
wrd(seinetwegen, adv([def,kaus])).
wrd(ihretwegen,  adv([def,kaus])).
wrd(unseretwegen,adv([def,kaus])).
wrd(euretwegen,  adv([def,kaus])).

wrd(meinethalben,      adv([def,kaus])).
wrd(anstandshalber,    adv([def,kaus])).
wrd(sicherheitshalber, adv([def,kaus])).

wrd(also,              adv([def,kaus])).
wrd(daher,             adv([def,kaus])).

wrd(demnach,           adv([def,kaus])).
wrd(folglich,          adv([def,kaus])).
wrd(infolgedessen,     adv([def,kaus])).
wrd(mithin,            adv([def,kaus])).
wrd(nämlich,           adv([def,kaus])). % Stellung! *(nämlich Sve)
wrd(somit,             adv([def,kaus])).
% wrd(so,                adv([def,kaus])).
```

(vi) Konditionaladverbien:

⟨*Lexika/adverbien.wrd*⟩+≡

```
wrd(andernfalls,       adv([def,kond])). % gleichfalls
wrd(bestenfalls,       adv([def,kond])).
wrd(gegebenenfalls,    adv([def,kond])).
wrd(nötigenfalls,      adv([def,kond])).
wrd(schlimmstenfalls,  adv([def,kond])).
wrd(sonst,             adv([def,kond])).
wrd(genaugenommen,     adv([def,kond])).
wrd(strenggenommen,    adv([def,kond])).
```

(vii) Konzessivadverbien:

⟨*Lexika/adverbien.wrd*⟩+≡
```
wrd(dennoch,           adv([def,konz])).
wrd(dessenungeachtet,  adv([def,konz])).
wrd(doch,              adv([def,konz])).
wrd(gleichwohl,        adv([def,konz])).
wrd(nichtsdestoweniger,adv([def,konz])).
wrd(nichtsdestotrotz,  adv([def,konz])).
wrd(trotzdem,          adv([def,konz])).
```

(viii) Konsekutivadverbien:

⟨*Lexika/adverbien.wrd*⟩+≡
```
% wrd(so,               adv([def,kons])). % so daß
```

(ix) Lokaladverbien:

⟨*Lexika/adverbien.wrd*⟩+≡
```
wrd(anderswo, adv([indef,lok])).
wrd(auswärts, adv([def,lok])).
wrd(außen,    adv([def,lok])).
wrd(da,       adv([def,lok])).
wrd(dort,     adv([def,lok])).
wrd(draußen,  adv([def,lok])).
wrd(drinnen,  adv([def,lok])).
wrd(drüben,   adv([def,lok])).
wrd(hier,     adv([def,lok])).
wrd(hinten,   adv([def,lok])).
wrd(innen,    adv([def,lok])).
wrd(irgendwo, adv([indef,lok])).
wrd(links,    adv([def,lok])).
wrd(nebenan,  adv([def,lok])).
wrd(nirgendwo,adv([def,lok])).
wrd(nirgends, adv([def,lok])).
wrd(oben,     adv([def,lok])).   % oben-an, oben-auf
wrd(rechts,   adv([def,lok])).
wrd(unten,    adv([def,lok])).
wrd(überall,  adv([def,lok])).
wrd(vorn,     adv([def,lok])).
wrd(wo,       adv([qu ,lok])).
```

(x) Direktionaladverbien:

⟨*Lexika/adverbien.wrd*⟩+≡
```
wrd(daher,    adv([def,dir(her)])).   % pp([def],[aus(dat)]) ?
wrd(dorther,  adv([def,dir(her)])).
wrd(hierher,  adv([def,dir(her)])).
wrd(woher,    adv([qu, dir(her)])).

wrd(dahin,    adv([def,dir(hin)])).   % pp([def],[in(akk)]) ?
%wrd(dorthin,  adv([def,dir(hin)])).
wrd(hierhin,  adv([def,dir(hin)])).
wrd(wohin,    adv([qu, dir(hin)])).

% überall-,irgendwo-,anderswo-nirgendwo- her|hin
```

```
wrd(abwärts,  adv([def,dir(hin)])).
wrd(aufwärts, adv([def,dir(hin)])).
wrd(bergab,   adv([def,dir(hin)])).   % talaufwärts ?
wrd(bergauf,  adv([def,dir(hin)])).
wrd(fort,     adv([def,dir(hin)])).   % nur Verbpräfix?
wrd(heim,     adv([def,dir(hin)])).
wrd(heimwärts,adv([def,dir(hin)])).
wrd(rückwärts,adv([def,dir(hin)])).
wrd(seitwärts,adv([def,dir(hin)])).
wrd(vorwärts, adv([def,dir(hin)])).
wrd(weg,      adv([def,dir(hin)])).   % nur Verbpräfix?
```

(xi) Instrumentaladverbien:

⟨*Lexika/adverbien.wrd*⟩+≡

```
% Die folgenden Pronominaladverbien konstruiert man besser ueber
% mod([Def,Art]) --> pp([Praep,Art,...]),
% da sie auch Argument-PP's repraesentieren koennen.
%
% wrd(damit,  adv([def,inst])).
% wrd(hiermit,adv([def,inst])).
% wrd(womit,  adv([qu,inst])).

% Aus Adjektiven gebildete Adverbien:
% wrd(längst, adv([def,temp])).
```

1.2.11 Restriktion von Merkmalen bei Wortarten

Für die Vorberechnung der First-Relation zur Voraussage nützlicher Grammatikregeln während der Syntaxanalyse müssen die Merkmale bei den Wort- und Phrasenkategorien teilweise ignoriert werden, da sonst die Vorberechnung zu aufwendig wird und zu große Hilfstabellen erzeugt.

Bei den Wortarten werden die Merkmale nach folgender Tabelle ignoriert:

⟨*Lexika/wortarten.cat.pl*⟩+≡

```
restrict(v(Verbklasse,Form),v(Verbklasse,Form)) :- !. % vorlaeufig
% restrict(v([hilfsverb,Verb],[Pers,Num,Temp,Mod]),
%        v([hilfsverb,Verb],[Pers,Num,_,_])) :- !.
restrict(vn(Art,Form),vn(_,_)) :- !.
restrict(nn(Art,Form),nn(_,_)) :- !.
% restrict(pron([Def,Pers,Gen],[Num,Kas]),pron([Def,Pers,_],_)) :- !.
% restrict(pron(DefArt,[Gen,Num,Kas,Pers]),pron(DefArt,[_,_,_,Pers])) :- !.
restrict(pron(DefArt,Form),pron(_DefArt,_Form)) :- !.
restrict(pron(Art,[Gen,Num,Kas]),pron(Art,[_,_,_])) :- !.
restrict(n([Gen],[Num,Kas]),n([_],[_,_])) :- !.
restrict(n([Gen,Rahmen],[Num,Kas]), n([Gen,Rahmen],[_,_])) :- !.
restrict(n([Gen,Rahmen],[Adekl,Num,Kas]), n([Gen,Rahmen],[Adekl,_,_])) :- !.
restrict(a(Art,[Komp,Adekl,Gen,Num,Kas]), a(Art,[_Komp,Adekl,_,_,_])) :- !.
restrict(a(Art,[Komp,praedik]),a(Art,[Komp,praedik])) :- !.
% restrict(pr([[Kas],AdvF],[Praep]),pr([[Kas],_],[Praep])) :- !.
% restrict(po([[Kas],AdvF],[Praep]),po([[Kas],_],[Praep])) :- !.
restrict(pr([[Kas]],[Praep]),pr([[Kas]],[Praep])) :- !.
restrict(po([[Kas]],[Praep]),po([[Kas]],[Praep])) :- !.
```

```
restrict(det([Def,Adekl],[Gen,Num,Kas]),det([Def,Adekl],[_,_,_])) :- !.
restrict(na(Art, _Form), na(_Art,_Form)) :- !.
restrict(adv(Art, Form), adv(_,_)) :- !.
restrict(proPP(Art,Form), proPP(_,_)) :- !.
```

Von den Merkmalen der folgenden Wortkategorien werden (vorerst) alle zur Vorberechnung berücksichtigt:

⟨*Lexika/wortarten.cat.pl*⟩+≡

```
restrict(pdet([[Kas], Gen], [Praep]),pdet([[Kas], Gen], [Praep])) :- !.
restrict(pc1(Art,Form),pc1(Art,Form)) :- !.
restrict(pc2(Art,Form),pc2(Art,Form)) :- !.
restrict(anrede([Gen],[Num,Kas]),anrede([Gen],[Num,Kas])) :- !.
restrict(titel([unsp],[Num]),titel([unsp],[Num])) :- !.
restrict(ordinalzahl,ordinalzahl) :- !.
restrict(kardinalzahl,kardinalzahl) :- !.
restrict(part(Art, []), part(Art,[])) :- !.
restrict(conj([],[Junktor]), conj([],[Junktor])) :- !.
restrict(conj([Vst], [Junktor]), conj([Vst],[Junktor])) :- !.
restrict(praef([Praefix], []), praef([Praefix],[])) :- !.
```

Bei der Erzeugung eines `.epl`- oder `.glp`-Parsers wird mit `pruefe_restrictor/0` geprüft, ob zu jeder Kategorie eine Einschränkung in `restrict/2` formuliert ist. Fehlt eine solche Einschränkung, so kann die für diese Parser vorberechnete Relation `first/2` zu klein sein und deshalb eine korrekte Analyse irrtümlich fehlen.

1.3 Nominalphrasen

1.3.1 Klassen- und Formparameter

In der Datei `Grammatiken/np.cat.pl` geben wir wieder gesondert die Merkmale von Nominalphrasen und ihre Wertebereiche an und erklären sie zu Hilfsprädikaten, die in die für die Parser vorberechneten Grammatiken übertragen werden.

Das folgende Prädikat `kategorie/1` wird ggf. in weiteren Teilgrammatiken erweitert und ist daher entsprechend deklariert. (Bisher wird es nicht verwendet; ggf. muß beim Beginn der Vorberechnungen ein vorhandenes Prädikat `kategorie/1` gelöscht werden.)

⟨*Grammatiken/np.cat.pl*⟩≡

```
% :- use_module('Lexika/wortarten.cat.pl').
:- ensure_loaded('Lexika/wortarten.cat.pl').
auxiliary_predicates([persGenNum/3,definitheit_np/1,genusErw/1]).

:- multifile kategorie/1.
```

Die Kategorie der Nominalphrasen hat folgende Klassen- und Formparameter:

⟨*Grammatiken/np.cat.pl*⟩+≡

```
kategorie(np([Def,Pers,Gen],[Num,Kas])) :-
        persGenNum(Pers,Gen,Num),
        kasus(Kas),
        definitheit_np(Def).

kategorie(n1([Gen],[Komp,Adekl,Num,Kas])) :-
```

```
        genusErw(Gen),
        numerus(Num),
        kasus(Kas),
        (komparation(Komp); Komp = -),
        (adjektivdeklination(Adekl); Adekl = -).

kategorie(en([Gen],[Num,Kas])) :-
        genusErw(Gen), numerus(Num), kasus(Kas).
```

Zur Vorberechnung der First-Relation der Grammatik sollen nur die Merkmale der Definitheit, des Genus, und des Numerus berücksichtigt werden:

⟨*Grammatiken/np.cat.pl*⟩+≡

```
restrict(np([Def,Pers,Gen],[Num,Kas]),np([_Def,_,_Gen],[_Num,_])) :- !.
% restrict(n1([Gen],[Komp,Adekl,Num,Kas]),n1([_Gen],[_,_Adekl,_Num,_])) :- !.
restrict(n1([Gen],[Komp,Adekl,Num,Kas]),n1([_Gen],[_,Adekl,_Num,_])) :- !.
restrict(en(Art,Form),en(_,_)) :- !.
```

Eine genauere Beschreibung erfordert mindestens noch ein Merkmal, das die Komparationsstufe eingebetteter Adjektivmodifikationen angibt: man braucht sie zur Einschränkung der Nominalphrasen in Vergleichskonstruktionen wie `ein größerer Fehler als dieser`. Allerdings wäre dieses Komparationsmerkmal, wenn es ein Formmerkmal ist, nur bei bestimmten Nominalphrasen variierbar.

Für optionale Konstituenten legen wir noch fest:

⟨*Grammatiken/np.cat.pl*⟩+≡

```
kategorie(?C) :- kategorie(C).
restrict(?C,?C):- !.
```

Falls die synthetischen Merkmale einer optionalen Konstituente die Eigenschaften der übergeordneten Phrase mitbestimmen, müssen diese Merkmale auch dann einen Wert haben, wenn die optionale Konstituente fehlt.

Das ist z.B. für den Deklinationstyp `Adekl` des Adjektivs der Fall, wenn man `Adekl` als Merkmal der `n1` vergibt. Wenn hier keine Adjektivphrase auftritt, geben wir das durch `Komp = -` und `Adekl = -` an; Phrasenflexionsprogramme können dann am `Adekl`-Merkmal der `n1` erkennen, daß diese Nominalphrasen nicht komparierbar sind:[9]

Dazu muß aber für jede Kategorie `A` gesondert formuliert werden, für welche Variablenbelegung `?A` löschbar ist. Hierbei kann `A` eine zusammengesetzte Kategorie sein, aber dann erscheint `?A` als Knoten im Baum! Für atomare Kategorien ist das durch die Behandlung von Ketten- und Löschregeln verhindert. Falls `A` von der Form `(B;C)` ist, kann man statt `?(B;C)` die Disjunktion `(?B;?C)` und dann die Regeln für diese Teile schreiben. (Bei optionalen Konjunktionen `?(B,C)` geht das aber nicht mit `(?B,?C)`.)

Die Merkmalwerte und Kongruenzprädikate für Nominalphrasen sind wie folgt:

⟨*Grammatiken/np.cat.pl*⟩+≡

```
persGenNum(3,Gen,pl) :- genusErw(Gen).
persGenNum(3,Gen,sg) :- genus(Gen).
persGenNum(Pers,unsp,Num) :-
        (Pers = 1; Pers = 2), numerus(Num).

definitheit_np(def).
```

[9] Entsprechend sollte das Nichtvorliegen/Vorliegen einer NP-Genitivergänzung kodiert werden: diese kollidiert ggf. mit einem Possessivum als Determinator.

```
definitheit_np(indef).
definitheit_np(rel).
definitheit_np(qu).
definitheit_np(quant).
definitheit_np(poss).
definitheit_np(refl).
```

Die Merkmalbereiche der Art- und Formmerkmale `person/1`, `genus/1`, `numerus/1`, `kasus/1`, `adjektivdeklination/1` und `komparation/1` haben die üblichen Werte, die in `Lexika/wortarten.cat.pl` festgelegt wurden; sie werden von `np.grm`) geladen, und müssen in den erzeugten Parser exportiert werden:

⟨*Grammatiken/np.cat.pl*⟩+≡

```
auxiliary_predicates([kasus/1,numerus/1,person/1,genus/1,genusErw/1]).
auxiliary_predicates([adjektivdeklination/1,komparation/1]).
```

Die indefiniten Pronomina im Singular wie `man,jemand`, haben im CISLEX das Genus `unsp`, sollten aber `mask` haben: im Moment werden sie durch `persGenNum` ausgeschlossen! (Man soll *seine* Grammatik einfach halten!)

1.3.2 Grammatikregeln

Wir geben optionale Konstituenten in der Form `?(Kategorie)` an, wofür am Ende bestimmte Löschregeln angegeben werden, die die Belegung der synthetischen Merkmale festlegen. (Dazu sind die Parser erweitert worden.)

⟨*Grammatiken/np.grm*⟩≡

```
% Deklarartionen:

:- op(1100,xfx,'-->').
:- multifile auxiliary_predicates/1.

:- consult('Lexika/wortarten.cat.pl').
:- consult('Grammatiken/np.cat.pl').

% Hilfsprädikate:

auxiliary_predicates([start_symbol/1,kongruenzGenusNP/2,kongruenzAdeklNP/2]).

start_symbol(np([Def,Pers,Gen],[Num,Kas])).
```

(i) Nominalphrasen in der dritten Person können aus einem Determinator und einem (optional) modifizierten Nomen gebildet werden. Optional kann ein Relativsatz folgen:

⟨*Grammatiken/np.grm*⟩+≡

```
np([Def,3,Gen],[Num,Kas]) -->
        det([Def,AdeklD],[GenD,Num,Kas]),
        n1([Gen],[Komp,Adekl,Num,Kas]),
        ?(s([rel(GenD,Num)],[_Temp,_Mod,letzt]))
        :-
        % definitheit_np(Def),
        % kasus(Kas),
        % numerus(Num),
        % komparation(Komp),
        kongruenzGenusNP(Gen,GenD),
        kongruenzAdeklNP(Adekl,AdeklD).
```

Das Genus des Determinators wird vom Nomen regiert, die Adjektivdeklination des modifizierten Nomens vom Determinator:

⟨*Grammatiken/np.grm*⟩+≡
```
kongruenzGenusNP(Gen,GenD) :-
        % genus(Gen),
        (GenD = Gen ; GenD = unsp).
kongruenzAdeklNP(Adekl,AdeklD) :-
        % adjektivdeklination(AdeklD),
        (Adekl = AdeklD ; Adekl = -).
```

⟨*Testbett/Beispiele/np.txte*⟩≡
```
kein Zufall. keine Fragen. einige Menschen. diese Frau. welcher Mann.
solchen popularisierenden Schlussfolgerungen. % ?? solche:det(quant)
```

(ii) In einer Nominalphrase der dritten Person darf statt eines Determinators auch eine Nominalphrase im Genitiv stehen:

⟨*Grammatiken/np.grm*⟩+≡
```
np([Def,3,Gen],[Num,Kas]) -->
        np([Def,3,_Gen],[_Num,gen]),
        n1([Gen],[Komp,Adekl,Num,Kas]),
        ?(s([rel(GenD,Num)],[_Temp,_Mod,letzt]))
        :-
        % definitheit_np(Def),
        % kasus(Kas),
        % numerus(Num),
        % komparation(Komp),
        kongruenzGenusNP(Gen,GenD),
        kongruenzAdeklNP(Adekl,ge).
```

Falls statt des Determinators eine Nominalphrase im Genitiv steht, bestimmt ihre Definitheit die der gesamten Phrase, und deren Adjektivdeklination ist gemischt. Hierunter werden auch `dessen,wessen` behandelt, die also im Unterschied zu `sein` nicht als Determinatoren geführt werden.

⟨*Testbell/Beispiele/np.txte*⟩≡
```
Prof. Darwins Entdeckung.
deren Entdeckung.
wessen Wissenschaftlers Entdeckung.
```

Fehler: Die Regel verlangt nicht, daß die vorangestellte `np` nicht pronominalisiert ist:

⟨*Fehler*⟩≡
```
1. np([indef, 3, fem], [sg, akk])
      np([indef, 3, fem], [sg, gen])
         pron([indef, pos], [fem, sg, gen, 3])
            einer
      n1([fem], [-, -, sg, akk])
         n([fem], [sg, akk])
            Person
```

was unbedingt ausgeschlossen werden muß.

(iii) Das modifizierte Nomen kann durch eine vorangestellte (deklinierte) Adjektivphrase und durch nachgestellte Genitiv- oder Präpositional-Attribute modifiziert sein. Das adjektivisch modifizierte Nomen erbt Komparation und Adjektivdeklination vom Adjektiv:

⟨*Grammatiken/np.grm*⟩+≡
```
n1([Gen],[Komp,Adekl,Num,Kas]) -->
        ?ap([Komp,Adekl,GenA,Num,Kas]),
        (n([Gen],[Num,Kas]) ; en([Gen],[Num,Kas]) ; en([unsp],[Num,Kas])),
        (?np([Def2,3,Gen2],[Num2,gen]) ; ?pp([DefPP],[Praep]))
        :-
        % kasus(Kas),
        % numerus(Num),
        % komparation(Komp),
        % adjektivdeklination(Adekl),
        kongruenzGenusNP(Gen,GenA).
```

⟨*Testbett/Beispiele/np.txte*⟩+≡
```
technische Eingriffe. viele Romane. eine verantwortliche Person.
```

⟨*Grammatiken/np.grm*⟩+≡
```
% n1([Gen],[-,-,Num,Kas]) -->
%       n([Gen],[Num,Kas]),
%       ?(np([Def2,3,Gen2],[Num2,gen]) ; pp([DefPP],[Praep])).
%       % :-
%       % kasus(Kas), genus(Gen), numerus(Num).
```

⟨*Testbett/Beispiele/np.txte*⟩+≡
```
Problem der Naturwissenschaften.
Mittel der Evolution.
Versuch Darwins.
Abwesenheit von Zwang.
Versuche zur Optimierung.
```

Bem.: Man kann zwischen 'freien' Ergänzungen und Komplementen unterscheiden, wenn man bei den Nomen einen Komplementrahmen einfügt - was für nominalisierte Verben sinnvoll ist.

(iv) Nominalisierte Adjektive, die durch vorangestellte adjektivische oder nachgestelle nominale oder präpositionale Ergänzungen erweitert sein können, sind erweiterte Nomina:

⟨*Grammatiken/np.grm*⟩+≡
```

n1([Gen],[Komp,Adekl,Num,Kas]) -->
        ?ap([Komp,Adekl,GenA,Num,Kas]),
        na([Gen],[_Komp,_Adekl,Num,Kas]),
        (?np([Def2,3,Gen2],[Num2,gen]) ; ?pp([DefPP],[Praep]))
        :-
        % kasus(Kas),
        % numerus(Num),
        % komparation(Komp),
        % adjektivdeklination(Adekl),
        kongruenzGenusNP(Gen,GenA).
```

⟨*Testbett/Beispiele/np.txte*⟩+≡
```
ein Betroffener.
der Betroffene. % scheitert am Genus unsp im Lexikoneintrag.
der Betroffenen.
die Betroffenen.
den Betroffenen.
das Kontroverse an der Globalisierung.
```

Fehler: Die Lexikoneinträge zu nominalisierten Adjektiven scheinen mir nicht vollständig und fehlerhaft zu sein. Möglicherweise bewirkt auch ein Genus `undef` im Singular ein Problem mit der Kongruenz zum Artikel.

Soll man die Merkmale `Komp,Adekl` nach der adjektivischen Erweiterung oder nach den inhärenten Merkmalen des nominalisierten Adjektivs belegen? Der Unterschied tritt bei

⟨*Testbett/Beispiele/np.problematisch*⟩≡
```
der dümmste Junge.
der jüngste Dumme.
die noch Jüngeren.
ein Dümmerer als August ist kaum vorstellbar.
```

auf. Wir implementieren vorläufig die erste Version.

(v) Eigennamen, die durch eine vorangestellte Tätigkeitsbezeichnung oder eine Anrede erweitert sein können, bilden ein erweitertes Nomen:

⟨*Grammatiken/np.grm*⟩+≡
```
n1([Gen],[Komp,Adekl,Num,Kas]) -->
        ?ap([Komp,Adekl,GenA,Num,Kas]),
        ?n([Gen],[Num,Kas]),
        en([Gen], [Num,Kas]) :-
        kongruenzAdeklNP(Adekl,sw),
        kongruenzGenusNP(Gen,GenA).
```

⟨*Testbett/Beispiels/np.txte*⟩≡
```
der Wissenschaftler Sigmund Freud.
der fähige Wissenschaftler Darwin.
der fähige Darwin.
der fähige Prof. Dr. Darwin   % geht nicht
```

Die Adjektivdeklination ist auf `sw` eingeschränkt, damit die Konstruktion nur mit bestimmtem Artikel benutzt wird.

Fehler: Falls das Genus beim `en` unspezifiziert ist, stimmt die Kongruenzbedingung zu `GenA` nicht; z.T. wegen der Genusmerkmale der Adjektive. Zwei gleiche Analysen bei `der fähige Sigmund Freud`.

(vi) Modifizierte Nomina im Plural sind volle (indefinite) Nominalphrasen:

⟨*Grammatiken/np.grm*⟩+≡
```
np([indef,3,Gen],[pl,Kas]) -->
        n1([Gen],[Komp,Adekl,pl,Kas]),
        ?s([rel(unsp,pl)],[_Temp,_Mod,letzt]) :-
        persGenNum(3,Gen,pl),
        (Adekl = st; (Adekl = -, Komp = -)),
        (Kas = nom ; Kas = akk ; Kas = dat).
```

⟨*Testbett/Beispiele/np.txte*⟩+≡
```
Männer.
logische Probleme.
```

Bem.: Wir schließen den Genitiv aus, da sonst Fehler oder ungewöhnliche Analysen erlaubt werden, wenn die Nominalphrase als Genitivattribut einer anderen verwendet wird, z.B. bei `(Steigerung (Möglichkeiten:np(gen)):n1`.

Fehler: Dadurch werden aber seltenere Fälle wie `als Personen:gen` nicht mehr erkannt.

(vii) Modifizierte Nomina im Singular sind ebenfalls vollständige (indefinite) Nominalphrasen, wenn das Adjektivattribut stark dekliniert ist:

⟨*Grammatiken/np.grm*⟩+≡
```
np([indef,3,Gen],[sg,Kas]) -->
        n1([Gen],[Komp,Adekl,sg,Kas]),
        ?(s([rel(Gen,sg)],[_Temp,_Mod,letzt])) :-
        persGenNum(3,Gen,sg),
        (Adekl = st; (Adekl = -, Komp = -, Kas = nom)).
        % kasus(Kas).
```

⟨*Testbett/Beispiele/np.txte*⟩+≡
```
Kunst.
wissenschaftlicher Erkenntnis.
großem Zweifel.
größeren Zweifel.
von jeglichem Zweifel.
in immer größerem Umfang.
die Entwicklung intelligenter Software.
```

Nur im Nominativ (und im Akkusativ?) braucht kein Adjektiv vorhanden zu sein: `Haß ist eine Leidenschaft`. Aber im Genitiv und Dativ braucht man einen Artikel oder ein Attribut: `* die Entwicklung Software`.

(viii) Definite, interrogative und relativierende Personalpronomina sind Nominalphrasen, die nach der Person und dem Genus klassifiziert sind:

⟨*Grammatiken/np.grm*⟩+≡
```
np([Def,Pers,Gen],[Num,Kas]) -->
        pron([Def,Pron],[Gen,Num,Kas,Pers]) :-
        (Pron = pers ; Pron = pos ; Pron = neg),
        ((Pers = 3, genusErw(Gen)) ; ((Pers = 2 ; Pers = 1), Gen = unsp)).

np([Def,Pers,Gen],[Num,Kas]) -->
        pron([Def],[Gen,Num,Kas,Pers]) :-
        (Def = refl ; Def = qu),
        (Pers = 3 ; ((Pers = 2 ; Pers = 1), Gen = unsp)).

np([Def,Pers,Gen],[Num,Kas]) -->
        pron([Def],[Gen,Num,Kas]) :-
        Def = rel,
        (Pers = 3 ; ((Pers = 2 ; Pers = 1), Gen = unsp)).

np([Def,Pers,Gen],[Num,Kas]) -->
        pron([Def],[Gen,Num,Kas,Pers]),
        ?(s([rel(Gen,Num)],[_Temp,_Mod,letzt])) :-    % Fehler Pers=3: GenPron diff GenSrel
        Def = qu,
        (Pers = 3 ; ((Pers = 2 ; Pers = 1), Gen = unsp)).
```

Die Klassenmerkmale der Pronomina müssen hier und im Lexikon zum Teil noch überdacht bzw. an die obige Spezifikation angepaßt werden.

⟨*Testbett/Beispiele/np.txte*⟩+≡
```
wir. uns. sie. wem. wessen. nichts. etwas. einer. jemand. niemand.
man. sich. seiner.
```

(ix) Konjunktion und Disjunktion von Nominalphrasen sind Nominalphrasen im Plural:[10]

⟨*Grammatiken/np.grm*⟩+≡
```
np([Def,3,Gen],[pl,Kas]) -->
        np([Def1,3,Gen1],[Num1,Kas]),
        conj([],[UndOder]),
        np([Def2,3,Gen2],[Num2,Kas])
        :-
        % genusErw(Gen1), genus(Gen2),
        ((Gen1 = Gen2, Gen = Gen1) ; Gen = unsp),
        % definitheit_np(Def1), definitheit_np(Def_2),
        ((Def1 = Def2, Def = Def1) ; Def = Def1).
        % numerus(Num1),kasus(Kas).
```

Ihr Genus kann unspezifiziert sein, wenn eine der Nominalphrasen ein unspezifiziertes Genus hat, oder wenn die Teilphrasen unterschiedliche Genera haben. Unterschiedliche Definitheitseigenschaften der Teilphrasen sind hier ausgeschlossen worden.

⟨*Testbett/Beispiele/np.txte*⟩+≡
```
Romane oder Filme.
Wissenschaft und Technik.
Virchow und Haeckel.
Experten und Gutachtern.
Hirnforschern, Psychologen und Neurophilosophen.
sozialen Veränderungen und wissenschaftlicher Erkenntnis.
die Zustimmung gleicher und freier Staatsbürgerinnen und Staatsbürger.
```

(x) Eigennamen (inklusive Anrede und Titel) sowie Vornamen bilden eine Nominalphrase der dritten Person (nach den `n1`-Regeln aber auch adjektivisch erweiterte Nomina):

⟨*Grammatiken/np.grm*⟩+≡
```
np([def,3,Gen],[Num,Kas]) -->
        anrede([Gen],[Num,Kas]),
        en([Gen2],[Num,Kas]),
        ?(s([rel(Gen,Num)],[_Temp,_Mod,letzt]))
        :- kongruenzGenusNP(Gen,Gen2).
np([def,3,Gen],[Num,Kas]) -->
        en([Gen],[Num,Kas]),
        ?(s([rel(Gen,Num)],[_Temp,_Mod,letzt])).
np([def,3,Gen],[Num,Kas]) -->
        vn([Gen],[Num,Kas]),
        ?(s([rel(Gen,Num)],[_Temp,_Mod,letzt])).
```

(Das Genus der Anrede überschreibt das ggf. unspezifische Genus des Eigennamens, daher geht es nicht in *einer* Regel mit optionaler Anrede vor dem Eigennamen.)

⟨*Testbett/Beispiele/np.txte*⟩+≡
```
Hr. Freud.
Hrn. Dr. Freud.
Fr. Freud.
Sigmund. Maria.
```

Obwohl Numerusmarkierung kaum an Vornamen, Genusunterschiede (im Deutschen) kaum bei Nachnamen auftreten, und Kasusunterschiede bei beiden fast nur beim Genitiv vorkommen, sehen wir diese Merkmale für den zusammengesetzten Namen vor (z.B. für den Vergleich mit

[10] Diese Regel bringt die Vorberechnung des Predictors an Platzgrenzen, wenn die Merkmalbereiche nicht auskommentiert werden!

einer Anrede oder einer Tätigkeits- oder Funktionsbezeichnung wie in `der Mediziner Sigmund Freud`):

Wir lassen bis zu drei Titeln vor dem Vor- und Nachnamen zu. Im Plural und im Singular ohne Vorname bleibt das Genus unspezifiziert:

⟨*Grammatiken/np.grm*⟩+≡
```
en([unsp], [Num,Kas]) -->
        ?(titel([unsp],[Num])),
        ?(titel([unsp],[Num])),
        ?(titel([unsp],[Num])),
        nn([],[Num,Kas]).
```

(Für seltene Formen, wie denen in `die Meiers`, `alle Hänse`, `den Fritzen`, `Franzens Feste`, kann man dann Einträge im Lexikon angeben.)

Zwei Vornamen im Singular, an denen das Genus abgelesen wird, sind vor dem Nachnamen erlaubt:[11]

⟨*Grammatiken/np.grm*⟩+≡
```
en([Gen], [sg,Kas]) -->
        ?(titel([unsp],[sg])),
        ?(titel([unsp],[sg])),
        ?(titel([unsp],[sg])),
        vn([Gen],[sg,KasV]),
        ?(vn([Gen],[sg,KasV])),
        nn([],[sg,Kas])
        :- (Kas = gen, KasV = nom ; Kas = KasV).
```

⟨*Testbett/Beispiele/np.txte*⟩+≡
```
Platon.
Prof. Darwin.
Dr. Sigmund Freud.
Prof. Dr. Dr. Freud.
```

Der Kasus des Vornamens weicht im Genitiv vom Kasus des Eigennamens ab, weshalb wir ihn dann auf `nom` fixieren.

Noch mehr aufeinanderfolgende Titel und Vornamen könnte man –auch für Reihungen von Adjektiven– durch eine Notation `{Cat}+` erreichen, die im Parser je nach 1-Vorausschau gelöscht, zu `Cat, {Cat}+` oder nur zu `Cat` expandiert würde. (Das ließe sich in den Earley-Parser einbauen.)

(xi) Für optionale Konstituenten müssen die Werte der synthetischen Merkmale für den Fall festgelegt werden, wo die Konstituente fehlt:

⟨*Grammatiken/np.grm*⟩+≡
```
?ap([-,-,GenA,Num,Kas]) --> [].
?ordinalzahl --> [].
?kardinalzahl --> [].
?s([rel(Gen,Num)],[Temp,Mod,Vst]) --> [].
?pp([-],[Praep]) --> [].
?n([-],[Num,Kas]) --> [].
?np([-,Pers,Gen],[Num,Kas]) --> [].
?titel([Gen],[Num]) --> [].
?vn([Gen],[Num,Kas]) --> [].
```

[11] Ein Vorname ist obligatorisch; die Regeln kann man nur zusammenfassen, wenn die Merkmale aus optionalen Konstituenten durch Defaultwerte belegt werden, falls diese Konstituenten nicht vorhanden sind.

Falls keine optionalen Konstituenten in der Grammatik auftreten, muß man hilfsweise eine Löschregel angegeben, z.B. `loeschbar --> [].`, damit `del/1` existiert; sonst bekommt man einen Fehler.

Bemerkung 1.2 Zu den Konstruktionen von Nominalphrasen, die noch nachzutragen sind, gehören:

(xii) Nominalphrasen mit rechtsextrahiertem Relativsatz, wenn ein allgemeiner Mechanismus zur Konstituentenextraktion implementiert wird.

(xiii) Nominalphrasen mit Weglassung des Nomens, wie in

⟨*Testbett/Beispiele/np.txte (geht nicht)*⟩≡

```
eines seiner Probleme. viele der schwierigsten Probleme.
das schwierigste seiner Probleme. die 3 Wissenschaftler.
```

Mit den bisherigen Regeln erhält man nur die nicht-elliptischen Formen wie `das schwierigste [Problem] seiner Probleme`.

(xiv) Nominalphrasen der dritten Person, die aus einem Eigennamen für Sachen (Städte, Länder, Flüsse) bestehen. Diese Eigennamen werden im Lexikon (noch) als einfache Nomina klassifiziert. Das erlaubt zwar

⟨*Testbett/Beispiele/np.txte*⟩+≡

```
das schöne München. der Main.
```

aber nicht die Verwendung von `München` als volle Nominalphrase, wie in `Münchens Staßen. die Straßen Münchens. Frankfurt am Main`.

(xv) Nominalphrasen mit einer Funktions- oder Typbezeichnung, wie `die Stadt Frankfurt`, `das Altersheim Seeblick`, oder `der Metzger Hans Wurst`.

1.3.3 Erkennung und Analyse von Beispielphrasen

Im Verzeichnis `Testbett/Beispiele` finden sich Dateien mit Nominalphrasen, die dem Referenztext `Testbett/wissenschaftsreligion.txt` entnommen oder zumindest mit Wörtern aus einem der Referenztexte gebildet sind.

Mit `erkennePhrasen/0` (oder mit `Erkenne Testphrasen` aus dem Menü) kann man eine Datei lesen, in Phrasen zerlegen und die Phrasen mit dem Erkenner auf grammatiksche Korrektheit überprüfen lassen:

⟨*Erkennen von Beispielphrasen*⟩≡

```
?- erkennePhrasen.
Eingabedatei: 'Testbett/Beispiele/np.txte'.
Ausgabedatei: Testbett/Beispiele/np.txte.erk.app.epl.21.4.2002:22:51
...........................
Done.
```

Die Ergebnisse werden in eine Ausgabedatei im gleichen Verzeichnis geschrieben; sie erhält ein Suffix .erk.<Grammatik>.<Parser>.<Datum>:<Uhrzeit>. Für jede bearbeitete Phrase wird ein Punkt . am Bildschirm ausgegeben.

Entsprechend kann man mit `analysierePhrasen/0` (oder aus dem Menü mit `Analysiere Testphrasen`) die Eingabedatei lesen, in Phrasen zerlegen, die Phrasen mit dem Parser analysieren und die Ergebnisse auf die Ausgabedatei mit dem Suffix `.ana.<Grammatik>.<Parser>.<Datum>:<Uhrzeit>` schreiben:

⟨*Erkennen von Beispielphrasen*⟩+≡

```
?- analysierePhrasen.
Eingabedatei: 'Testbett/Beispiele/np.txte'.
Ausgabedatei: Testbett/Beispiele/np.txte.ana.app.epl.21.4.2002:23:2
...........................
Done.
```

An der Uhrzeit (Minute) kann man die Ergebnisse verschiedener Tests unterscheiden, sodaß man nach einer Änderung der Grammatik, des Lexikons oder des Parsers prüfen kann, ob die bisher erkannten oder analysierten Beispiele vom geänderten Programm ebenfalls erkannt oder analysiert werden. (Die Analysen muß man allerdings von Hand oder mit `ediff` vergleichen.)

Die Testdatei muß nicht im Verzeichnis `Testbett/Beispiele` stehen. Dort stehen weitere Dateien mit Nominalphrasen aus dem Referenztext `wissenschaftsreligion.txt`, nämlich `Testbett/Beispiele/np.txt` und (sortierter) `Testbett/Beispiele/np.etxt`; sie enthalten aber schon Adjektive und Präpositionalobjekte; Dateien mit den jeweiligen Ergebnissen des Erkenners und Parsers finden sich ebenfalls dort.

Einige Beispiele, die nicht funktionieren (manche funktionierten in früheren Versionen, wie man an den Ergebnisdateien sehen kann), sind:

⟨*Testbett/Beispiele/np.fehler*⟩≡

```
nichts anderem.
anderes.
eine solche Erklärung.
der Berliner Ausstellung ‘Sieben Hügel’.
Freuds Entdeckung des unbewussten.
sich selbst reproduzierende Maschinen.
menschlicher Natur und Technik.
Naturwissenschaften, technischen Wissenschaften und Ökonomie.
```

1.4 Präpositionalphrasen

Wir geben eine Beispielgrammatik für Präpositionalphrasen an und hoffen, damit die vorhandenen Beispiele in solche mit gleichen Analysen und unanalysierbare Beispiele einteilen zu können, mit denen die Grammatik verbessert werden kann.

1.4.1 Klassen- und Formparameter

In der Datei `Grammatiken/pp.cat.pl` werden die Merkmale und ihre Wertebereiche von Präpositionalphrasen definiert. Sie werden bei den Vorberechnungen der Grammatiken und später bei der Satzanalyse gebraucht. Damit sie bei den Vorberechnungen in die an die Parser angepassten Grammatiken übertragen werden, müssen sie zu Hilfsprädikaten erklärt werden:

⟨*Grammatiken/pp.cat.pl*⟩≡

```
auxiliary_predicates([definitheit_pp/1]).
```

Präpositionalphrasen klassifizieren wir nach ihrer Definitheit und der einleitenden Präposition (mit Kasus), die für die Abhängigkeit vom Verb bei Präpositionalobjekten benötigt wird:[12]

⟨*Grammatiken/pp.cat.pl*⟩+≡
```
kategorie(pp([Def],[Praep])) :-
        definitheit_pp(Def),
        praeposition(Praep).
```

Zur Vorberechung der First-Relation sollen sowohl die Definitheit wie die Form (in der die Präposition vermerkt ist) berücksichtigt werden:

⟨*Grammatiken/pp.cat.pl*⟩+≡
```
% restrict(pp([Def],Form),pp([Def],Form)) :- !.
restrict(pp([Def],Form),pp([_Def],_Form)) :- !.
```

Bei den Merkmalen wird die Definitheit zur Unterscheidung von definiten, interrogativen, und relativierenden Präpositionalphrasen wie `auf dem Stuhl`, `auf welchem Stuhl`, `auf dem` benutzt. (Beachte, daß `praeposition/1` vom Lexikon abhängt.) Mit der Definitheit werden, wie bei anderen Phrasen, Unterschiede in der Verwendung beschrieben, die u.a. für die Bestimmung der Satzart wichtig sind.

Definite Präpositionalphrasen sind z.B. `in der Grube`, `darin`, *indefinite* sind z.B. `in einer Grube`, `in Gruben`.

⟨*Grammatiken/pp.cat.pl*⟩+≡
```
definitheit_pp(def(Pers,Num)) :-
        person(Pers), numerus(Num).
definitheit_pp(indef(Pers,Num)) :-
        person(Pers), numerus(Num).
definitheit_pp(quant(Pers,Num)) :-
        person(Pers), numerus(Num).
```

Für Präpositionalobjekte von Verben, die sich reflexiv auf das Subjekt beziehen, z.B. bei `(es) mit sich bringen`, braucht man eine Reflexiv-Markierung, für die Kongruenz eines eingebetteten Reflexivpronomen mit dem Satzsubjekt:[13]

⟨*Grammatiken/pp.cat.pl*⟩+≡
```
definitheit_pp(refl(Pers,Num)) :-
        person(Pers), numerus(Num).
```

Relativierende Präpositionalphrasen wie in `der Fehler, auf den niemand achtet`, werden nach dem Genus des Bezugsnomens unterklassifiziert:

⟨*Grammatiken/pp.cat.pl*⟩+≡
```
definitheit_pp(rel(Gen,sg)) :-
        genus(Gen).
definitheit_pp(rel(unsp,pl)).
```

[12] Das Prädikat `kategorie/1` wird z.Zt. nicht benutzt und dient mehr der Dokumentation.

[13] Braucht man auch eine Markierung für Kongruenz in Person und Numerus: `Ich bin stolz (auf dich:def):def. Du bist stolz (auf dich:refl):refl.` ? Allerdings scheint das Präpositionalobjekt mit eingebettetem Personalpronomen nie indefinit sein zu können (auch bei indefinitem Subjekt). Man sollte Reflexivpronomen benutzen, um Gleichheit des Objekts mit dem Subjekt auszudrücken, unabhängig davon, ob das Verb reflexiv ist, d.h. *nur* ein mit dem Subjekt gleiches Objekt erlaubt.

Interrogative Präpositionalphrasen, z.B. `mit welchem Trick`, erfordern normalerweise keine Unterscheidung nach dem Genus; wenn die eingebettete Nominalphrase ein Interrogativpronomen ist, kann man aber nach dessen Genus unterscheiden, um den (semantisch begründeten) Unterschied im Gebrauch von `mit wem` und `mit was` nicht zu verwischen:

⟨*Grammatiken/pp.cat.pl*⟩+≡
```
definitheit_pp(qu(Gen,Num)) :-
        genusErw(Gen), numerus(Num).
```

(Eigentlich bräuchte man hier den Numerus wohl nicht mehr!)

Als Formparameter geben wir bei Präpositionalphrasen die jeweilige (ggf. zweiteilige) Präposition an. Dann können Präpositionalobjekte in Komplementrahmen durch die jeweilige Präposition angegeben werden: sie regieren die Form des Präpositionalobjekts.

⟨*Grammatiken/pp.cat.pl*⟩+≡
```
praeposition(Praep) :-
        % setof(P,word(P,pr(_,_)),Ps), % viel zu langsam
        setof(P,wrd(P,pr(_,_)),Ps),
        member(Praep,Ps).
```

1.4.2 Grammatik für Präpositionalphrasen

Eine Grammatik für Präpositionalphrasen setzt neben den Parametern der Kategorie auch die Eigenschaften und Grammatikregeln (mindestens) für Nominalphrasen (siehe Abschnitt 1.3) voraus. Sie werden von der Grammatik für Präpositionalphrasen geladen:

⟨*Grammatiken/pp.grm*⟩≡
```
:- consult('Lexika/wortarten.cat.pl').
:- ensure_loaded('Grammatiken/pp.cat.pl').
:- ensure_loaded('Grammatiken/np.cat.pl').

% Deklarartionen:

:- op(1100,xfx,'-->').

% Hilfsprädikate: (ausser denen aus den .cat-Dateien)

auxiliary_predicates([start_symbol/1,ppDefinitheit/5]).

start_symbol(pp([Def],[Form])).
```

Die Grammatikregeln sind (im Vergleich zum Lexikon) relativ einfach.

(i) Präpositionalphrasen können aus einer Präposition vor einer Nominalphrase bestehen; die Präposition regiert (normalerweise) den Kasus der Nominalphrase:

⟨*Grammatiken/pp.grm*⟩+≡
```
pp([Def],[Praep]) -->
        pr([[Kas]],[Praep]), np([DefNP,Pers,Gen],[Num,Kas]):-
%       word(Praep,pr([[Kas]],[Praep])),
%       ((Praep = als ; Praep = wie) -> Pers = 3 ; true),
%       persGenNum(Pers,Gen,Num),
%       kasus(Kas),
        ppDefinitheit(DefNP,Pers,Gen,Num,Def).
```

Die Präpositionen, die den Kasus nicht regieren, z.B. `in,an,auf`, die ja mit 2 Kasus verwendet werden, und die Präpositionen `wie,als`, die mit allen 4 Kasus verwendet werden, werden im Formmerkmal `Praep` zusammen mit einem Kasus angegeben, z.B. `auf(akk)`, also quasi als unterschiedliche Präpositionen behandelt. (Prüfen, ob das nur bei denen so gemacht ist!)

Die Definitheit der Präpositionalphrase hängt in diesem Fall von der Definitheit der Nominalphrase und deren Genus, Numerus und Person ab:

⟨*Grammatiken/pp.grm*⟩+≡
```
ppDefinitheit(DefNP,Pers,Gen,Num,Def) :-
        definitheit_np(DefNP),
        ((DefNP = rel ; DefNP = qu)
        -> Def =.. [DefNP,Gen,Num]
        ; Def =.. [DefNP,Pers,Num]).
```

(Wegen `Def =.. [DefNP,...]` muß `DefNP` ein Prolog-Atom sein, keine Variable. Daher ist `ppDefinitheit` nötig!)

⟨*Testbett/Beispiele/pp.etxt*⟩≡
```
durch die Evolution. auf einen Gedanken. durch technische Eingriffe.
als eine gesicherte Erkenntnis.
mit mir. durch dich. mit sich.
```

Fehler: `als fortschrittliche Maßnahme` geht nicht, da nur eine `np` zugelassen ist; wir brauchen auch `n1([Gen], [Komp, Adekl, sg, Kas])`.

(ii) Indefinite Präpositionalphrasen können auch aus einer Präposition und einem einfachen Nomen im Singular bestehen, wie bei `aus Eifersucht`. (Diese Regel ist unnötig, *wenn* die entsprechenden Nomenformen stets –wie im Plural– volle Nominalphrasen sind.)

⟨*Grammatiken/pp.grm*⟩+≡
```
pp([indef(3,sg)],[Praep]) -->
        pr([[Kas]],[Praep]), n([Gen],[sg,Kas]).
```

Allerdings scheinen semantisch bedingte Einschränkungen erforderlich zu sein, die hier ignoriert werden: `mit Lust/Geld/*Maschine`, `aus Eifersucht/*Geld`, zumindest bei den adverbial gebrauchten Präpositionalphrasen.

⟨*Testbett/Beispiele/pp.etxt*⟩+≡
```
von Darwin. außer Zweifel. nach Frankfurt.
```

(iii) Dieselbe Regel wie für Präposition mit Nominalphrase gilt auch für Postposition und Nominalphrase:

⟨*Grammatiken/pp.grm*⟩+≡
```
pp([Def],[Postp]) -->
        np([DefNP,Pers,Gen],[Num,Kas]), po([[Kas]],[Postp]) :-
%       word(Postp,po([[Kas]],[Postp])),
%       persGenNum(Pers,Gen,Num),
%       kasus(Kas),
        ppDefinitheit(DefNP,Pers,Gen,Num,Def).
```

⟨*Testbett/Beispiele/pp.etxt*⟩+≡
```
der Chancen wegen. seiner Behauptung nach.
```

(iv) Ebenso können Circumpositionen mit einer Nominalphrase eine Präpositionalphrase bilden:

⟨*Grammatiken/pp.grm*⟩+≡
```
pp([Def],[(Prae,Post)]) -->
        pc1([[Kas]],[(Prae,Post)]),
        np([DefNP,Pers,Gen],[Num,Kas]),
        pc2([[Kas]],[(Prae,Post)]) :-
%       word(Prae,pc1([[Kas]],[(Prae,Post)])),
%       persGenNum(Pers,Gen,Num),
%       kasus(Kas),
        ppDefinitheit(DefNP,Pers,Gen,Num,Def).
```

Wir geben die aus den zwei Bestandteilen `(Prae,Post)` bestehende Circumposition durch zwei Konstituenten mit den Kategorien `pc1` und `pc2` an und notieren den regierten Kasus bei beiden als Klassenparameter.

⟨*Testbett/Beispiele/pp.etxt*⟩+≡
```
in das 20. Jahrhundert hinein. um der Zukunft Willen.
um das Haus herum.
```

(v) Präpositionen, die mit einem Artikel verschmolzen sind, bilden mit einem artikellosen Nominalausdruck im Singular und von entsprechendem Genus eine definite Präpositionalphrase:

⟨*Grammatiken/pp.grm*⟩+≡
```
pp([def(3,sg)],[Praep]) -->
        pdet([[Kas],Gen],[Praep]), n1([Gen],[Komp,Adekl,sg,Kas])
        :- (Adekl = sw ; Adekl = -).
```

Die Adjektivdeklination der `n1` ist undefiniert oder die schwache Deklination.

⟨*Testbett/Beispiele/pp.etxt*⟩+≡
```
im Kampf gegen das Verbrechen. ins 20. Jahrhundert. am Main. im schönen München.
```

(vi) Gewisse Präpositionalphrasen können durch eine Präposition zu einer Präpositionalphrase erweitert werden, z.B. `bis an die Grenze`:

⟨*Grammatiken/pp.grm*⟩+≡
```
pp([Def],[Praep]) -->
        pr([[PraepAdvF]],[Praep]), pp([Def],[Praep2])
        :- (PraepAdvF = Praep2
           ; wrd(Praep2, pr([Kas,PraepAdvF],[Praep2])), Praep2 =.. [_]
           ; wrd(Praep3, pr([Kas,PraepAdvF],[Praep2])), Praep2 =.. [Praep3,Kas]
           ).
```

⟨*Testbett/Beispiele/pp.etxt*⟩+≡
```
bis ins 20. Jahrhundert.
bis in das 20. Jahrhundert.
bis auf Darwin.
bis zum Main.
```

Hierbei wird die regierte adverbiale Funktion nicht an der Objektphrase abgelesen, sondern über `wrd/2` an den Lexikoneinträgen ihrer Präposition. (Wenn man die adverbialen Funktionen der Präpositionalphrasen in Merkmalen speichert, erhält man eine viel zu große Mehrdeutigkeit.)

(vii) Schließlich treten Präpositionen auch vor Adverbien auf und erweitern diese zu adverbial verwendeten Präpositionalphrasen, z.B. `ab heute`: *Dazu müssen die Adverbien von Hand klassifiziert werden und die adverbiale Funktion der Präpositionen muß (durch eine Variante von* `pr`*)* *zugänglich gemacht werden.*

⟨*Grammatiken/pp.grm*⟩+≡
```
pp([Def],[Praep]) -->
        pr([[AdvF]],[Praep]), adv([Def,AdvF],[]).
%       :- adverbialeFunktion(AdvF).
```

Bem.: Das Definitheitsmerkmal solcher Präpositionalphrasen sollte vielleicht `Def(AdvF)` oder `Def,AdvF` sein. Es paßt nicht mehr zu den Deklarationen.

⟨*beispiel*⟩≡
```
1. pp([def], [bis])
      pr([[temp]], [bis])
         bis
      adv([def, temp], [])
         heute

1. pp([qu], [bis])
      pr([[temp]], [bis])
         bis
      adv([qu, temp], [])
         wann
```

Man könnte die adverbiale Funktion in das Definitheitsmerkmal der PP aufnehmen, aber das geht nicht analog bei Präpositionalphrasen wie `mit dem Messer`, wo man die möglichen adverbialen Funktionen aus der Präposition herleiten kann.

(viii) Definite Pronominaladverbien und Konjunktionaladverbien, die wir im Lexikon als Pro-Präpositionalphrasen klassifiziert haben, lassen als Ergänzungen einen Verbletztsatz zu, z.B. `darauf, daß jemand hilft,` oder `deswegen, weil niemand half,`.

Diese Phrasen gelten als Präpositionalphrasen, damit man bei den Verben die entsprechenden Objekte als Präpositionalobjekte angeben kann: `word(hoffen, v([ [nom,auf] ],Form))` erlaubt dann sowohl `auf Hilfe hoffen`, `worauf hoffen`, `darauf hoffen` als auch `darauf, daß jemand hilft, hoffen`.

⟨*Grammatiken/pp.grm*⟩+≡
```
pp([def(3,sg)],[Praep]) -->
        proPP([def(3,sg)],[Praep]), ?s([daß],[Temp,Mod]).
```

⟨*beispiel*⟩+≡
```
darauf, daß niemand geht.
damit, daß ihr kommt.
```

⟨*Grammatiken/pp.grm*⟩+≡
```
pp([qu(unsp,sg)],[Praep]) --> proPP([qu(unsp,sg)],[Praep]).
```

⟨*Beispiele/pp.txt*⟩≡
```
worauf. weswegen.
```

(ix) Hilfsweise wird eine Löschregel angegeben, damit `del/1` existiert (sonst bekommt man einen Fehler):

⟨*Grammatiken/pp.grm*⟩+≡
```
loeschbar --> [].

?s([daß],[Temp,Mod]) --> (s([daß],[Temp,Mod]) ; {true}).
?ergaenzung([Komp]) --> (ergaenzung([Komp]) ; {true}).
?pp([Def],[als(nom)]) --> (pp([Def],[als(nom)]) ; {Def = -}).
```

Zum Testen dieser Grammatik braucht man ein paar Regeln für eingebettete Phrasen. (Ohne gewisse Teilphrasengrammatiken macht die Vorberechnung hier keinen großen Sinn!)

Beispiel 1.3 Wir kombinieren die obige Grammatik mit der vorausgegangenen Beispielgrammatik für Nominalphrasen. Das Prädikat `auxiliary_predicates/1` wird in verschiedenen Dateien teilweise definiert und muß daher besonders deklariert werden:

⟨*Grammatiken/pp1.grm*⟩≡
```
% :- multifile auxiliary_predicates/1. % schon in Grammatiken/np.grm
```
  ⟨*Grammatiken/np.grm*⟩
  ⟨*Grammatiken/pp.grm*⟩

Zu beachten ist, daß im allgemeinen die Hilfsprädikate von verschiedenen Teilgrammatiken benötigt werden, aber nur einmal geladen und in die vorberechnete Grammatik exportiert werden dürfen.

1.4.3 Extraktion von Beispielen

Zu den Beispieltexten `Testbett/*.txt` wurden mit Werkzeugen des CISLEX von Petra Maier Vollformenlexika erstellt. Einige Päpositionalphrasen des Texts `wissenschaftsreligion.txt` sind in einer Datei `Testbett/Beispiele/pp.txt` gesammelt.

⟨*Testbett/Beispiele/pp.txt*⟩≡
```
als fortschrittliche Maßnahme im Kampf gegen das Verbrechen.
als gesicherte wissenschaftliche Erkenntnis.
als Bedrohung und als glückliche Chance.
als falsch.
anders als bei Platon.
als wissenschafts- und technikfeindlich.
spannender als viele Romane oder Filme.
als eine menschliche Problemlösungspraxis.
als eine ideale Form, um etwas zu tun.
größere Gefahren als die einzelnen Ergebnisse der Wissenschaft selbst.
als mit Freuds Entdeckung des Unbewussten.
eine größere Rolle als bislang angenommen wurde.
er bestehe letztlich in nichts anderem als verschiedenen Versuchen zur Optimierung.
als Arena für sein Balzverhalten.
als biologische Wesen in der natürlichen Welt und als Kulturwesen in einer Welt symbolischer Bedeutungen.
als ein erfolgreiches Mittel der Evolution.
durch das.
nichts als eine Illusion.
sich wechselseitig als Personen zu deuten.
als Personen.
die Deutung des Menschen als freie, verantwortliche Person.
die Deutung des Menschen als eine verantwortliche Person.
als Geschlechtswesen.
als determinierte Wesen.
als eine verantwortliche Person.
als verantwortliche Personen.
als "Züchtungsregeln".
als Züchtungsregeln.
in der sie nichts aussagen als sich selbst.
an der Endlichkeit.
```

an den pädagogischen und sozialhistorischen Ausstellungen.
an den Ende des 19. Jahrhunderts ausgetragenen Streit.
an kulturell vermittelten Kriterien.
an diese Illusion.
an der Vorstellung eines sich selbst reproduzierenden Roboters.
an naturwissenschaftlichen Erklärungen und technischen Eingriffsmöglichkeiten.
an dem großen Zulauf.
an einem Kunstobjekt.
an diesem zufälligen Ort.
an der Universität Frankfurt am Main.
angesichts der Berliner Ausstellung ‘Sieben Hügel’.
angesichts der Bedrohungsszenarien.
auf wissenschaftliche Autorität.
auf eine auffallende Bereitschaft, die Strapazen zu verdammen.
auf Ethnien, Gesellschaften und Kulturen am Anfang des 20. Jahrhunderts.
auf den logischen Fehler.
auf ihre Funktion im Wettbewerb um die günstigsten Fortpflanzungschancen.
auf den Gedanken.
auf Natur.
Auf dieses Symbolsystem.
auf die Deutung des Menschen als freie, verantwortliche Person.
auf die Zustimmung gleicher und freier Staatsbürgerinnen und Staatsbürger.
auf genetisch determinierte, den Gesetzen der Soziobiologie unterworfene Wesen.
die Seiten, auf denen sie angeboten werden.
auf denen.
auf seine Funktion für die Evolution.
aus dem Thriller und aus dem Neuen Testament.
aus einem Sein ein Sollen, aus der Natur verbindliche Verhaltensnormen herleiten.
aus der Soziobiologie.
aus der Perspektive eines naturwissenschaftlichen Beobachters.
aus Freiheit.
aus unserer Selbstdeutung.
aus naturwissenschaftlicher Sicht.
dass diese Verhaltensweisen ausschließlich damit erklärt werden können.
dass wir uns ausschließlich als determinierte Wesen verstehen.
wo wir unser Selbstverständnis ausschließlich an naturwissenschaftlichen
Erklärungen und technischen Eingriffsmöglichkeiten ausrichten.
außer Zweifel.
bei Platon.
bei solchen popularisierenden Schlussfolgerungen.
bei weitem übertreffen.
bei Abwesenheit von Zwang.
durch die Wissenschaft.
durch Männer.
durch technische Eingriffe.
durch Kunst.
durch den Prestigegewinn.
durch Roboter oder sich selbst reproduzierende Maschinen.
durch Maschinen.
durch die Sprache.
durch Kommunikation.
ein erfolgreiches Mittel der Evolution, durch das sie es beeinflussen.
durch eine solche Erklärung.

```
durch denselben Prozess, der dazu führt.
durch eine göttliche Strafe.
mithilfe von Wissenschaft und Technik.
zwischen Virchow und Haeckel.
zwischen Naturwissenschaften, technischen Wissenschaften und Ökonomie.
zwischen sozialen Veränderungen und wissenschaftlicher Erkenntnis.
zwischen Experten und Gutachtern.
zwischen Hirnforschern, Psychologen und Neurophilosophen.
zwischen menschlicher Natur und Technik.
von dem methodischen Zweifel.
vom methodischen Zweifel.
```

Davon können manche schon mit der Grammatik analysisert werden, andere nicht. Neben den wegen der Koordination ausgeschlossenen Fällen handelt es sich hauptsächlich um Probleme mit `als` vor `n1`-Ausdrücken und Vorkommen von `sich` oder `selbst`:

⟨*Negativbeispiele*⟩≡

```
0: als fortschrittliche Maßnahme im Kampf gegen das Verbrechen. - Ungrammatisch.
1: als gesicherte wissenschaftliche Erkenntnis. - Ungrammatisch.
9: größere Gefahren als die einzelnen Ergebnisse der Wissenschaft selbst. - Ungrammatisch.
13: als Arena für sein Balzverhalten. - Ungrammatisch.
16: durch das. - Ungrammatisch.
20: die Deutung des Menschen als freie, verantwortliche Person. - Ungrammatisch.
28: in der sie nichts aussagen als sich selbst. - Ungrammatisch.
31: an den Ende des 19. Jahrhunderts ausgetragenen Streit. - Ungrammatisch.
32: an kulturell vermittelten Kriterien. - Ungrammatisch.
34: an der Vorstellung eines sich selbst reproduzierenden Roboters. - Ungrammatisch.
39: an der Universität Frankfurt am Main. - Ungrammatisch.
42: auf wissenschaftliche Autorität. - Ungrammatisch.
50: auf die Deutung des Menschen als freie, verantwortliche Person. - Ungrammatisch.
54: auf denen. - Ungrammatisch.
62: aus naturwissenschaftlicher Sicht. - Ungrammatisch.
70: bei Abwesenheit von Zwang. - Ungrammatisch.
81: durch eine solche Erklärung. - Ungrammatisch.
```

2 Anhang 4: Beispieltext

Der Referenztext in `Testbett/wissenschaftsreligion.txt` ist ein Artikel aus DIE ZEIT, Nr. 27, 2000. Die dazu erstellten Lexika finden sich (unbearbeitet) nach Wortarten gegliedert in `Testbett`, die überprüften und teilweise geänderten Einträge im Verzeichnis `Lexika`.

Heiliges Erschauern

Über die Heraufkunft einer neuen Wissenschaftsreligion und ihre Gefahren

von Klaus Günther

Am Ende der Weimarer Republik bereitete die sozialdemokratische Reichstagsfraktion einen Gesetzentwurf vor, der für rückfällige Straftäter, so genannte „Gewohnheitsverbrecher“, die freiwillige Sterilisation vorsah. Als einige Zeit später die Nazis ein Gesetz über die zwangsweise Sterilisation von Gewohnheitsverbrechern verabschiedeten, wurde dies von der Auslands-SPD vorbehaltlos als fortschrittliche Maßnahme im Kampf gegen das Verbrechen begrüßt. Dass Kriminalität erblich sei, galt als gesicherte wissenschaftliche Erkenntnis.

Trotz vereinzelter Bemühungen ist es bisher nicht gelungen, ein „Kriminalitätsgen“ ausfindig zu machen. Heute würde man vermutlich auch nicht zu einer so einschneidenden Maßnahme wie einer Sterilisation schreiten, sondern vielleicht einen genchirurgischen Eingriff versuchen. Indes war die Vermutung, Kriminalität sei erblich, schon damals nicht so sicher bestätigt, wie diejenigen glaubten, die daraus politische Schlussfolgerungen für die Strafgesetzgebung zogen. Doch die Autorität der Naturwissenschaften stand außer Zweifel, und ihre eindrucksvollen Fortschritte, vor allem in der von Darwin begründeten Evolutionstheorie, ließen der Politik scheinbar keine andere Wahl.

Eine noch viel stärkere Überwältigung der Öffentlichkeit durch die Wissenschaft scheint gegenwärtig stattzufinden. Wiederum sind es neue, in rascher Folge gewonnene Erkenntnisse über die Natur des Menschen und Techniken ihrer Veränderung, welche ein heiliges Erschauern hervorrufen. Neben Humangenetik und Hirnforschung, Soziobiologie und Evolutionspsychologie treten die beeindruckenden Leistungen der Computertechnik, der Nanotechnologie und jener Wissenschafts- und Technikzweige, welche beide Entwicklungen miteinander verknüpfen, also Biophysik und -informatik, die Entwicklung intelligenter Software oder die Robotik. Deren Möglichkeiten erscheinen zugleich als Bedrohung und als glückliche Chance.

Wiederum sind es vor allem die auf wissenschaftliche Autorität gestützten Schlussfolgerungen, welche die Zukunft bedrohlich erscheinen lassen. Wenn die Soziobiologie Vergewaltigungen von Frauen durch Männer mit einer genetischen Verhaltensprogrammierung erklärt, ist dann nicht jeder Vergewaltiger unschuldig? Wenn Menschen ihr Erbgut durch technische Eingriffe verändern oder gar im Wege der Klonierung verdoppeln können, muss man sich dann nicht vor den Missbrauchsgefahren fürchten und Kontrollen einführen? Sind solche Regeln nicht eigentlich Züchtungsregeln für Menschen? Wenn ein Psychotiker wie der „Unabomber“ die Nanotechnik in die Hände bekommt - droht dann der Welt nicht eine Katastrophe, gegen die sich die Pläne der notorisch Bösen in jedem James-Bond-Film geradezu idyllisch ausnehmen?

‘Wir sind leichtgläubig gegenüber der Autorität der Wissenschaft’

Gewiss wäre es falsch, solche Gefahren herunterzuspielen oder gar zu leugnen, wie umgekehrt die Augen vor den neuen Möglichkeiten zu verschließen. Irritierend sind jedoch die Leichtgläubigkeit und Leichtfertigkeit, mit welchen Wissenschaft und Technik Autorität zugeschrieben werden. Das gilt auch noch für jene radikalen Gegner, die hinter jeder soziobiologischen Forschungshypothese gleich Faschismus wittern und daher die Soziobiologie am liebsten ganz verbieten möchten. Die Kardinaltugend der Wissenschaft ist jedoch die

Skepsis, denn nur der beständige Zweifel motiviert zur Überprüfung scheinbar gesicherter Erkenntnisse, bis sie sich als falsch herausstellen.

Leider wird diese Tugend oftmals auch von der Wissenschaft selbst nicht beachtet, wenn sie euphorisch oder furchtsam von den ungeahnten Möglichkeiten der Zukunft schwärmt. Lässt dieses Staunen, anders als bei Platon, die Kritik verstummen? Die wissenschaftlichen und technischen Fortschritte treffen jedenfalls auf eine auffallende Bereitschaft, die Strapazen der Kritik nicht mehr auf sich zu nehmen oder Bedenken als wissenschafts- und technikfeindlich zu verdammen. „Wissenschaft“, so war kürzlich zu lesen, „ist spannend wie ein Thriller, spannender als viele Romane oder Filme.“ Ja, Wissenschaft kann sehr spannend sein - aber warum soll die Selbstauslegung des Menschen in Literatur und Film weniger spannend sein? Weil Kunst dem Betrachter die Fragwürdigkeit seiner eigenen Selbst- und Weltdeutung offenbart und Gewissheiten relativiert? Die gläubige Bejahung der Wissenschaft räumt dagegen mit Selbstzweifeln auf. Warum soll man sich durch Kunst ins Leiden an der Endlichkeit treiben lassen, wenn man sich doch klonen kann?

Das geht freilich nur, wenn dabei die Wissenschaft selbst sakralisiert und zu einer säkularen Religion erhoben wird. Dann werden einfach ihre kulturellen und politischen Relativierungen abgestreift, und die konstruktivistische Natur der Wissenschaft als eine menschliche Problemlösungspraxis wird unsichtbar gemacht. Ulrich Raulff hat in der ‘FAZ’ angesichts der Berliner Ausstellung ‘Sieben Hügel’ die nötigen Stichworte geliefert. Endlich würden die Wissenschaftsobjekte dieser Ausstellung kontextfrei präsentiert - „zur Freude all derer, die sich an den pädagogischen und sozialhistorischen Ausstellungen satt gesehen haben ... Zugleich erscheint sie als eine ideale Form, um eine Wissenschaftsreligion zu zelebrieren, die endlich die kulturellen Ketten des neunzehnten und die politischen Ideologien des zwanzigsten Jahrhunderts abgeschüttelt hat und im Begriff ist, eine reine Religion zu werden: die stärkste Form der Bindung und Bündelung geistiger Energie.“

Das ist die Zumutung einer Paradoxie. Die Wissenschaft, die vom methodischen Zweifel lebt, die sich nur der Bemühung um Falsifikation verdankt, soll fraglos hingenommen und von jeglichem Zweifel, der ja nur ein törichter wäre, gereinigt werden. Die Vernunft soll ihre Ansprüche aufgeben; die Gläubigen sollen demütig ihr „Amen“ beten. Demgegenüber scheint die kulturelle, politische und sozialhistorische Reflexion der Wissenschaft langweilig. Sie macht aus dem Thriller eine trockene Abhandlung und aus dem Neuen Testament eine historische Untersuchung über das Leben Jesu. So wird im Zeitalter der Globalisierung Wissenschaft zur neuen Religion, die unser Selbstverständnis angreift, um es zu verändern.

Dabei könnte schon eine einfache Überlegung davon überzeugen, dass es die reine Naturwissenschaft ohne sozialhistorischen Kontext nicht gibt: Einer der wichtigsten Paradigmenwechsel in der Evolutionstheorie steht in einem auffälligen Zusammenhang mit sozialhistorischen Veränderungen. Es dürfte nämlich kein Zufall sein, dass die Evolutionstheorie die Erhaltung der Art zum wichtigsten Selektionskriterium zu einer Zeit erhob, als der Nationalismus den Vorrang des Kollektivs vor dem Einzelnen forderte. Und wenn heute demgegenüber behauptet wird, die treibende Kraft der menschlichen Evolution sei allein die Weitergabe der individuellen Gene, so dürfte ein Zusammenhang mit dem gegenwärtig dominierenden Individualismus der Ökonomie nicht ganz fern liegen.

Man denke nur an den Ende des 19. Jahrhunderts ausgetragenen Streit zwischen Virchow und Haeckel über den „demokratischen“ oder „autoritären“ Aufbau der Zelle. An dieser Kontroverse lässt sich nachvollziehen, wie sehr die Naturwissenschaften bis in die Wahl ihrer Metaphern den kulturellen und sozialhistorischen Kontexten verhaftet sind. Auch bieten die Verflechtungen zwischen Naturwissenschaften, technischen Wissenschaften und Ökonomie genügend Anlass, kontextorientiert über die Wissenschaft zu denken. Gewiss löst sich Wissenschaft dabei nicht in Sozialhistorie auf, und der Nachweis eines genetischen Zusammenhangs zwischen sozialen Veränderungen und wissenschaftlicher Erkenntnis berührt deren Wahrheitsgehalt nicht. Aber allein schon der Streit zwischen Experten und

Gutachtern über die politisch relevanten Fragen der Wissenschaft sollte uns vor einer religiösen Einstellung bewahren.

Möglicherweise birgt die Wissenschaftsreligion größere Gefahren als die einzelnen Ergebnisse der Wissenschaft selbst. In der Geschichte sind es nämlich stets die Wissenschaftsgläubigen gewesen, die Popularisierer und großen Vereinfacher, die sich zu leichtfertigen Schlussfolgerungen hinreißen ließen. Das eindrucksvollste und zugleich schrecklichste Beispiel ist die simple Übertragung der Darwinschen Evolutionstheorie auf Ethnien, Gesellschaften und Kulturen am Anfang des 20. Jahrhunderts. Vor allem zwei Probleme werden bei solchen popularisierenden Schlussfolgerungen leicht übersehen: der so genannte „naturalistische Fehlschluss“ und der Reduktionismus. Der Philosoph David Hume und mit ihm die Aufklärung hatten auf den logischen Fehler aufmerksam gemacht, den begeht, wer aus einem Sein ein Sollen, aus der Natur verbindliche Verhaltensnormen herleiten will. Ob eine Verhaltensweise oder ein biologischer Vorgang „normal“ ist, ist eine Frage der Kultur und nicht der Natur. Oder mit einem Beispiel aus der Soziobiologie: Selbst wenn die Behauptung zutreffen sollte, dass Männer in der Evolution ein Verhaltensprogramm ausdifferenziert haben, das eine Disposition zur Vergewaltigung von Frauen erzeugt, folgt daraus nicht, dass es „normal“ ist, wenn Männer Frauen vergewaltigen. Dem voraus geht die Setzung einer Norm, die sich an kulturell vermittelten Kriterien des Nützlichen und Schädlichen, Gerechten und Ungerechten orientiert.

Die Suggestionskraft wissenschaftlicher Entdeckungen verführt oft dazu, mit ihnen alles und jedes zu erklären. Das war mit Darwins Evolutionstheorie nicht anders als mit Freuds Entdeckung des Unbewussten. Wenn die Evolutionspsychologie bestimmte Verhaltensweisen der Menschen auf ihre Funktion im Wettbewerb um die günstigsten Fortpflanzungschancen zurückführt, bedeutet das weder, dass diese Verhaltensweisen ausschließlich damit erklärt werden können, noch, dass sämtliche menschliche Verhaltensweisen allein darin ihre Ursache haben. Es mag ja sein, dass dieser Wettbewerb eine größere Rolle in der Evolution von Kulturen spielt, als bislang angenommen wurde. Der Variantenreichtum menschlicher Kulturäußerungen wird jedoch schlicht verkannt, wenn von ihm behauptet wird, er bestehe letztlich in nichts anderem als verschiedenen Versuchen zur Optimierung von Fortpflanzungschancen.

Vielleicht wurde ein Physiker zur nobelpreiswürdigen Lösung eines Problems der Quantentheorie durch den Prestigegewinn angetrieben, den er sich davon im Kampf der Männer um die Frauen mit den „besten“ Genen erhoffte. Doch damit ist weder erklärt, warum dieser Mann die Quantenphysik und nicht die Evolutionspsychologie als Arena für sein Balzverhalten wählte; noch lässt sich entscheiden, ob seine quantentheoretischen Lösungen wahr oder falsch sind. Und alle Versuche einer monokausalen Erklärung scheitern spätestens dann, wenn es um singuläre Ereignisse menschlichen Verhaltens geht. Die Evolutionspsychologie vermag wichtige Einsichten zur Erklärung kriegerischen Verhaltens liefern. Aber damit lässt sich nicht vollständig erklären, warum 1618 der Dreißigjährige Krieg ausbrach.

‘Was der Mensch ist, lässt sich nicht von außen bestimmen’

Ein ähnlicher Reduktionismus liegt auch den Befürchtungen über eine Entmachtung des Menschen durch Roboter oder sich selbst reproduzierende Maschinen zugrunde. Es wäre falsch, die Gefahren dieser technologischen Entwicklungen zu leugnen. Aber die Behauptung, Maschinen könnten Menschen ersetzen, setzt voraus, dass man vorher festgelegt hat, wodurch der Mensch bestimmt sei. Gewisse mathematische Fähigkeiten des Menschen lassen sich schon jetzt durch Maschinen ersetzen und sogar bei weitem übertreffen. Aber deshalb ist noch niemand auf den Gedanken gekommen, menschliche Intelligenz auf diese mathematischen Fähigkeiten zu reduzieren, um dann anschließend zu behaupten, der ganze Mensch würde in Zukunft durch Maschinen ersetzt werden.

Was den Menschen ausmacht, lässt sich nicht allein von außen bestimmen. Menschen leben nämlich immer in zwei Welten zugleich - als biologische Wesen in der natürlichen

Welt und als Kulturwesen in einer Welt symbolischer Bedeutungen, die vor allem durch die Sprache repräsentiert und durch Kommunikation vermittelt wird. Zu dieser Welt gehören auch die Selbstdeutungen der Menschen, mit denen sie bestimmen, wer sie sind und wer sie sein wollen. Die Frage, wie sich diese zwei Welten zueinander verhalten, ist eines der hartnäckigen und vielleicht auch unlösbaren Probleme der Philosophie, die gegenwärtig vor allem zwischen Hirnforschern, Psychologen und Neurophilosophen diskutiert werden.

Gewiss lassen sich kulturelle Symbolsysteme ihrerseits wiederum „biologisch erklären“ - vielleicht als ein erfolgreiches Mittel der Evolution, durch das menschliche Gehirne sich wechselseitig so beeinflussen, dass sie die höherstufigen Strukturen erwerben, die ihnen ein Überleben in einer hochkomplexen Umwelt ermöglichen. Aber von einer solchen Erklärung führt kein Weg zu dem kulturellen Bedeutungsgehalt selbst. Was die Betroffenen jeweils unter einem bestimmten kulturellen Symbol verstehen, von welchem sie in ihren sozialen Beziehungen Gebrauch machen, erschließt sich durch eine solche Erklärung nicht. Wir brauchen sie auch nicht, wenn wir verstehen wollen, wie Menschen sich selbst und die Welt, in der sie leben, deuten. Ein Mann, der einer Frau seine Liebe erklären will, scheitert, wenn er sich dabei ohne Ironie des soziobiologischen Vokabulars bedient. Einer Frau, die vor der Entscheidung steht, ob sie ihre Schwangerschaft abbrechen will oder nicht, ist nicht damit geholfen, dass ein Verhaltensbiologe sie über die „wahren“ Ursachen ihrer Entscheidung aufklärt und ein Hirnforscher nach Untersuchung ihrer Gehirnstrukturen das Verhalten voraussagt. Nun mag Freiheit aus der Perspektive eines naturwissenschaftlichen Beobachters nichts als eine Illusion sein, und es ist müßig, darüber zu spekulieren, ob es so etwas wie einen Anfang ohne Ursache, eine Wirksamkeit aus sich selbst in der Natur geben kann oder nicht. Nach allem, was wir wissen, gibt es sie nicht. Sobald Menschen aber miteinander kommunizieren, einander ein Versprechen geben, eines Fehlverhaltens beschuldigen oder einander verzeihen, können sie nicht umhin, an diese Illusion zu glauben und sich wechselseitig als Personen zu deuten und zu behandeln, die anders hätten handeln können und daher auch fähig und willens sind, einander Rechenschaft über ihr Handeln in der Weise zu geben, als ob es ihr jeweiliges Handeln wäre - und nicht das Resultat von Naturprozessen.

Auch wenn sich die Kultur der Selbst- und Weltdeutung nicht vollständig auf Natur reduzieren lässt, so bleibt es doch zumindest eine offene Frage, wie sich diese Kultur unter dem Einfluss einer Wissenschaftsreligion verändern wird. Zwei gegenläufige Entwicklungen zeichnen sich ab: Auf der einen Seite nehmen die Verfügungsmöglichkeiten des Menschen über seine eigene Natur und die seiner Nachkommen erheblich zu. Zu unserer gegenwärtigen Kultur der Welt- und Selbstdeutung gehört es nun aber, dass wir uns wechselseitig als Personen deuten, die über Wahlmöglichkeiten verfügen und - bei Abwesenheit von Zwang - auch frei sind. Wer sich in dieser Situation für die eine Handlung entscheidet, hätte sich auch für die andere entscheiden, also anders handeln können. Weil wir einander wechselseitig diese Freiheit zuschreiben, können wir uns auch verantwortlich machen.

In dem Maße aber, wie es uns möglich wird, menschliche Gene zu verändern, hat die genetische Ausstattung eines Menschen für uns nicht mehr die Bedeutung eines hinzunehmenden Schicksals. Wenn wir sie verändern können, dann müssen wir auch die Verantwortung dafür übernehmen, ‘ob’ wir sie verändern oder nicht. Auf dieses Symbolsystem, auf die Deutung des Menschen als freie, verantwortliche Person, beziehen sich auch jene Wissenschaftler, die wie Bill Joy nach einer Verantwortung von Wissenschaft und Technologie rufen. Mit der Ausdehnung unserer Verfügungsmöglichkeiten über die innere und äußere Natur wächst also auch die Zurechnung der Folgen zur individuellen Verantwortung - und dann auch die Haftung für zurechenbare schädliche Folgen - ins Unermessliche.

Es könnte jedoch sein, dass diese Kultur der Verantwortung durch denselben Prozess untergraben wird, der zu der gewaltigen Steigerung unserer Möglichkeiten führt. Es geht nicht nur um das Problem, ob es der Menschheit gelingen wird, die Folgen von Wissenschaft und Technik zu beherrschen. Fraglich ist auch, ob wir unsere heute praktizierte

Kultur der Verantwortung auch künftig noch fortsetzen können, weil wir inzwischen unser Selbstverständnis so verändern, dass wir uns gar nicht mehr als verantwortliche Personen verstehen. Das erscheint vor allem dann fraglich, wenn wir den Predigern der Wissenschaftsreligion folgen und die Forschungsergebnisse umstandslos in unser Selbstverständnis übernehmen: wenn wir zum Beispiel die Behauptung der Evolutionsspsychologie so übernehmen, dass wir uns wechselseitig als Geschlechtswesen deuten, ständig besorgt, ihre Position im Wettbewerb um die besten Gene für die Fortpflanzung zu verbessern. Wenn wir den Forschungen der Soziobiologie in der Weise folgen, dass wir uns ausschließlich als determinierte Wesen verstehen. Schließlich auch, wenn wir mit der Nanotechnologie die Grenze zwischen menschlicher Natur und Technik verwischen oder in immer größerem Umfang menschliche Gene zur Produktion von Ersatzteillagern verwenden.

Jürgen Habermas hat bereits die Frage gestellt, ob ein menschlicher Klon sich selbst noch als eine verantwortliche Person wird verstehen können, wenn er einen Designer für seine genetische Zusammensetzung verantwortlich machen muss. Wie berechtigt diese Frage ist, wird spätestens an der Vorstellung eines sich selbst reproduzierenden Roboters deutlich. Würde es selbstverständlich sein, diesen Roboter für seine „Handlungen“ verantwortlich zu machen und vor einem Strafgericht anzuklagen? Oder würde uns das ähnlich merkwürdig erscheinen wie die Strafaktion des persischen Königs Xerxes, der das Meer auspeitschen ließ?

Das Konzept einer verantwortlichen Person gehört zu den zentralen Fundamenten einer demokratischen Verfassung. Schon immer lautete das wichtigste Argument der Demokratiegegner, bestimmte Menschen müssten von der politischen Meinungs- und Willensbildung ausgeschlossen werden, weil sie unfähig seien, verantwortlich zu entscheiden. Doch die Gründung der Gesetzgebung und des Staates auf die Zustimmung gleicher und freier Staatsbürgerinnen und Staatsbürger wird nur dann zu einer hinreichenden Legitimation, wenn diese sich wechselseitig als Personen deuten, die aus Freiheit entscheiden und Verantwortung übernehmen können. Sobald wir mit der Freiheit auch die Verantwortung aus unserer Selbstdeutung streichen, verliert eine demokratische Legitimation des Staates ihren Sinn.

‘Werden Menschenrechte bald zu Züchtungsregeln?’

Es ist daher widersprüchlich, wenn Bill Joy angesichts der Bedrohungsszenarien eine Ethik der Wissenschaft fordert. Sein Ruf geht ins Leere, wenn wir uns nicht mehr als verantwortliche Personen verstehen können, wenn Verantwortung und Freiheit nicht mehr in unserer gemeinsam geteilten Selbstdeutung vorkommen. Dann wird es auch sinnlos, Wissenschaft und Technik über Normen steuern zu wollen. Diese Konsequenz droht aber in dem Maße, wie wir wissenschaftliche Ergebnisse so in unsere Selbstdeutung übernehmen, dass wir uns auf genetisch determinierte, den Gesetzen der Soziobiologie unterworfene Wesen reduzieren. Der Vorschlag, Werte und Normen des Humanismus als „Züchtungsregeln“ zu verstehen, nach denen Menschen selektiert werden, ist ein Schritt in diese Richtung. Zwischen den Regeln, welche die Bildung eines autonomen Gewissens ermöglichen, und Regeln für die Auswahl reproduktionswürdiger Gene wird dann kein Unterschied gemacht.

Wenn wir anfangen, Menschenrechte als Züchtungsregeln zu verstehen, haben wir sie schon abgeschafft. Es geht nicht darum, ob dieser oder jener genchirurgische Eingriff zulässig ist. Die Gefahren beginnen dort, wo wir unser Selbstverständnis ausschließlich an naturwissenschaftlichen Erklärungen und technischen Eingriffsmöglichkeiten ausrichten. Was sollten denn Freiheit und Verantwortung aus naturwissenschaftlicher Sicht bedeuten?

Es wird schwierig werden, die Kultur der Verantwortung gegen das neue Priestertum der Wissenschaftsreligion aufrechtzuerhalten. Wer sich im Internet einmal die Seiten angeschaut hat, auf denen weibliche Eizellen und männliche Samenzellen zur Versteigerung angeboten werden, lässt sich vielleicht noch verstören von den Werbetexten, die den evolutionären Vorteil schöner Körper anpreisen. Die Bilder der Menschen, die sich dafür hergeben, zeigen die glatten, durchtrainierten Körper der Werbewelt, denen vor allem eines gemeinsam ist: ihre Verwechselbarkeit und Austauschbarkeit. Der Mensch, der sich selbst auf seine Funktion für die Evolution reduziert, wird zu einem schönen Niemand. Dass es allerdings immer noch viele Menschen gibt, die so nicht leben wollen, kann man an dem großen Zulauf sehen, den Kunstgalerien wie die Tate Modern in London derzeit haben. Ja, auch Kunst kann man evolutionsbiologisch erklären - aber das interessiert die Kunstbetrachter in dem Augenblick nicht, wo sie an einem Kunstobjekt erfahren, dass sie deutende und verstehende Personen sind. In London kann man auch die Bilder des englischen Malers Lucien Freud sehen, die zumeist nackte menschliche Körper darstellen, in welche sich Spuren einer individuellen Lebensgeschichte eingezeichnet haben und die allen Maßstäben der Werbewelt widerstreiten. In ihrer bloß raumfüllenden Anwesenheit, in der sie nichts aussagen als sich selbst, so wie sie als gelebte Körper in diesem Augenblick an diesem zufälligen Ort da sind - daraus beziehen sie ihre Würde. Es ist die Würde von Menschen, deren Körper endlich ist und zugleich eine einmalige, unwiederholbare und deshalb unendliche Lebensgeschichte ausdrückt. Für diese Unendlichkeit geht uns vielleicht der Sinn verloren, je mehr wir uns mithilfe von Wissenschaft und Technik der körperlichen Unendlichkeit annähern. Nach ihr sehnten sich alle Untoten der Literatur, die durch eine göttliche Strafe dazu verdammt waren, nicht sterben zu können.

Klaus Günther ist Professor für Rechtstheorie und Strafrecht an der Universität Frankfurt am Main

DIE ZEIT, 27/2000

0 Seminarplan Satzgrammatik WS 2003/04

0.1 Organisation

(i) Mailing-Liste zur Diskussion von Problemen

(ii) `svn`-Archiv der Programme (und Doku), über das Netz zugänglich

(iii) 1. Vorbesprechung der Aufgabe, 2. Vorbesprechung *vor* der Implementierung

(iv) Rechtzeitiges Melden bei Problemen, Verbesserungsvorschläge

(v) Seminarvortrag mit Erläuterung des Problems, der Lösungsidee und Demobeispielen

0.2 Vorhandene Grammatik und Parser ($\geq$ 3 Termine, H.L.)

Bestandsaufnahme der bisherigen Grammatik

(i) Lexikalische und syntaktische Kategorien und ihre Merkmale

(ii) Lexikalische Information aus dem CISLEX und eigenen Stammlexika

(iii) Kodierung von Valenz und Komplementrahmen

(iv) Format der Grammatikregeln und Regelschemata

(v) Kodierung von Rektion und Kongruenz

(vi) Definitheitsmerkmal zur Unterscheidung von definiten, interrogativen, relativierenden Phrasen

(vii) Beispiel der Grammatiken für Nominal- und Präpositionalphrasen

Bestandsaufnahme und Bedienung der vorhandenen Tabellenparser

(i) Auswahlmöglichkeiten von Grammatiken und Parsern

(ii) Übersetzung einer Grammatik (Erzeugung eines speziellen Parsers, Optionen)

(iii) Aufruf von Tokenizer, Lexer, und Parser

(iv) Bedeutung der Analysetabelle und Fehlersuche

0.3 Seminarvorträge zu syntaktischen Phänomenen (Thema aussuchen!)

Aufgabe jeweils: (wenig Literatur, mehr Eigenarbeit mit entsprechender Unterstützung)

(i) Suchen von Beispielsätzen im Beispieltext (oder mit dessen Vokabular)

(ii) Lesen entsprechender Teile einer Standardgrammatik (Helbig/Buscha) oder HPSG-Literatur

(iii) Erstellen geeigneter Grammatikregeln (mit Kongruenz-, Rektions-, Ordnungsbedingungen)

(iv) Erstellen bzw. Korrigieren der Lexikoneinträge

(v) Testen der Beispiele (und Gegenbeispiele) mit dem Parser

Überarbeitung vorhandener Regeln

(i) Regeln für einfache Sätze im Aktiv (mit Vollverben): (a) Kopf-Komplement-Schemata, (b) Anordnung von Konstituenten, (c) Definitheitsbedingungen

(ii) Relativ- und Fragesätze (Überprüfen und Korrigieren)

(iii) Adjektivphrasen (Klassifikaton und Merkmale)

(iv) Prädikativsätze (im Positiv und Komparativ)

(v) Sätze mit Präfixverben

(vi) Komplement- und Adverbialsätze (untergeordnete Teilsätze)

Ausdehnung der syntaktischen Abdeckung

(i) Sätze mit reflexiven (und reziproken) Verben

(ii) Sätze im Aktiv mit 'synthetischem Tempus', d.h. Perfekt, Futur

(iii) Sätze im Passiv (Präsens, Imperfekt),

(iv) Infinitivsätze (einfache Formen)

(v) Sätze mit Anhebungs- und Kontrollverben

(vi) Korrelate für Objektsätze und Adverbialsätze

(vii) Partizipien und ihre Objekte (als Attribut und Adverbial)

(viii) Vergleichskonstruktionen (einfache Fälle)

Experimentelle Themen

(i) Satzjunktoren und Zerlegung in unabhängige Teilsätze

(ii) Koordination (mit Problemen beim Parsen)

(iii) Relationsnomen und Nominalisierung von Verben und Adjektiven

(iv) Phrasentransformationen

- (a) Korrigieren und Anpassen von NP-Beugung
- (b) Nominalisierung einfacher Aussagen
- (c) Modus- und Tempusflexion bei einfachen Sätzen
- (d) Aktiv-Passiv-Transformation einfacher Sätze

0.4 Seminarvorträge zur Implementierung: Parser, Hilfsprogramme

(i) Lexemfilter: Beseitigung unbrauchbarer lexikalischer Lesarten (vor dem Parsen)

(ii) Parsen mit Metaregeln: Vorberechnungen von Hilfstabellen für nicht-expandierte Regeln

(iii) Parsen mit Metaregeln: Voraussage brauchbarer *Meta*regeln?

(iv) Erkennung von Teilsatzgrenzen und rekursives Parsen von Teilsätzen

(v) Parsen von diskontinuierlichen Konstituenten (und Extraktion von Teilen)