Automatische Wortformerkennung für das Deutsche im Rahmen von MALAGA

Magisterarbeit

in der Philosophischen Fakultät II

(Sprach- und Literaturwissenschaften)

der Friedrich-Alexander-Universität

Erlangen-Nürnberg

vorgelegt von
Oliver Lorenz
geb. am 18.06.1970 in Bayreuth

Inhaltsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einführung

Die vorliegende Arbeit beschäftigt sich mit der Morphologie der deutschen Sprache und deren Beschreibung durch ein Grammatiksystem zur automatischen Wortformerkennung. Dieses Grammatiksystem benutzt den bei [Hausser 1989] vorgestellten Ansatz zur morphologischen Analyse, LA-MORPH. LA-MORPH ist ein Konzept zur Anwendung der linksassoziativen Grammatik auf die Morphologie. Für die Implementation von linksassoziativen Grammatiksystemen existiert das Programmpaket Malaga, mit dem auch das Grammatiksystem, das der vorliegenden Arbeit zugrundeliegt, entwickelt wurde.

1.1 Motivation

Die Wortformerkennung stellt die Grundlage dar für die meisten linguistischen Aufgabenstellungen, die sich mit Wortformen oder aus Wortformen zusammengesetzten Strukturen beschäftigen. Unabhängig davon, ob es sich bei der Aufgabenstellung im Kern um eine lexikalische, lexikographische, syntaktische, semantische oder pragmatische Untersuchung handelt, ist es immer notwendig, grundlegende Informationen über die Wortformen im Datenmaterial zu bekommen. Dabei ist bei der Betrachtung der Wortformen deren innere und deren syntagmatisches Verhalten von Interesse:

- Aus welchen Teilen und auf welche Art und Weise ist die Wortform zusammengesetzt? Für welche Bedeutungen (Grundformen) stehen die Teile? Nach welchen Regeln sind die Teile zusammengesetzt?

- Welche syntagmatischen Eigenschaften hat die Wortform?

Während die erste Fragestellung in erster Linie unter die linguistischen Teildisziplinen der Lexikologie und der Morphologie fallen, befindet sich die zweite Fragestellung an der Nahtstelle zwischen Morphologie und Syntax.
Bei der automatischen Wortformerkennung werden durch Computerprogramme realisierte linguistische Methoden benutzt, um die innere Form und/oder das syntagmatische Verhalten einer Wortform zu beschreiben.
Je nach Fragestellung und untersuchter Sprache bieten sich dazu mehrere Möglichkeiten an. Handelt es sich bei der untersuchten Sprache beispielsweise um eine Sprache mit analytischem Bau (wie z.B. das Englische oder das Chinesische), so ist die Frage nach der inneren Form einer Wortform meist von geringem

Interesse, da diese Sprachen aufgrund weitgehender Flexionslosigkeit i.d.R. nur über sehr wenige, sehr einfach strukturierte Wortformen zu einer Grundform verfügen. Da bei diesen Sprachen das syntagmatische Verhalten auch hauptsächlich nicht durch morphologische Mittel ausgedrückt wird, sondern vom syntagmatischen Kontext der Wortform abhängt (Wortstellung), kann auch nur in geringem Maße von der inneren Form auf das syntagmatische Verhalten der Wortform geschlossen werden. Liegt das Hauptinteresse der Untersuchung hinsichtlich der Wortformen auf deren syntagmatischem Verhalten, so bietet es sich – bei analytischem Sprachbau – an, nur in geringem Maße oder gar nicht morphologisch zu analysieren und die Zuordnung der syntaktischen Kategorien über voll kategorisierte Wortformlisten geschehen zu lassen.
Im Gegensatz dazu weisen Sprachen mit synthetischem Sprachbau meist eine sehr komplexe innere Form der Wortformen auf. Diese Sprachen verwenden zur Kodierung des syntagmatischen Verhaltens entweder die Flexion oder die Agglutination, die beide eine große Zahl verschiedener Wortformen zu einer Grundform mit sich bringen. Dies ist gerade bei der flektierenden Sprache Deutsch der Fall. Das Deutsche verfügt weiterhin über sehr komplexe morphologische Mechanismen zur Ableitung neuer Wortformen aus vorhandenen und zur Zusammensetzung von Wortformen. Die beiden Aspekte der inneren Form und des syntagmatischen Verhaltens lassen sich bei diesen Sprachen kaum sinnvoll trennen, da die innere Form ja teilweise den Zweck hat, das syntagmatische Verhalten zu kodieren. Daher ist in jedem Fall eine morphologische Analyse notwendig. Diese wird durch Grammatikformalismen durchgeführt, die in geeigneter Weise als Computerprogramme realisiert werden.
Die vorliegende Arbeit beschreibt ein Grammatiksystem zur automatischen Wortformerkennung für die deutsche Sprache. Dieses verwendet den Grammatikformalismus der linksassoziativen Grammatik (LAG), deren Konzept zur morphologischen Anwendung, LA-MORPH, in der Implementationssprache Malaga realisiert ist.

1.2 Aufgabenstellung

Aufgabe der Arbeit ist, ein System zur automatischen Wortformerkennung des Deutschen mithilfe des Programmpaketes Malaga zu erstellen. Das System soll in der Lage sein, beliebigen korrekt geformten deutschen Wortformen ihre syntaktischen Kategorien zuzuordnen, Aussagen über die morphologische Struktur der untersuchten Wortformen zu liefern und allen Wortformen die jeweiligen Grundformen, aus denen sie zusammengesetzt ist, zuordnen zu können. Dazu sollen bereits vorhandene Grundformlexika verwendet werden; wo nötig, sind neue Lexika zu erstellen. Durch die Entwicklung des Grammatiksystems soll auch geprüft werden, inwieweit sich Malaga zur Erstellung von Morphologiegrammatiken eignet, da bisher noch keine großen Grammatiksysteme mit Hilfe von Malaga erstellt wurden.

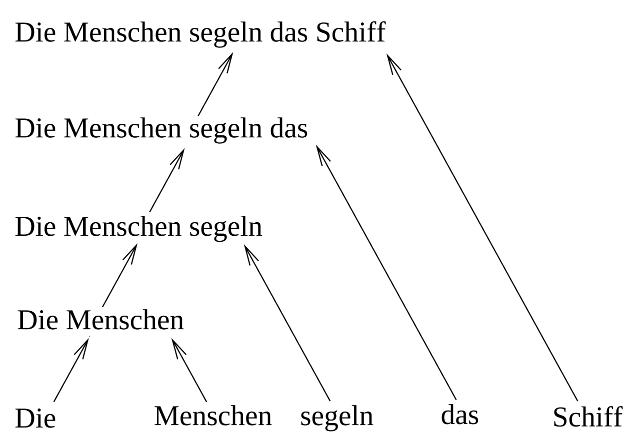
Kapitel 2

Linksassoziative Morphologie

Dieses Kapitel beschreibt beschreibt den Grammatikformalismus der linksassoziativen Grammatik (LAG) und ein Konzept zur Anwendung der LAG auf die morphologische Analyse, LA-MORPH.

2.1 Linksassoziative Grammatiken

Der Grammatikformalismus der linksassoziativen Grammatiken wurde erstmals bei [Hausser 1985] vorgeschlagen. Im Gegensatz zu den Formalismen der Kategorialgrammatik und der Phrasenstrukturgrammatik beruht die Linksassoziative Grammatik nicht auf dem Prinzip der möglichen Ersetzungen, sondern auf dem Prinzip der möglichen Fortsetzungen. Die Ableitungsreihenfolge ist dabei zeitlinear, die Ableitung geschieht oberflächenkompositional.
Bei linksassoziativen Grammatiken werden die einzelnen Segmente, aus denen eine Äußerung besteht, von links nach rechts aneinander konkateniert[1], wie in folgendem Beispiel zu erkennen ist (aus: [Schreiber 1995], S.13):



Dabei geschieht jeder Konkatenationsschritt durch Anwendung einer linksassoziativen Regel; eine linksassoziative Regel kombiniert den bereits analysierten Anfang der Äußerung (der zu Beginn der Analyse leer ist) mit dem nächsten

[1] Dies gilt natürlich nur, wenn man, wie bei der lateinischen Schrift üblich, die Zeit von links nach rechts anträgt.

Segment der Äußerung. Die Konkatenation der beiden wird dann bei der nächsten Regelanwendung wieder als analysierter Wortanfang verwendet, der mit dem nächsten Segment kombiniert wird. Daher muß zu jeder Regel auch ein Paket von möglichen Nachfolgeregeln (also der o.g. möglichen Fortsetzungen) spezifiziert werden.
Eine linksassoziative Regel hat die folgende Form:

$$r_i\text{: } [\ \textit{Anf-CAT}\ \textit{NeuSeg-CAT}\ ] \Rightarrow [\ \{\ rp_i\ \}\ \text{NeuAnf-CAT}]$$

wobei *Anf-CAT* die Kategorie des Anfangs der Äußerung, *NeuSeg-CAT* die Kategorie des nächsten Segments und *NeuAnf-CAT* die Kategorie ist, die den im Regelpaket rp_i angegebenen Nachfolgeregeln als neuer analysierter Anfang übergeben werden soll. Die Entscheidung, ob eine Regelanwendung erfolgreich ist, wird aufgrund der Kategorien *Anf-CAT* und *NeuSeg-CAT* getroffen.
Linksassoziative Grammatiken basieren also auf dem Prinzip der möglichen Fortsetzungen, weil durch die Nachfolgeregeln die Gestalt der möglichen Fortsetzungen beschrieben wird. Die Ableitungsreihenfolge ist zeitlinear, weil die Analyse der Segmente der Äußerung in der Reihenfolge vonstatten geht, in der sie geäußert wurden.
Zur algebraischen Definition der linksassoziativen Grammatik vgl. [Hausser 1988]; zu Fragen der Komplexität bei linksassoziativen Grammatiken vgl. [hausser 1992].

2.2 LA-MORPH

LA-MORPH ist ein Ansatz zur Anwendung des linksassoziativen Grammatikformalismus auf die Morphologie, der bei [Hausser 1989] vorgestellt wird.
Da die morphologische Ableitung einer Äußerung in der LAG oberflächenkompositional verläuft, müssen als Teilsegmente der Äußerung zwangsläufig Allomorphe, nicht Morpheme, angenommen werden (zu den Begriffen Allomorph und Morphem siehe 4.1.1). Dementsprechend müssen alle Allomorphe zur Laufzeit der morphologischen Kombination bereits in kategorisierter Form vorliegen.
Zu diesem Zweck werden in LA-MORPH alle Allomorphe vor der Laufzeit regelhaft aus einem Grundformlexikon generiert. Dies geschieht mit der Hilfe von Allomorph-Regeln. Ein Allomorphgenerator liest dabei einen Grundformeintrag aus dem Lexikon und generiert mit den Allomorph-Regeln alle kategorisierten Allomorphe des Morphems (also der Grundform). Die erzeugten Allomorphe werden im Allomorphlexikon abgespeichert, das dann zur Laufzeit des Analysemoduls aus den generierten Allomorphen mittels linksassoziativer Regeln Wortformen ableitet.
Eine weitere Problematik ist die Identifikation der Segmentgrenzen bei der linksassoziativen morphologischen Analyse. Während die Segmentgrenzen bei der syntaktischen Analyse zumindest bei geschriebener Sprache unproblematisch zu identifizieren sind, weil die einzelnen Segmente i.d.R. durch Zwischenräume voneinander getrennt sind, existieren bei der morphologischen Analyse keinerlei der Äußerung inhärenten Segmentierungsmarkierungen. Dementsprechend sind auch die Schlüssel zu den Allomorphen, nämlich die Allomorphoberflächen, vor der Analyse nicht bekannt.

Dieser Problematik trägt LA-MORPH dadurch Rechnung, daß bei der Analyse einer Wortform jedes Allomorph, das zeichenweise mit dem Beginn der noch nicht abgeleiteten Eingabe übereinstimmt, als neues Segment verwendet wird. Der noch nicht abgeleitete Rest der Eingabe wird also zeichenweise mit den Oberflächen im Allomorphlexikon verglichen; bei jedem Allomorph, das mit dem potentiellen Schlüssel aus der Eingabe hinsichtlich der Oberfläche übereinstimmt, wird versucht, dieses mit dem bereits analysierten Wortanfang zu kombinieren.

Es wurden bereits mehrere Implementationen von LA-MORPH realisiert. Die kategoriale Information wurde bei diesen durch LISP-Ausdrücke und in späteren Versionen durch reguläre Ausdrücke beschrieben. Beide Varianten hatten den grundlegenden Nachteil, daß die Beschreibung der Regeln gerade bei komplexeren Grammatiken zunehmend unübersichtlich wurde. Eine Darstellung der kategorialen Information mit Attribut-Werte-Paaren, die sich auch zu komplexeren Strukturen kombinieren lassen, versprach, dieses Problem zu lösen und auch das Niveau der möglichen linguistischen Abstraktion zu erhöhen[2]. Diese Überlegungen führten zur Konzeption und Implementation von Malaga, das im nächsten Kapitel beschrieben wird.

[2]Quelle: Persönliches Gespräch mit Prof.Dr. Hausser und Björn Beutel.

Kapitel 3

MALAGA

In diesem Kapitel wird die Programmiersprache und Programmumgebung Malaga behandelt werden. Malaga wurde an der Abteilung für Computerlinguistik der Universität Erlangen-Nürnberg von Björn Beutel und Gerald Schüller konzipiert und implementiert; das in Malaga integrierte Modul zur morphologischen Disambiguierung stammt von Markus Schulze.
Der Name Malaga ist ein selbstreferenzierendes Akronym und steht für „**M**alaga **a**rbeitet **l**inks-**a**ssoziative **G**rammatiken **a**b"Ėr bezeichnet zum einen eine Programmiersprache zur Implementation von linksassoziativen Grammatiken; zum anderen steht er für eine Umgebung von Programmen zur Erzeugung und Manipulation der Lexikon und Regeldaten. Beide Aspekte werden ausführlich in [Beutel 1995] beschreiben; in diesem Kapitel sollen nur die wichtigsten Aspekte von Malaga kurz umrissen werden.

3.1 Die Programmiersprache Malaga

Malaga ist eine Programmiersprache, die eigens zum Zweck der Implementation von linksassoziativen Grammatiken entwickelt wurde. Malaga weist die folgenden Charakteristiken auf (vgl. [Beutel 1995], S.31):

- Malaga verfügt über strukturierte Werte. Einfache Werte in Malaga sind Segmente (also Kategorienamen), Zeichenketten und Fließkommazahlen. Diese einfachen Werte können zu komplexen Werten strukturiert werden: zu Verbunden von Attribut-Werte-Paaren, zu Listen und zu Multisegmenten.

- Malaga verfügt über strukturierte Anweisungen. Es existiert ein wie bei vielen Programmiersprachen vorhandenes Blockkonzept sowie Auswahl- und Wiederholungsstrukturen. Variablen sind nur in dem Block definiert, in dem sie deklariert wurden.

- Malaga ist typenfrei, d.h. jeder Variablen dürfen beliebige Werte zugewiesen werden.

- Malaga-Grammatiken terminieren immer.

- Malaga verzichtet aus Komplexitätsgründen auf Unifikation von Merkmalsstrukturen. Vielmehr stehen in Malaga eine Reihe von Operatoren zur Verfügung, die es erlauben, Malaga-Werte explizit zu konstruieren.

3.1.1 Bezeichner, Werte und Ausdrücke

Alle Variablennamen, Segmentnamen, Attributnamen und Regelnamen werden in Malaga Bezeichner genannt. Malaga-Bezeichner dürfen aus Groß- und Kleinbuchstaben, dem Unterstrich, dem Pluszeichen, dem Minuszeichen und vom zweiten Zeichen an auch aus Ziffern bestehen. Groß- und Kleinschreibung wird in Malaga nicht unterschieden. Schlüsselwörter von Malaga dürfen nicht als Bezeichner verwendet werden. Ob ein Bezeichner ein Segment, ein Attribut oder eine Regel bezeichnet, kann Malaga aus dem Kontext selbst erschließen; nur Variablen müssen durch ein vorangestelltes Dollarzeichen $ gekennzeichnet werden.
Werte können in Malaga die einfachen Werte Segment, Zeichenkette und positive Fließkommazahl sein. Diese einfachen Werte können zu komplexen Werten strukturiert werden; diese strukturierten Werte sind in Malaga der Verbund, die Liste und das Multi-Segment.
Verbunde sind Mengen von Attribut-Werte-Paaren. Attribute sind dabei Zugriffsschlüssel, mit deren Hilfe auf die den Attributen zugeordneten Werte zugegriffen werden kann. Verbunde werden in Malaga durch eckige Klammern ([und]) notiert; die darin befindlichen Attribut-Werte-Paare werden durch Kommata voneinander getrennt. Ein Attribut-Werte-Paar besteht aus einem durch einen nachfolgenden Doppelpunkt gekennzeichneten Attributnamen (einem Bezeichner) und einem beliebigen Malaga-Wert:

$$[\ a_1 : w_1,\ a_2 : w_2,\ \ldots]$$

wobei a_i für einen Attributnamen und w_i für den zugeordneten Wert steht. Jeder Attributnamen darf in demselben Verbundname nicht mehr als einmal vorkommen.
Malaga-Listen sind Folgen von beliebigen Malaga-Werten. Die Folge der Werte ist geordnet. Listen werden durch spitze Klammern ($<$ und $>$) notiert; die in der Liste aufgezählten Werte werden durch Kommata getrennt:

$$< w_1, w_2, \ldots >$$

Multi-Segmente sind Segmente, die mit einer klar definierte Anzahl anderer Segmente mittels des Kongruenzoperators ˜(s.u.) kongruieren. Multi-Segmente müssen in der Segmenttabelle (s.u.) folgendermaßen deklariert werden:

$$*\ \textit{Multi-Segment-Name} := <\ \textit{Segmentname}_1,\ \textit{Segmentname}_2,\ \ldots>$$

Ausdrücke sind alle Konstrukte, die einen gültigen Malaga-Wert darstellen. Gültige Malaga-Werte sind einfache und strukturierte Werte sowie Variablen. Malaga-Werte können mit folgenden Operatoren verküpft werden:

- Mit dem Operator „.“ kann auf ein bestimmtes Attribut eines Verbundes oder ein bestimmtes Element einer Liste zugegriffen werden. Ein Ausdruck der Form „$v.a$“ liefert den Wert des Attributs a im Verbund v. Auf Listen angewendet liefert ein Ausdruck der Form „$l.n$**L** das n-te Element von links aus der Liste l; ein Ausdruck der Form „$l.n$**R** liefert das n-te Element von rechts.

- Der Operator „*“ atomisiert ein Segment zu einer Liste; ein Ausdruck der Form „$*\ a$“ liefert die Liste aller atomaren Segmente, die mit a mittels des Kongruenzoperators übereinstimmen. Ist a ein atomares Segment, so enthält die Liste nur das Segment a; ist a ein Multi-Segment, so liefert der Ausdruck die Liste, die alle Segmente enthält, mit denen a deklariert wurde.

- Der „*“-Operator hat noch eine zweite Verwendung: er multipliziert zwei Fließkommazahlen. Ein Ausdruck der Form $a * b$ liefert das Produkt der Fließkommazahlen a und b.

- Der Operator „/“ macht es möglich, Attribute aus einem Verbund bzw. Elemente aus einer Liste zu löschen. Auf Verbunde angewendet liefert ein Ausdruck der Form v/a einen Verbund, der alle Attribute enthält, die im Verbund v enthalten sind, mit Ausnahme des Attributs a. Auf Listen angewendet liefert ein Ausdruck der Form l/n**L** eine Liste, die alle Elemente die in der Liste l enthalten sind, mit Ausnahme des n-ten Elements von links (bzw. von rechts, wenn der Ausdruck l/n**R**) lautet.

- Der Operator „&“ verkettet zwei Werte gleichen Typs, wenn dies Zeichenketten, Listen, Verbunde oder Fließkommazahlen sind.

  - Sind a und b Zeichenketten, so konkateniert der Ausdruck a & b die beiden Zeichenketten zu einer.
  - Sind a und b Listen, so liefert der Ausdruck a & b eine Liste, die alle Elemente der Liste a und alle Elemente der Liste b (in dieser Reihenfolge) enthält.
  - Sind a und b Verbunde, so liefert der Ausdruck a & b einen Verbund, der alle Attribute enhält, die in den Verbunden a und b enthalten sind. Enthalten a und b ein Attribut gleichen Namens, so wird in den Ergebnisverbund das Attribut-Werte-Paar aus dem Verbund b eingetragen.
  - Sind a und b Fließkommazahlen, so liefert der Ausdruck a & b die Summe der beiden Zahlen a und b.

3.1.2 Bedingungen

Bedingungen in Malaga können entweder wahr oder falsch sein. In Malaga existieren folgende Vergleichsoperatoren:

- Der Vergleichsoperator „=“ prüft, ob zwei Werte identisch sind. Die Bedingung $a{=}b$ ist wahr, wenn a und b identisch sind.

- Der Vergleichsoperator „/=“ prüft, ob zwei Werte nicht identisch sind. Die Bedingung $a/{=}b$ ist wahr, wenn $a{=}b$ falsch ist.

- Der Vergleichsoperator „**less**“ prüft, ob eine Fließkommazahl kleiner als eine andere Fließkommazahl ist. Die Bedingung a **less** b ist wahr, wenn a kleiner ist als b.

- Der Vergleichsoperator „~“ prüft, ob zwei Werte miteinander kongruieren. Eine Bedingung der Form „a ~ b ist wahr,
  - wenn a und b beides Segmente sind und die Listen $*a$ und $*\ b$ mindestens ein gemeinsames atomares Element haben
  - wenn a und b beides Listen sind und beide Listen mindestens ein gemeinsames Element haben.
- Der Vergleichsoperator „**in**“ hat zwei Verwendungen. Eine Bedingung der Form a **in** b ist wahr,
  - wenn b ein Verbund ist und das Attribut a im Verbund b enthalten ist;
  - wenn b eine Liste ist und a ein in der Liste enthaltener Verbund ist.

Weiterhin ist es möglich, Zeichenketten mit regulären Ausdrücken zu vergleichen. Eine Bedingung der Form **match** $a = b$ ist wahr, wenn a eine Zeichenkette ist und b ein regulärer Ausdruck, der die Zeichenkette korrekt beschreibt (zu regulären Ausdrücken vgl. [Beutel 1995], S.37). Es kann auch mit der Konkatenation mehrerer regulärer Ausdrücke verglichen werden; eine Bedingung der Form

$$\textbf{match}\ a = b_1\&\ b_2\ \&\ \ldots$$

ist wahr, wenn sich die Zeichenkette a in mehrere Teile zerlegen läßt, von denen in der richtigen Reihenfolge jeder mit einem der regulären Ausdrücke beschrieben werden kann. Durch die **match**-Bedingung können auch Variablen Werte zugewiesen werden:

$$\textbf{match}\ a = \$A{:}b_1\&\ b_2\&\ \ldots$$

Durch diese Bedingung wird der Variablen *$A* der Teil der Zeichenkette als Wert zugewiesen, der von dem regulären Ausdruck b_1 beschrieben wird. Die Belegung von Variablen durch Mustervergleich ist beschränkt; so darf eine **match**-Bedingung nicht Teil einer komplexen Bedingung sein.
Komplexe Bedingungen sind Bedingungen, die durch logische Operatoren miteinander verknüpft werden. Dabei stehen folgende Operatoren zur Verfügung:

- Die komplexe Bedingung **not** b ist wahr, wenn die Bedingung b nicht wahr ist.
- Die komplexe Bedingung b_1 **and** b_2 ist wahr, wenn die Bedingungen b_1 und b_2 beide wahr sind.
- Die komplexe Bedingung b_1 **or** b_2 ist wahr, wenn mindestens eine der beiden Bedingungen b_1 und b_2 wahr ist.

3.1.3 Die Segmenttabelle

Alle Segmente, die in einer Grammatik verwendet werden, müssen in der Segmenttabelle deklariert werden. Die Deklaration geht dadurch vonstatten, daß alle Segmentnamen in der Tabelle aufgezählt werden. Jedem Segmentnamen muß dabei ein Semikolon folgen. Multi-Segmente müssen wie oben beschrieben deklariert werden. Multi-Segmente dürfen nur in der Segmenttabelle deklariert werden. Jeder Segmentname, gleich ob er für ein atomares Segment oder ein Multi-Segment steht, darf in der Segmenttabelle nur einmal deklariert werden.

3.1.4 Lexika

Eine Grundlage jeder Malaga-Grammatik ist das Grundform-Lexikon. Ein Lexikon ist eine Aufzählung von Grundformeinträgen, von denen jeder durch ein Semikolon abgeschlossen werden muß. Ein Grundformeintrag kann ein beliebiger gültiger Malaga-Wert sein. In der Praxis wird man sich meist dazu entscheiden, Lexikoneinträge als Verbunde zu organisieren, wenn auch für bestimmte Zwecke eine Strukturierung als Listen oder gar einfache Werte ebenfalls denkbar erscheint. In der dieser Arbeit zugrundeliegenden Grammatik sind alle Lexikoneinträge als Verbunde organisiert.

3.1.5 Regeln

In Malaga existieren zwei Arten von Regeln: Allomorph-Regeln und Kombinationsregeln. Die Allomorphregeln werden dazu herangezogen, aus den Einträgen des Grundformlexikons alle Allomorphe zu generieren und im Allomorphlexikon abzuspeichern. Morphologische Kombinationsregeln werden dazu verwendet, die Allomorphe im Allomorphlexikon regelhaft zu konkatenieren (bzw. Wortformen regelhaft zu Allomorphfolgen zu zerlegen). Syntaktische Kombinationsregeln werden dazu verwendet, um Wortformen, die von Allomorphregeln analysiert wurden, syntagmatisch zu konkatenieren.

Allomorphregeln

Jeder Satz von Allomorph-Regeln beginnt mit dem Schlüsselwort `initial-rules`; hinter ihm werden, durch Kommata voneinander getrennt, die Namen aller Regeln aufgelistet, die auf einen eingelesenen Lexikoneintrag zuerst angewendet werden sollen.
Allomorphregeln beginnen immer mit dem Schlüsselwort `allo-rule` gefolgt vom Namen der Regel und einem Semikolon. Dem folgt eine Folge von Anweisungen (s.u.), die den eingelesenen Lexikoneintrag bearbeiten. Die Regel wird von dem Schlüsselwort `end allo-rule`, gefolgt von einem Semikolon, abgeschlossen.

Kombinationsregeln

Jeder Satz von Kombinationsregeln beginnt mit der Angabe des Startzustandes. Der Startzustand besteht aus der Kategorie des leeren Wortanfangs (bzw. Satzanfangs) und einem initialen Regelpaket.
Die Kategorie des leeren Wortanfangs wird hinter dem Schlüsselwort `initial-cat` spezifiziert und kann ein beliebiger Malaga-Wert sein. Sie wird durch ein Semikolon angeschlossen.

Das initiale Regelpaket gibt an, welche Regeln zuerst versuchen sollen, den leeren Wortanfang und das erste Allomorph zu verknüpfen. Dazu werden hinter dem Schlüsselwort `initial-rules` alle zuerst anzuwendenden Regeln, voneinander durch Kommata getrennt, aufgezählt. Das initiale Regelpaket wird durch ein Semikolon abgeschlossen.
Jede Kombinationsregel (mit Ausnahme der Endregel, s.u.) beginnt mit dem Schlüsselwort `combi-rule` gefolgt vom Namen der Regel und einem Semikolon. Dem folgt eine Folge von Anweisungen (s.u.), die versuchen, den bisherigen analysierten Wortanfang mit dem neu gelesenen Allomorph zu verknüpfen. Die Regel wird von dem Schlüsselwort `end combi-rule`, gefolgt von einem Semikolon, abgeschlossen.
In jedem Satz von Kombinationsregeln muß sich auch eine Endregel befinden. Eine Endregel liest kein neues Allomorph ein, sondern versucht, den bisher analysierten Wortanfang als wohlgeformt zu akzeptieren. Sie beginnt mit dem Schlüsselwort `end-rule`, gefolgt vom Regelnamen und einem Semikolon. Dem folgt eine Folge von Anweisungen, deren letzte die `accept`-Anweisung (s.u.) sein muß. Die Regel wird von dem Schlüsselwort `end end-rule`, gefolgt von einem Semikolon, abgeschlossen.

Anweisungen

Malaga verfügt über eine Reihe von Anweisungen, um Regeln zu strukturieren. Sie werden hier nur soweit erklärt, wie sie im vorliegenden Grammatiksystem verwendet werden; zu etwaigen erweiterten Formen hier erklärter Anweisungen und hier nicht verwendeten Anweisungen vgl. [Beutel 1995], Anhang B.13. Alle Anweisungen müssen durch ein Semikolon angeschlossen werden.
In der dieser Arbeit zugrundeliegenden Grammatik werden folgende Anweisungen verwendet:

- Die `generate`-Anweisung kann nur in Kombinationsregel verwendet werden und gibt an, welche Allomorphe bei der Paradigma-Generierung mit dem `paradigm`-Kommando (vgl. 3.2.5) verwendet werden sollen. Dazu können hinter dem Schlüsselwort `generate` entweder eine Folge von Allomorph-Oberflächen als Zeichenketten-Konstanten durch Kommata getrennt aufgezählt werden, oder durch das Schlüsselwort `base` angegeben werden, daß zur Generierung alle Allomorphe verwendet werden sollen, die die vom `paradigm`-Kommando übergebene Grundform haben.

- Die `start`-Anweisung wird in Allomorph- und Kombinationsregeln in unterschiedlicher Form verwendet. In Allomorph-Regeln hat die `start`-Anweisung immer die Form

  `start` *$Variable*;

  dadurch wird der Lexikoneintrag (bzw. die Ausgabe der Vorgängerregel) in die Variable *$Variable* eingelesen.

  In Kombinationsregeln (mit Ausnahme der Endregel) hat die `start`-Anweisung die Form

  `start` *$Variable*$_1$, `next` *$Variable*$_2$;

dadurch wird der bereits analysierte Wortanfang in die Variable *\$Variable*$_1$ und das neu gelesene Allomorph in die Variable *\$Variable*$_2$ eingelesen.

In Endregeln hat die `start`-Anweisung dieselbe Form wie in Allomorph-Regeln; hier wird jedoch in die Variable *\$Variable* der bisher analysierte Wortanfang (in diesem Fall die gesamte Wortform) eingelesen.

- Tests haben in Malaga die Form

  ? *Bedingung*;

  und prüfen, ob *Bedingung* wahr ist. Ist dies nicht der Fall, so wird die Abarbeitung der Regel abgebrochen.

- Zuweisungen haben in Malaga die Form

  \$Variable := *Wert*;

  wodurch der Variablen *\$Variable* der Wert *Wert* zugewiesen wird. Die Variable darf vor der Zuweisung noch nicht definiert sein. Die Variable existiert nur in dem Anweisungsblock, in dem sie definiert wurde. Soll einer bereits definierten Variable ein neuer Wert zugewiesen werden, so muß der Zuweisung das Schlüsselwort `set` vorangestellt werden.

- Die `choose`-Anweisung hat die Form

  `choose` *\$Element* `in` *\$Liste*;

  und erzeugt für jedes Element der Liste *\$Liste* einen eigenen Pfad, in dem das jeweilige Listenelement in der Variable *\$Element* gespeichert ist.

- Die `foreach`-Anweisung hat die Form

  `foreach` *\$Element* `in` *\$Liste*: *Anweisungen* `end foreach`;

  und wendet sequentiell die Anweisungsfolge *Anweisungen* auf jedes Listenelement der Liste *\$Liste* an. In jedem Durchlauf ist das jeweilige Listelement in der Variablen *\$Element* abgespeichert. Die foreach-Anweisung wird nur erfolgreich beendet, wenn jeder Durchlauf von *Anweisungen* erfolgreich verläuft.

- Die `if`-Anweisung hat die folgende Form:

`if`	*Bedingung*$_1$	`then`	*Anweisungen*$_1$
`elseif`	*Bedingung*$_2$	`then`	*Anweisungen*$_2$
`else`			*Anweisungen*$_3$
`end if`;			

  Die `elseif` darf beliebig oft (auch keinmal) wiederholt werden; die `else`-Zeile darf keinmal oder einmal vorkommen. Wenn *Bedingung*$_1$ wahr ist, wird die Anweisungsfolge *Anweisungen*$_1$ ausgeführt; ist *Bedingung*$_1$ nicht wahr und *Bedingung*$_2$ wahr, so wird die Anweisungsfolge *Anweisungen*$_2$ ausgeführt; ist keine der vor der `else`-Zeile aufgeführten Bedingungen wahr, so wird die Anweisungsfolge *Anweisungen*$_3$ ausgeführt.

- Die `assert`-Anweisung hat die Form

  `assert` *Bedingung*;

  und bricht die Regelabarbeitung mit einer Laufzeit-Fehlermeldung ab, wenn *Bedingung* nicht wahr ist. Diese Anweisung wird verwendet, um logische Fehler im Grammatiksystem zu finden.

- Die `result`-Anweisung hat die Form

  `result` *Wert*, `rules` $Regel_1$, $Regel_2$, ...;

  und übergibt den Malaga-Wert *Wert* an alle hinter dem Schlüsselwort `rules` spezifizierten Nachfolgeregeln. In Endregeln hat die `result`-Anweisung folgendes Format:

  `result` *Wert*, `accept`;

  In diesem Fall akzeptiert die `result`-Anweisung die bisher analysierte Eingabe als wohlgeformt, wenn bereits die gesamte Eingabe gelesen wurde.

  Wenn man mehr als ein Resultat an Nachfolgeregeln übergeben will bzw. als wohlgeformt akzeptieren will, können auch beliebig viele `result`-Anweisungen hintereinander wiederholt werden.

- Die `allo`-Anweisung hat die Form

  `allo` *Oberfläche*, `cat` *Kategorie*, `base`; *Grundform*

  wobei *Oberfläche* die Zeichenkette ist, die als Oberflächen-Schlüssel des erzeugten Allomorphs ins Allomorphlexikon eingetragen werden soll, *Kategorie* der Wert ist, der die zugehörige Kategorie sein soll, und *Grundform* die zugehörige Grundform. Der `base`-Teil der Anweisung kann weggelassen werden; in diesem Fall wird *Oberfläche* als Grundform eingetragen. Es können beliebig viele `allo`-Anweisungen hintereinander wiederholt werden.

- Die `case`-Anweisung hat die Form

`case`		
`of`	$Bedingung_1$:	$Anweisungen_1$
`of`	$Bedingung_2$:	$Anweisungen_2$
...		
`else`		$Anweisungen_n$
`end case`;		

  wobei die `else`-Zeile auch weggelassen werden kann. Trifft $Bedingung_1$ zu so wird die Anweisungsfolge $Anweisungen_1$ ausgeführt; ist dies nicht der Fall und trifft $Bedingung_2$ zu, so wird die Anweisungsfolge $Anweisungen_2$ ausgeführt; trifft keine der Bedingungen vor der `else`-Zeile zu, so wird $Anweisungen_n$ ausgeführt. Der Unterschied zur `if`-Anweisung ist, daß die Anweisungsfolgen in der `case`-Anweisung mit einer Ergebnis-Anweisung (also einer `result`-Anweisung oder einer `allo`-Anweisung) enden müssen, dies jedoch in einer `if`-Anweisung nicht dürfen.

- Die `fail`-Anweisung hat das Format

  `fail`;

  und bricht den gerade analysierten Regelpfad ab.

- Die `error`-Anweisung hat das Format

  `error`;

  und bricht die Ausführung aller Pfade mit Ausgabe einer Fehlermeldung unter Angabe der Quelltextzeile ab.

- Die `include`-Anweisung hat die Form

  `include "`*Dateiname*`"`;

  und setzt den Inhalt der Datei mit dem Namen *Dateiname* an der Stelle des Quelltexts textuell ein, an der sich die `include`-Anweisung befindet. Diese Anweisung kann in der Segmenttabelle, im Lexikon und in Regeln verwendet werden.

3.1.6 Die Dateien einer Malaga-Grammatik

Die verschiedenen Komponenten einer Malaga-Grammatik, also Segmenttabelle, Lexikon, Allomorphregeln und Kombinationsregeln, befinden sich in verschiedenen Dateien. Die Dateien haben dabei denselben Grundnamen und zusätzlich noch ein für die Art der Datei spezifisches Suffix. Die Komponenten der Grammatik der vorliegenden Arbeit haben alle den Grundnamen `dmm`. Die veschiedenen Teildateien haben dabei folgende Suffixe:

- die Segmentdatei enthält die Segmenttabelle und hat das Suffix `.seg`;
- die Lexikondatei enthält das Grundformlexikon und hat das Suffix `.lex`;
- Die Allomorph-Regel-Datei enthält die Allomorph-Regeln und hat das Suffix `.all`;
- Die Kombi-Regel-Datei enthält die Kombinationsregeln. Handelt es sich um morphologische Kombinationsregeln, so hat die Datei das Suffix `.mor`; bei syntaktischen Kombinationsregeln hat sie das Suffix `.syn`.
- Die Projektdatei hat das Suffix `.pro`. Ihre Funktion wird bei der Vorstellung des Malaga-Programms **malmake** erläutert (vgl. 3.2.4).

3.2 Die Programmumgebung Malaga

Die Programmumgebung Malaga umfaßt eine Reihe von Programmen, mit deren Hilfe Lexikon- und Regeldateien erzeugt, verwaltet und verwendet werden können.

3.2.1 Das Programm malseg

Das Programm **malseg** übersetzt die in der Segmentdatei befindliche Segmenttabelle in ein internes Format. Das Programm wird aufgerufen durch

`malseg` *Segmentdatei*

wobei *Segmentdatei* der Name der Segmentdatei (mit Suffix) ist. Die übersetzte Datei hat dann das Suffix `.seg_c`.

3.2.2 Das Programm malcom

Das Programm **malcom** übersetzt Malaga-Regeldateien. Das Programm wird aufgerufen durch

`malcom` *Segmentdatei Regeldatei*

wobei *Regeldatei* die Datei mit den zu kompilierenden Allomorph- oder Kombinationsregeln ist (mit dem jeweiligen Suffix `.all`, `.mor` oder `.syn`), und *Segmentdatei* der Name der zugehörigen Segmentdatei (mit Suffix `.seg`, nicht `.seg_c`). Die Segmentdatei muß in kompilierter Form vorliegen, damit **malcom** aufgerufen werden kann.

3.2.3 Das Programm mallex

Das Programm **mallex** generiert mit Hilfe der Allomorph-Regel-Datei und der Segmentdatei aus allen Einträgen im Grundformlexikon die Einträge des Allomorphlexikons. Das Programm wird aufgerufen durch

`mallex` *Segmentdatei Allo-Regel-Datei Lexikondatei*

wobei *Segmentdatei* der Name der Segmentdatei (mit Suffix `.seg`), *Allo-Regel-Datei* der Name der Allomorph-Regel-Datei (mit Suffix `.all`) und *Lexikondatei* der Name der Lexikondatei ist, die das Grundformlexikon enthält. Die Segmentdatei und die Allomorph-Regel-Datei müssen in kompilierter Form vorliegen, damit **mallex** aufgerufen werden kann.
mallex kann auch interaktiv gestartet werden; zu der interaktiven Benutzerschnittstelle von **mallex** vgl. [Beutel 1995], S.14ff. Um **mallex** interaktiv aufzurufen, muß es mit der Option `-debug` aufgerufen werden.
Alternativ zum oben genannten Aufruf kann **mallex** unter Angabe der Projektdatei (s.u.) aufgerufen werden:

`mallex` *Projektdatei*

wobei *Projektdatei* der Name der Projektdatei (mit Suffix `.pro`) ist.

3.2.4 Das Programm malmake

Die eben aufgezählten Programme können alle einzeln für sich aufgerufen werden. Um dem Grammatikentwickler und -benutzer die Benutzung dieser Programme zu erleichtern, existiert das Programm **malmake**. Es überprüft, ob alle für die Ausführung eines Programmes notwendigen Dateien vorhanden sind

und in kompilierter Form vorliegen. Liegt eine Datei nicht in kompilierter Form vor, so ruft **malmake** das entsprechende Programm auf, das die Datei dann kompiliert.
Damit **malmake** die Abhängigkeiten zwischen den Dateien korrekt erfassen kann, müssen diese in der Projektdatei mit dem Suffix `.pro` deklariert werden. Dazu existieren verschiedene Schlüsselwörter, die die Funktion der beteiligten Dateien angeben:

- hinter dem Schlüsselwort `seg:` müssen die Pfade zur Segmentdatei und allen Dateien, die mit der `include`-Anweisung in sie eingeschlossen werden, durch Leerzeichen voneinander getrennt aufgezählt werden;
- hinter dem Schlüsselwort `all:` müssen die Pfade zur Allomorph-Regel-Datei und allen Dateien, die mit der `include`-Anweisung in sie eingeschlossen werden, aufgezählt werden;
- hinter dem Schlüsselwort `lex:` müssen die Pfade zur Lexikondatei und allen Dateien, die mit der `include`-Anweisung in sie eingeschlossen werden, aufgezählt werden;
- hinter dem Schlüsselwort `mor:` müssen die Pfade zur morphologischen Kombinationsregeldatei und allen Dateien, die mit der `include`-Anweisung in sie eingeschlossen werden, aufgezählt werden;
- hinter dem Schlüsselwort `syn:` müssen die Pfade zur syntaktischen Kombinationsregeldatei und allen Dateien, die mit der `include`-Anweisung in sie eingeschlossen werden, aufgezählt werden; die Spezifikation der Syntaxregeln ist optional;

Wenn sich die Deklaration einer Komponente über mehr als eine Zeile erstrecken soll, muß das entsprechende Schlüsselwort vor jeder der Zeilen stehen.
Da die Projektdatei auch von den Programmen **mallex** und **malaga** (s.u.) aufgerufen werden kann, können den beiden Programmen Befehle ihrer interaktiven Benutzerschnittstellen übergeben werden, die beim Aufruf der Programme ausgeführt werden sollen. Dazu existieren analog zu obigen Schlüsselwörtern die Schlüsselwörter `mallex:` und `malaga:`.

3.2.5 Das Programm malaga

Das Programm **malaga** ist die eigentliche Benutzerschnittstelle des Programmpakets. Mit seiner Hilfe können Wortformen oder Sätze analysiert werden und Paradigmen, Wortformen und Sätze generiert werden. Es wird aufgerufen durch

> `malaga` *Segmentdatei Morpho-Kombiregeldatei Lexikondatei Syntax-Kombiregeldatei*

oder durch

> `malaga` *Projektdatei*

Die Angabe einer Syntax-Kombiregeldatei ist sowohl bei Aufruf unter Angabe aller Teildateien als auch unter Angabe der Projektdatei optional.
Die interaktiven Kommandos von **malaga** werden bei [Beutel 1995] ausführlich dokumentiert. Hier sollen nur die wichtigsten Kommandos kurz besprochen werden:

- Das Kommando `ma` analysiert Wortformen. Dazu wird die zu analysierende Wortform hinter dem Kommando angegeben.

- Das Kommando `ga-file` liest eine Datei mit Wortformen ein und analysiert jede der Wortformen. Jede Wortform muß dabei in der Datei in einer eigenen Zeile stehen. Der Pfad zu der Datei wird hinter dem Kommando angegeben. Die Analyseergebnisse werden in eine Datei ausgegeben, die den Namen der Eingabe mit dem zusätzlichen Suffix `.cat` hat.

- Das Kommando `mg` generiert Wortformen. Es hat die Syntax

  `mg` *Zahl* $Allomorph_1$ $Allomorph_2$ ... $Allomorph_n$

  wobei mittels der Regeln alle wohlgeformten Wortformen generiert werden, die aus den angegebenen Allomorphen und aus nicht mehr als *Zahl* Allomorphen bestehen.

- Das Kommando `paradigm` gibt alle zu einer Grundform zugehörigen Wortformen aus. Dazu wird die Grundform hinter dem Kommando angegeben.

- Das Kommando `path` gibt eine textuelle Darstellung der Analysepfade aus.

- Kann eine Wortform nicht erfolgreich analysiert werden, so können mit dem Kommando `longest` die längsten Analysepfade textuell ausgegeben werden.

- Das Kommando `tree` gibt eine graphische Darstellung der Analysepfade aus, wenn **malaga** unter der graphischen Benutzerobefläche X-Windows gestartet wird und die Programmpakete Tcl7.4 und Tk4.0 installiert ist (vgl. dazu [Beutel 1995], S.28ff).

- Das Kommando `help` gibt eine Übersicht über alle Befehle aus.

- Das Kommando `quit` beendet das Programm **malaga**.

Kapitel 4

Das behandelte Fragment des Deutschen

Dieses Kapitel beschreibt das im erstellten Grammatiksystem behandelte morphologische Fragment des Deutschen.
Dazu werden zuerst die linguistischen Begriffe eingeführt und ggf. ihre Signifikanz für das Grammatiksystem erläutert. Anschließend werden voneinander getrennt die verschiedenen Wortarten hinsichtlich dieser Begriffe beschrieben.
Grundlage für die Systematik der Flexion bildet (außer, wo anders angegeben) im wesentlichen Engels *Deutsche Grammatik* [Engel 1988]. Grundlage für die Beschreibung der Wortbildung bilden die Werke von Erben [Erben 1983], Fleischer [Fleischer 1982] und Fleischer/Barz [Fleischer und Barz 1982].

4.1 Einführung

Der Teilbereich der Linguistik, der die Grundlage für die regelgesteuerte automatische Wortformerkennung bildet, ist die Morphologie. Die Morphologie befaßt sich mit den Prozessen und Regelhaftigkeiten, die bei der Bildung von Wortformen eine Rolle spielen.
Wortformen können aus mehreren Teilen zusammengesetzt sein. Zentrale Aspekte dieser Tatsache sind die Teile, aus denen die Wortformen zusammengesetzt sind, und die Art und Weise, wie die Teile zusammengesetzt werden.
Hinsichtlich der Beschreibung der Teile sind die Begriffe Morph, Morphem und Allomorph von Bedeutung. Alle drei Begriffe beschreiben verschiedene Aspekte dieser Wortteile.
Was die Art der Zusammensetzung angeht, so wird traditionell unterschieden zwischen der Flexion (Zusammensetzung zum Zweck der Zuordnung grammatikalischer Kategorien) und der Wortbildung (semantische Zusammensetzung oder Modifikation). Das weite Feld der Wortbildung wird noch unterteilt in Komposition (Zusammensetzung von Lexemen zu komplexen Wörtern) und Derivation (Ableitung von neuen Wörtern aus Lexemen).

4.1.1 Morph, Morphem und Allomorph

Der Begriff des Morphs (vgl. [Bußmann 1990], ↑ Morph) bezeichnet die konkret realisierte lautliche Äußerung eines Wortteils. Alle Morphe, die konkrete Repräsentanten derselben Bedeutung sind, sind konkrete Äußerungen eines Morphems (vgl. [Bußmann 1990], ↑ Morphem). Der abstrakte Begriff des Morphems bezeichnet also den Träger einer Bedeutung; das Morphem ist die kleinste bedeutungstragende Einheit einer Sprache.
Die konkrete lautliche Gestalt von Morphen eines Morphems kann variieren. Diese Varianz kann entweder unsystematisch sein, also ohne grammatikalische Signifikanz, oder systematisch, d.h. hinter der Varianz verbirgt sich irgendeine erfaßbare Systematik, die von grammatikalischer Bedeutung ist.
Bei systematischer Varianz müssen sich Kriterien dafür finden lassen, in welchen Umgebungen welche Varianten auftreten. Diese Umgebungen (bzw. deren Abgrenzung) bezeichnet man als Verteilung oder Distribution. Wenn diese Verteilung der Varianten komplementär ist, wenn also jede Variante nur in der Umgebung vorkommt, in der die anderen Varianten nicht vorkommen, dann bezeichnet man die Varianten als Allomorphe eines Morphems (vgl. [Bußmann 1990], ↑ Allomorph).
Dem Begriff der Allomorphie kommt in der linksassoziativen Morphologie eine zentrale Rolle zu, insofern als hier die oberflächenkompositionale Zusammensetzung von Wortformen auf Allomorphbasis und nicht auf Morphembasis beschrieben wird.
Man unterscheidet phonologisch motivierte und morphologisch motivierte Allomorphie. Bei phonologisch motivierter Allomorphie wird die lautliche Gestalt des Allomorphs von der phonetischen Umgebung bestimmt; die Auslautverhärtung im Deutschen ist ein Beispiel phonologischer Allomorphie. Allomorphie, die nicht phonologisch motiviert ist, heißt morphologische Allomorphie.
Die vorliegende Arbeit behandelt die Oberflächen von Morphen jedoch nicht auf phonetischer, sondern auf orthographischer Basis. Dadurch verschwinden an einigen Stellen phonologische Allomorphiephänomene, die in der Orthographie nicht widergespiegelt werden, wie z.B. die Auslautverhärtung. Dafür tauchen an anderen Stellen Variationsphänomene auf, die von rein orthographischen Phänomenen herrühren, beispielsweise bei der Substitution von <*ß*> durch <*ss*>. Zwar ist hier strenggenommen der Begriff der Allomorphie nicht korrekt; im Rahmen der Aufgabenstellung und in Anbetracht der Tatsache, daß die Orthographie des Deutschen als eine (wenn auch an einigen Stellen phonologisch unzureichende) Abbildung der lautlichen Gestalt des Deutschen betrachtet wird, erscheint die Benutzung des Allomorphiebegriffs dennoch angemessen.
Bei der Beschreibung von morphologischen Allomorphiephänomenen lassen sich diese in vier Klassen der Regularität einteilen: reguläre, semi-reguläre, semi-irreguläre und irreguläre Allomorphie (vgl. [Hausser 1996], S.345).

- reguläre Allomorphie: Es existiert nur ein Allomorph (Morphem); es existiert also gar keine Allomorphie.
- semi-reguläre Allomorphie: Es existiert mehr als ein Allomorph; die Gestalt aller Allomorphe kann aus der Oberfläche der Grundform geschlossen werden.
- semi-irreguläre Allomorphie: Es existiert mehr als ein Allomorph; die Gestalt der Allomorphe kann nicht aus der Oberfläche der Grundform

geschlossen werden, läßt sich jedoch regelhaft ableiten, wenn zusätzliche kategoriale Information über die Irregularität vorhanden ist.

- irreguläre Allomorphie: Es existiert mehr als ein Allomorph; die Gestalt der Allomorphe läßt sich weder aus der Oberfläche der Grundform noch aus sonstiger kategorialer Information regelhaft ableiten.

4.1.2 Flexion

Die Flexion drückt wortklassenspezifisch bestimmte grammatikalische Funktionen von Wortformen aus (vgl. [Bußmann 1990], ↑ Flexion). Dabei können nur Lexeme (also Substantive, Verben und Adjektive) flektiert werden. Man bezeichnet die Flexion von Verben auch als Konjugation und die Flexion von Substantiven und Adjektiven auch als Deklination. Bei Adjektiven tritt nicht nur die Deklination, sondern auch die Komparation auf.
Die Flexion kann in mehreren Formen vonstatten gehen, die auch kombiniert vorkommen können. Im Deutschen kann flektiert werden durch

- Abwandlung des Stammes (Umlautung, Ablautung, Konsonantenwechsel) und/oder durch

- Anfügung von Flexionsendungen an den Stamm.

Eine Eigenschaft der Flexion ist, daß die Flexionsmarkierungen meist mehr als nur eine grammatikalische Funktion ausdrücken, also polysem sind (im Gegensatz zur Agglutination, bei der jedes Agglutinationssuffix nur eine grammatikalische Funktion ausdrückt, vgl. [Bußmann 1990], ↑ Agglutination). So drückt beispielsweise eine Substantivflexionsendung die Flexionskategorien Kasus und Numerus zugleich aus.
Eine Flexionsendung kann in mehr als einer Wortklasse auftauchen. So kommt z.B. die Flexionsendung *-en* sowohl bei Substantiven, Adjektiven und bei Verben vor. Desweiteren kann eine Flexionsendung auch bei einer Wortklasse für mehr als eine Kombination von Flexionsparametern stehen. So kann die Flexionsendung *-en* bei Verben sowohl den Infinitiv als auch die 1. und 3. Person Plural darstellen. Außerdem wird eine bestimmte Kombination von Flexionskategorien bei Vertretern einer Wortklasse nicht immer durch die gleiche Flexionsendung repräsentiert; beispielsweise kann der Genitiv Singular eines Substantivs durch die Flexionsendungen *-(e)s* (wie in *des Garten-s, des Ball-(e)s, des Holzes*) oder *-en* (wie in *des Student-en*) markiert sein oder auch ohne Flexionsendung (wie in *der Frau*).

4.1.3 Komposition

Der erste große Teilbereich der Wortbildung ist die Komposition. Als Komposition wird die Zusammensetzung von Lexemen (also Verben, Substantiven und Adjektiven) zu komplexen Wörtern bezeichnet (vgl. [Bußmann 1990], ↑ Komposition).
Die Komposition läßt sich untergliedern in parataktische (gleichordnende) und hypotaktische (unterordnende) Komposition (vgl. [Erben 1983], S.55).
Bei der parataktischen Komposition (Kopulativkomposition) bringen alle Kompositionsglieder ihre für die Bedeutung der Zusammensetzung relevanten

Merkmale in gleich starkem Maße ein. Im Regelfall besitzen beide Komponenten dieselbe Wortklasse. Während Substantiv–Substantiv-Kopulativkomposita (z.B. *Arzt-Kosmonaut, Schauspieler-Autor*) und Adjektiv–Adjektiv-Kopulativkomposita (z.B. *süß-sauer, naßkalt*) zwar selten, jedoch produktiv sind, kommen Verb–Verb-Kopulativkomposita kaum vor (und wenn, dann meist nur fachsprachlich, wie z.B. *fließpressen* in der Fertigungstechnik), und stellen auch kein produktives Wortbildungmuster dar (vgl. [Erben 1983], S.57).
Bei der hypotaktischen Komposition unterscheidet man die Zusammenrückung einer syntaktischen Gruppe (wie in *Saure-Gurken-Zeit*), die kaum produktiv ist, und die Determinativkomposition, bei dem ein (i.d.R. vorangestelltes) Bestimmungswort ein Grundwort genauer spezifiziert. Die Determinativkomposition stellt einen der produktivsten Wortbildungsprozesse des Deutschen dar. Dabei können Substantive, Adjektive und Verben an der Komposition beteiligt sein, wenn auch bestimmte Kombinationsmuster bei weitem häufiger sind als andere (vgl. dazu [Drosdowski 1984], S.407).
Die Komposition kann eine große Anzahl von semantischen Relationen zwischen den beteiligten Komponenten ausdrücken. Bisher ist es noch nicht in hinreichendem Maße gelungen, formal erfaßbare Kriterien für die semantische Wohlgeformtheit einer Komposition zu beschreiben (vgl. [Erben 1983], S.62). Aufgrund dieser Tatsache und in Ermangelung semantischer Information in den vorliegenden Grundformlexika müssen in dieser Arbeit die morphosemantischen Implikationen der Komposition unbehandelt bleiben; sie würden den Rahmen der Arbeit auch bei weitem sprengen. Die Komposition wird hier daher als durch Wortklasse und morphologische Merkmale restringierte Konkatenation von Lexemen behandelt.
Zu einer ausführlichen Beschreibung aller Aspekte der Komposition vgl. [Engel 1988] und [Fleischer 1982].

4.1.4 Derivation

Der andere große Bereich der Wortbildung ist die Derivation. Als Derivation wird die Ableitung neuer Wörter aus einem vorhandenen Lexem bezeichnet. Die Derivation zerfällt in zwei Bereiche: explizite Derivation (Derivation vom einen Lexem durch Affigierung mit einem Nicht-Lexem) sowie implizite Derivation (Derivation ohne Affigierung) (vgl. [Erben 1983], S.66f).

Implizite Derivation

Die implizite Derivation umfaßt die Derivation durch lautliche Änderung (i.d.R. Wechsel des Stammvokals und/oder Konsonantenwechsel, wie z.B. in *klingen – Klang, fliehen – Flucht*) sowie die Nullableitung, bei der ein Wort ohne lautliche Veränderung von einer Wortklasse in eine andere Wortklasse hinübertritt (wie z.B. in *deutsch* (Adj.) – *Deutsch* (Subst.)) (vgl. [Bußmann 1990], ↑ Derivation). Die Derivation durch lautliche Änderung (innere Derivation) wird in dieser Arbeit im Rahmen des Lexikons behandelt werden. Das Prinzip der Oberflächenkompositionalität macht eine Behandlung dieser Derivationsform in der morphologischen Analyse nicht möglich, so daß die Bearbeitung bereits bei der Allomorpherstellung geschehen müßte. Da diese Derivationsprozesse jedoch nicht produktiv sind, erschien diese Lösung nicht angemessen, so daß die Behandlung im Lexikon vorgezogen wurde.

Die Nullableitung (Konversion) wird wegen des oberflächenkompositionalen Ansatzes der LAG und angesichts der weitreichenden Möglichkeiten der Nullableitung in dieser Arbeit nicht behandelt werden. Zwar wäre es möglich, die Wortklasse einer Nullableitung durch Beachtung von Groß- und Kleinschreibung zumindest einzuschränken; da MALAGA jedoch nicht zwischen Groß- und Kleinbuchstaben unterscheidet, entfällt auch diese Möglichkeit. Da eine Beschreibung im Rahmen der Morphologie zu einer großen Zahl von Ambiguitäten führen würde, erscheint eine Disambiguierung im syntaktischen Kontext effizienter.

Explizite Derivation

Die explizite Derivation zerfällt in zwei Teile, je nachdem, ob das Affix vor oder nach dem Grundmorphem angeheftet wird. Affigierung vor dem Grundmorphem bezeichnet man als Präfigierung; wird das Affix hinten an das Grundmorphem angeheftet, so spricht man von Suffigierung.
Bei der Präfigierung wird im Deutschen ein frei oder gebunden vorkommendes Morphem vor ein Lexem geheftet. Dabei werden die nur gebunden vorkommenden Präfixe (wie z.B. *be-* in *be-halten*, *un* in *un-würdig*) als Präfixe im engeren Sinne bezeichnet; die Präfixe, die auch frei vorkommen können (wie z.B. *auf* in *auf-halten, Auf-satz*) bezeichnet man als Präfixoide (vgl. [Erben 1983], S.25). Die Präfigierung ist zwar bei den meisten Präfixen nicht produktiv, aber reihenbildend, also nicht an ein einzelnes Grundmorphem gekoppelt. Die Wortklasse eines Grundmorphems bleibt durch die Präfigierung unverändert.
Die Suffigierung heftet ein frei oder gebunden vorkommendes Morphem hinten an ein Grundmorphem an. Analog zur Präfigierung spricht man von Suffixen im engeren Sinne (wie z.B. *-ung* in *Scheid-ung*) sowie von Suffixoiden (wie z.B. *-zeug* in *Werk-zeug, Flug-zeug*) (vgl. [Erben 1983], S.25). In den meisten Fällen sind die Suffixe wortklassenspezifisch und können, müssen jedoch nicht in einer Wortklassenänderung resultieren. In den meisten Fällen ist die Suffigierung reihenbildend produktiv, wenn auch semantisch restringiert.

4.1.5 Orthographie

Die Orthographie des Deutschen muß an zwei Stellen betrachtet werden:

- Hinter kurzen Stammvokalen stehen in der deutschen Orthographie oft Doppelkonsonanten. Wird durch Allomorphie aus dem kurzen Vokal ein langer, muß unter Umständen der Doppelkonsonant zu einem einfachen verkürzt werden. Wechsel der Vokallänge tritt nur bei Ablautung (vgl. 4.2.1) auf.

- Steht hinter einem kurzen Stammvokal *-ss* am Stammende, so muß ein weiteres Allomorph auf *-ß* erzeugt werden. Dieses Allomorph wird verwendet, wenn der Stamm alleine steht oder wenn eine auf *-t-* beginnende Flexionsendung konkateniert wird. Diese Regel muß bei der Flexion von Verben, Substantiven und Adjektiven beachtet werden.

4.2 Die Flexion

4.2.1 Die Konjugation

Verben sind Worte, zu denen sich entweder finite Formen oder ein Partizip II bilden lassen (vgl. [Engel 1988], S.388). Die Konjugation, also die Flexion von Verben, ist, soweit sie durch morphologische Mittel ausgedrückt wird, abhängig von vier Parametern:

- Person. Es existieren drei Personen.
- Numerus. Es existieren zwei Numeri: Singular und Plural.
- Tempus. Durch rein morphologische Mittel können zwei Tempora ausgedrückt werden: Präsens und Präteritum. Alle anderen Tempora werden durch syntaktische Fügungen gebildet.
- Modus. Es existieren drei Modi: Indikativ, Konjunktiv und Imperativ. Während Indikativ und Konjunktiv in allen Personen, Numeri und Tempora existieren, läßt sich der Imperativ nur bei bestimmten Person-Numerus-Kombinationen und nur im Präsens bilden.

Alle Verbformen, die sich hinsichtlich dieser Parametermatrix einteilen lassen, bezeichnet man als finite Verbformen. Neben den finiten Formen existieren noch drei infinite Verbformen:

- Der Infinitiv (Grundform)
- Das Partizip I (Partizip Präsens)
- Das Partizip II (Partizip Perfekt)

Im allgemeinen unterscheidet das Deutsche hinsichtlich der Flexion schwache und starke Verben, sowie Verben, die Eigenschaften beider Klassen vereinen. Das Kriterium für die Unterscheidung dieser Klassen ist die Art und Weise, wie die Verben ihr Präteritum und ihr Partizip II bilden (vgl. [Engel 1988], S.393 und [Bußmann 1990], ↑ Starke vs. schwache Verben).

Schwache Verben

Die schwachen Verben bilden das Präteritum dadurch, daß sie im Gegensatz zu den Präsensformen zwischen Stamm und Flexionssuffix ein *t* infigieren (vgl. [Engel 1988], S.393). Ihr Partizip II bilden die schwachen Verben durch das Präfix *ge-* (mit Ausnahme der Verben mit nichttrennbaren Präfixen (vgl. 4.3.1), und aller Verben nichtgermanischen Ursprungs, die auf *-ieren* enden) und durch das Suffix *-t* (vgl. Tabelle 4.1).
Bezüglich ihres Allomorphie-Regularitätsgrades lassen sich bei den schwachen Verben zwei Untergruppen unterscheiden. In der Regel besitzen schwache Verben nur ein Allomorph (sind also regulär); eine Ausnahme hiervon bilden die Verben, deren Infinitiv auf *-eln* oder *-ern* endet. Diese bilden zwei Allomorphe: einen Stamm, der auf *-el* bzw. *-er* endet, und einen, der auf *-l* bzw. *-r* endet. Das Allomorphieverhalten läßt sich bei diesen Verben aufgrund der Oberfläche ihrer Grundform schließen; sie sind also semi-regulär.

Verb	1. Pers.Sg. Präsens	1. Pers.Sg. Präteritum	Partizip II
glauben	glaub-e	glaub-t-e	ge-glaub-t
sagen	sag-e	sag-t-e	ge-sag-t
blicken	blick-e	blick-t-e	ge-blick-t

Tabelle 4.1: Erkennungsmerkmale der schwachen Verben

Die schwachen Verben bilden alle ihre flektierten Formen des mit Hilfe von vier Flexionssuffixen: *-e*, *-en*, *-st* und *-t*, wobei die letzten drei Flexionssuffixe in verschiedenen Varianten verwirklicht sein können:

- Bei Verbstämmen, deren Grundform auf *-eln* oder *-ern* endet, wird das Flexionssuffix *-en* zu *-n*.
- Nach dentalen Plosiven und Konsonantenclustern, die aus einem nicht-liquiden Konsonanten (also nicht /l/ oder /r/) und einem nasalen Konsonanten (also /m/ oder /n/) bestehen (nasale Doppelkonsonanz), wird *-st* zu *-est* und *-t* zu *-et*. Dementsprechend werden bei den Indikativformen des Präteritums immer diese um *-e-* erweiterten Formen verwendet, da die Präteritalinfixe *-t-* bzw. *-et-* auf einen dentalen Plosiv enden.
- Nach Sibilanten (also /s/, /z/ und /ʃ/) wird *-st* zu *-t*.
- Im Konjunktiv werden *-st* und *-t* immer zu *-est* und *-et*.

Die Flexionssuffixe stehen dabei im Präsens für folgende Person–Numerus-Kombinationen (vgl. 4.2):

- *-e* steht für 1. Pers. Sg. Ind., für 1. und 3. Pers. Sg. Konj. sowie für die 2. Pers. Sg. Imperativ vertraulich;
- *-st* (bzw. nach Dentalen und nasaler Doppelkonsonanz *-est* und bei auf Sibilanten endenden Stämmen *-t*) steht für die 2. Pers. Sg. Ind.;
- *-en* (bzw. *-n* nach Stämmen mit der Grundformendung *-eln* oder *-ern*) steht für die 1. und 3. Pers. Pl. Indikativ und Konjunktiv, für die 1. Pers. Pl. Imperativ und für die Distanzform des Imperativs;
- *-t* (bzw. nach Dentalen und nasaler Doppelkonsonanz *-et*) steht für die 3. Pers. Sg. Ind., für die 2. Pers. Pl. Ind. und für die 2. Pers. Pl. Imperativ vertraulich;
- Die 2. Pers. Sg. Konj. endet immer auf *-est*;
- Die 2. Pers. Pl. Konj. endet immer auf *-et*.

Die Präteritumstämme werden dadurch gebildet, daß der Präsensstamm mit *-t* suffigiert wird. Somit enden alle Präteritumstämme auf einen Dental und *-st* und *-t* werden immer zu *-est* und *-et*. Die Flexionsendungen im Präteritum lauten also:

- *-t-e* steht immer für die 1. und 3. Pers. Sg. Indikativ und Konjunktiv

		glauben	reden atmen segnen	reisen schwitzen täuschen
Indikativ Präsens	Sg. 1.	-e	-e	-e
	2.	-st	-est	-t
	3.	-t	-et	-t
	Pl. 1.	-en	-en	-en
	2.	-t	-et	-t
	3.	-en	-en	-en
Konjunktiv Präsens	Sg. 1.	-e	-e	-e
	2.	-est	-est	-est
	3.	-e	-e	-e
	Pl. 1.	-en	-en	-en
	2.	-et	-et	-et
	3.	-en	-en	-en
Indikativ Präteritum	Sg. 1.	-t-e	-et-e	-t-e
	2.	-t-est	-et-est	-t-est
	3.	-t-e	-et-e	-t-e
	Pl. 1.	-t-en	-et-en	-t-en
	2.	-t-et	-et-et	-t-et
	3.	-t-en	-et-en	-t-en
Konjunktiv Präteritum	Sg. 1.	-t-e	-et-e	-t-e
	2.	-t-est	-et-est	-t-est
	3.	-t-e	-et-e	-t-e
	Pl. 1.	-t-en	-et-en	-t-en
	2.	-t-et	-et-et	-t-et
	3.	-t-en	-et-en	-t-en
Imperativ Präsens vertr.	Sg. 2.	-(e)	-(e)	-(e)
	Pl. 1.	-en	-en	-en
	2.	-t	-et	-t
Imperativ Präsens dist.	Sg. 2.	-en	-en	-en
Infinitiv		-en	-en	-en
Partizip I		-end	-end	-end
Partizip II	(ge-)	-t	-et	-t

Tabelle 4.2: Flexionsverhalten der regulären schwachen Verben

- *-t-est* steht immer für die 2. Pers. Sg. Indikativ und Konjunktiv
- *-t-en* steht immer für die 1. und 3. Pers. Pl. Indikativ und Konjunktiv
- *-t-et* steht immer für die 2. Pers. Pl. Indikativ und Konjunktiv.

Bei den semi-regulären Verben, also den Verben, deren Infinitiv auf *-eln* oder *-ern* enden, ist noch die Distribution der beiden Allomorphe zu beachten (vgl. Tabelle 4.3). Bei den Verben auf *-eln* wird das *-l*-Allomorph bei der 1. Person Singular Indikativ Präsens und der 1., 2. und 3. Person Singular und der 2. Person Plural des Konjunktiv Präsens sowie bei dei der 2. Person Singular Imperativ Präsens verwendet. Bei den auf *-ern* endenden Verben wird das *-er*-Allomorph bei allen Formen verwendet; zusätzlich kann das *-r*-Allomorph bei der 1. Person Singular Indikativ Präsens, bei der 1. und 3. Person Konjunktiv Präsens und der 2. Person Singular Imperativ Präsens verwendet werden[1].

[1]Strenggenommen handelt es sich bei den Varianten der *-ern*-Verben dementsprechend nicht um Allomorphe, da die Morphe nicht komplementär distribuiert sind. Vielmehr handelt es sich um Morphe, von denen eines nur in bestimmten Formen vorkommen kann.

		lächeln	zittern
Indikativ Präsens	Sg. 1.	lächl-e	zitt(e)r-e
	2.	lächel-st	zitter-st
	3.	lächel-t	zitter-t
	Pl. 1.	lächel-n	zitter-n
	2.	lächel-t	zitter-t
	3.	lächel-n	zitter-n
Konjunktiv Präsens	Sg. 1.	lächl-e	zitt(e)r-e
	2.	lächl-est	zitter-est
	3.	lächl-e	zitt(e)r-e
	Pl. 1.	lächel-n	zitter-n
	2.	lächl-et	zitter-et
	3.	lächel-n	zitter-n
Indikativ Präteritum	Sg. 1.	lächel-t-e	zitter-t-e
	2.	lächel-t-est	zitter-t-est
	3.	lächel-t-e	zitter-t-e
	Pl. 1.	lächel-t-en	zitter-t-en
	2.	lächel-t-et	zitter-t-et
	3.	lächel-t-en	zitter-t-en
Konjunktiv Präteritum	Sg. 1.	lächel-t-e	zitter-t-e
	2.	lächel-t-est	zitter-t-est
	3.	lächel-t-e	zitter-t-e
	Pl. 1.	lächel-t-en	zitter-t-en
	2.	lächel-t-et	zitter-t-et
	3.	lächel-t-en	zitter-t-en
Imperativ Präsens vertr.	Sg. 2.	lächl-e	zitt(e)r-e
	Pl. 1.	lächel-n	zitter-n
	2.	lächel-t	zitter-t
Imperativ Präsens dist.	Sg. 2.	lächel-n	zitter-n
Infinitiv		lächel-n	zitter-n
Partizip I		lächel-nd	zitter-nd
Partizip II		ge-lächel-t	ge-zitter-t

Tabelle 4.3: Flexionsverhalten der semi-regulären schwachen Verben

Verb	1. Pers.Sg. Präsens	1. Pers.Sg. Präteritum	Partizip II
tragen	trag-e	trug	ge-trag-en
schlafen	schlaf-e	schlief	ge-schlaf-en
brechen	brech-e	brach	ge-broch-en

Tabelle 4.4: Erkennungsmerkmale der starken Verben

Starke Verben

Die starken Verben bilden ihr Präteritum durch Ablautung und das Partizip II durch Präfigierung mit *ge-* (mit Ausnahme der Verben mit nichttrennabren Präfixen) und Suffigierung mit *-en* (vgl. [Engel 1988], S.393).
Bei allen starken Verben tritt Ablautung auf. Als Ablautung wird der systematische Wechsel des Stammvokals bezeichnet. Die ablautenden Verben stellen im Deutschen eine begrenzt offene Klasse dar; Neubildungen sind nur durch Präfigierung und Komposition möglich. Durch die Ablautung werden bis zu fünf verschiedene Stammformen (vgl. [Engel 1988], S.393) gebildet:

- die 1. Stammform, die zur Bildung des Infinitivs und aller Präsensformen mit Ausnahme der 2. und 3. Person Singular Indikativ Präsens verwendet wird;
- die 2. Stammform, die zur Bildung der 2. und 3. Person Singular Indikativ Präsens verwendet wird;
- die 3. Stammform, die zur Bildung des gesamten Paradigmas für das Präteritum verwendet wird;
- die 4. Stammform, die zur Bildung des gesamten Paradigmas für den Konjunktiv II verwendet wird;
- die 5. Stammform, die zur Bildung des Partizip II verwendet wird.

Dabei können einige dieser Stammformen zusammenfallen:

- die 1. und die 2. Stammform;
- die 1., die 2. und die 5. Stammform;
- die 3. und die 4. Stammform;
- die 3., die 4. und die 5. Stammform.

Je nach der Aufeinanderfolge der Stammformen werden die Ablaute zu Ablautreihen zusammengefaßt, wobei die Länge der ablautenden Vokale von Bedeutung ist. Im Deutschen existieren 39 Ablautreihen (vgl. [Drosdowski 1984], S.127).
Bei einigen ablautenden Verben tritt zusätzlich zum Vokalwechsel bei einigen Stammformen noch ein Wechsel des nachfolgenden Konsonanten auf (vgl. Tabelle 4.5).
Bezüglich ihres Allomorphie-Regularitätsgrades sind alle starken Verben semiirregulär; ihr Allomorphieverhalten läßt sich nicht aus der Grundformoberfläche

Verb	1. Pers.Sg. Präsens	1. Pers.Sg. Präteritum	Partizip II
sitzen	*sitz-e*	*saß*	*ge-sess-en*
ziehen	*zieh-e*	*zog*	*ge-zog-en*
schneiden	*schneid-e*	*schnitt*	*ge-schnitt-en*

Tabelle 4.5: Ablautung und Konsonantenwechsel bei einigen starken Verben

schließen. Die Allomorphe lassen sich allerdings regelhaft aus der Ablautreihe schließen, unter die das Verb fällt.
Die Flexion der starken Verben ähnelt in ihrem Gesamtbild der Flexion der schwachen Verben, wobei jedoch folgende Besonderheiten zu beachten sind:

- Ist die 2. Stammform die umgelautete Variante der 1. Stammform, oder wenn bei den beiden Stammformen ein Wechsel des Stammvokals von /e:/ zu /i:/ bzw. von /ε/ zu /i/ auftritt (*e/i*-Wechsel), und endet der Stamm auf einen Dental, so entfällt in der 2. Pers. Sg. Ind. Präsens die sonst bei Dentalstämmen auftretende e-Erweiterung; weiterhin die 3. Pers. Sg. Ind. Präsens wird durch Nullflexion gebildet.

- Im Allgemeinen wird zur Bildung des Imperativs die erste Stammform verwendet. Eine Ausnahme bilden die Verben, bei denen *e/i*-Wechsel auftritt; diese verwenden die 2. Stammform für die 2. Pers. Sg. Imperativ vertraulich, für die restlichen Imperativformen jedoch die 1. Stammform.

- Die starken Verben bilden ihren Präteritumstamm nicht durch Infigierung von *-t-* oder *-et-*, sondern durch Ablautung. Da dementsprechend nicht alle Präteritumstämme auf einen Dental enden, stehen, wo morphophonetisch möglich, wieder *-st* und *-t*. Im Konjunktiv Präteritum stehen jedoch immer die um *-e-* erweiterten Formen.

- Die 1. und 3. Person Singular Indikativ Präteritum wird nicht durch das Flexionssuffix *-e*, sondern durch Nullflexion gebildet.

- Das Partizip II wird nicht durch das Zirkumfix *ge-...-t*, sondern durch das Zirkumfix *ge-...-en* gebildet. Dabei kann nur eine der Stammformen zur Bildung des Partizip II verwendet werden. Welche dies ist, hängt allein von der Ablautreihe ab.

- Im allgemeinen ist die Distribution der Stammformen zu beachten (s.o.).

Gemischt starke und schwache Verben

Eine kleine Anzahl von Verben des Deutschen zeigt sowohl Merkmale der schwachen als auch der starken Verben.
Diese Verben (*rennen, kennen, brennen, nennen*) bilden das Präteritum und das Partizip II ebenso wie die schwachen Verben durch Suffigierung mit *t*. Sie weisen aber auch ein Hauptmerkmal der starken Verben, die Ablautung auf (vgl. Tabelle 4.7).
Weiterhin gibt es noch zwei Verben (*denken, bringen*) mit *t*-Suffix, Ablautung und Konsonantenwechsel (vgl. Tabelle 4.8).

		tragen	reiben	schneiden	halten	lesen
Indikativ	Sg. 1.	trag-e	reib-e	schneid-e	halt-e	les-e
Präsens	2.	träg-st	reib-st	schneid-est	hält-st	lies-t
	3.	träg-t	reib-t	schneid-et	hält	lies-t
	Pl. 1.	trag-en	reib-en	schneid-en	halt-en	les-en
	2.	trag-t	reib-t	schneid-et	halt-et	les-t
	3.	trag-en	reib-en	schneid-en	halt-en	les-en
Konjunktiv	Sg. 1.	trag-e	reib-e	schneid-e	halt-e	les-e
Präsens	2.	trag-est	reib-est	schneid-est	halt-est	les-est
	3.	trag-e	reib-e	schneid-e	halt-e	les-e
	Pl. 1.	trag-en	reib-en	schneid-en	halt-en	les-en
	2.	trag-et	reib-et	schneid-et	halt-et	les-et
	3.	trag-en	reib-en	schneid-en	halt-en	les-en
Indikativ	Sg. 1.	trug	rieb	schnitt	hielt	las
Präteritum	2.	trug-st	rieb-st	schnitt-est	hilet-est	las-t
	3.	trug	rieb	schnitt	hielt	las
	Pl. 1.	trug-en	rieb-en	schnitt-en	hielt-en	las-en
	2.	trug-t	rieb-t	schnitt-et	hielt-et	las-t
	3.	trug-en	rieb-en	schnitt-en	hielt-en	las-en
Konjunktiv	Sg. 1.	trüg-e	rieb-e	schnitt-e	hielt-e	läs-e
Präteritum	2.	trüg-est	rieb-est	schnitt-est	hielt-est	läs-est
	3.	trüg-e	rieb-e	schnitt-e	hielt-e	läs-e
	Pl. 1.	trüg-en	rieb-en	schnitt-en	hielt-en	läs-en
	2.	trüg-et	rieb-et	schnitt-et	hielt-et	läs-et
	3.	trüg-en	reib-en	schnitt-en	hielt-en	läs-en
Imperativ	Sg. 2.	trag-(e)	reib-(e)	schneid-(e)	halt-(e)	les-e/lies
Präsens	Pl. 1.	trag-en	reib-en	schneid-en	halt-en	les-en
vertr.	2.	trag-t	reib-t	schneid-et	halt-et	les-t
Imperativ Präsens dist.	Sg. 2.	trag-en	reib-en	schneid-en	halt-en	les-en
Infinitiv		trag-en	reib-en	schneid-en	halt-en	les-en
Partizip I		trag-end	reib-end	schneid-end	halt-end	les-end
Partizip II		ge-trag-en	ge-rieb-en	ge-schnitt-en	ge-halt-en	ge-les-en

Tabelle 4.6: Flexionsverhalten der starkenVerben

Verb	1. Pers.Sg. Präsens	1. Pers.Sg. Präteritum	Partizip II
rennen	renn-e	rann-t-e	ge-rann-t
kennen	kenn-e	kann-t-e	ge-kann-t

Tabelle 4.7: Gemeinsames Auftreten von Merkmalen der starken und schwachen Verben bei den gemischt stark-schwachen Verben

Verb	1. Pers.Sg. Präsens	1. Pers.Sg. Präteritum	Partizip II
denken	*denk-e*	*dach-t-e*	*ge-dach-t*
bringen	*bring-e*	*brach-t-e*	*ge-brach-t*

Tabelle 4.8: Konsonantenwechsel bei einigen gemischt stark-schwachen Verben

		rennen	denken
Indikativ Präsens	Sg. 1.	renn-e	denk-e
	2.	renn-st	denk-st
	3.	renn-t	denk-t
	Pl. 1.	renn-en	denk-en
	2.	renn-t	denk-t
	3.	renn-en	denk-en
Konjunktiv Präsens	Sg. 1.	renn-e	denk-e
	2.	renn-est	denk-est
	3.	renn-e	denk-e
	Pl. 1.	renn-en	denk-en
	2.	renn-et	denk-et
	3.	renn-en	denk-en
Indikativ Präteritum	Sg. 1.	rann-t-e	dach-t-e
	2.	rann-t-est	dach-t-est
	3.	rann-t-e	dach-t-e
	Pl. 1.	rann-t-en	dach-t-en
	2.	rann-t-et	dach-t-et
	3.	rann-t-en	dach-t-en
Konjunktiv Präteritum	Sg. 1.	rann-t-e	dach-t-e
	2.	rann-t-est	dach-t-est
	3.	rann-t-e	dach-t-e
	Pl. 1.	rann-t-en	dach-t-en
	2.	rann-t-et	dach-t-et
	3.	rann-t-en	dach-t-en
Imperativ Präsens vertr.	Sg. 2.	renn-(e)	denk-(e)
	Pl. 1.	renn-en	denk-en
	2.	renn-t	denk-t
Imperativ Präsens dist.	Sg. 2.	renn-en	denk-en
Infinitiv		renn-en	denk-en
Partizip I		renn-end	denk-end
Partizip II		ge-rann-t	ge-dach-t

Tabelle 4.9: Flexionsverhalten der gemischt stark-schwachen Verben

Ebenso wie die starken Verben sind die gemischt stark-schwachen Verben bezüglich ihres Allomorphieverhaltens semi-irregulär.
Die gemischt stark-schwachen Verben verhalten sich hinsichtlich ihrer Flexion wie schwache Verben. Es treten allerdings im Präsens und im Präteritum verschiedene Stämme auf.

Irreguläre Verben

Eine kleine Anzahl von Verben kann keiner der obigen drei Klassen zugeordnet werden. Dafür können einer oder mehrere der folgenden Gründe verantwortlich sein:

- Sie sind bezüglich ihres Allomorphieverhaltens irregulär; d.h., es existiert keine Möglichkeit, aus der Grundform alle Allomorphe abzuleiten. Dies ist zum Beispiel beim Hilfsverb *sein* der Fall.
- Es ist zwar möglich, alle Allomorphe regelhaft aus der Grundform abzuleiten; diese weisen aber Distribution und Flexionsverhalten auf, die nicht mit den Distributionen obiger drei Klassen verträglich sind. Beim Verb

dürfen stimmt beispielsweise die 1. Stammform nicht mit dem Infinitivstamm überein. Weiterhin bildet *dürfen* die 1. Person Singular Indikativ Präsens nicht wie alle anderen Verben durch das Flexionssuffix *-e*, sondern durch Nullflexion.

4.2.2 Die Deklination von Substantiven

Substantive sind Worte, die ein konstantes Genus haben (vgl. [Engel 1988], S.500). Die Deklination, also die Flexion der Substantive ist abhängig von zwei Parametern:

- Kasus. Es existieren vier Kasus: Nominativ, Genitiv, Dativ und Akkusativ.
- Numerus. Es existieren zwei Numeri: Singular und Plural.

Alle Substantivwortformen lassen sich hinsichtlich dieser Parametermatrix einteilen.
Als Klassifikationskriterium für die Substantive hat sich die Klassifikation nach der Flexionskategorie Numerus als angemessener erwiesen als eine Klassifikation nach der Flexionskategorie Kasus, da die Kasus im Deutschen in geringerem Ausmaß markiert sind als die Numeri (vgl. [Engel 1988], S.505). So fallen viele Kasusmarkierungen der deutschen Gegenwartssprache zusammen, während der Numerus in der überwiegenden Zahl markiert ist. Dementsprechend werden bei Engel die Substantive nach ihrer Pluralbildung in fünf Klassen eingeteilt; diese fünf Klassen werden dann noch nach der Bildung des Genitiv Singular und Umlautverhalten in 13 Subklassen aufgeteilt (vgl. [Engel 1988], S.505ff).
Die vorliegende Arbeit weicht von dieser Klassifikation geringfügig ab, insofern als die Pluralbildung nicht als übergeordnetes Klassifikationskriterium angesehen wird. Vielmehr werden die Substantive nach drei Kriterien klassifiziert, nämlich Bildung des Singular, Bildung des Plural und Umlautverhalten. Außerdem werden die Klassen/Subklassen Engels an einigen Stellen weiter differenziert. Die Klassifikation lehnt sich in dieser Hinsicht an die des Malaga-Vorgängersystems LAP an (vgl. [Schüller und Lorenz 1994]).
Im Singular werden 10 Klassen unterschieden (wobei der Nominativ immer unmarkiert ist) (vgl. Tabelle 4.10):

1. Singularparadigma ohne Flexionsendung: *Angst, Frau, Ärztin, Zunge*
2. Genitiv auf *-s* oder *-es*, Dativ auf *-e* oder unmarkiert, Akkusativ unmarkiert: *Mann, Amt, Eid*
3. Genitiv auf *-es*, Dativ unmarkiert, Akkusativ unmarkiert: *Dachs, Faß, Streß*
4. Genitiv auf *-s*, Dativ und Akkusativ unmarkiert: *Apartment, Koffer, Nickel*
5. Genitiv auf *-ses*, Dativ unmarkiert, Akkusativ unmarkiert: *Zirkus, Wagnis, As*
6. Genitiv, Dativ und Akkusativ auf *-en*: *Student, Akrobat, Bär*

	Nominativ	Genitiv	Dativ	Akkusativ
Klasse 1	Frau	Frau	Frau	Frau
	Ärztin	Ärztin	Ärztin	Ärztin
	Zunge	Zunge	Zunge	Zunge
Klasse 2	Mann	Mann-(e)s	Mann(-e)	Mann
	Amt	Amt-(e)s	Amt(-e)	Amt
	Eid	Eid-(e)s	Eid(-e)	Eid
Klasse 3	Dachs	Dachs-es	Dachs	Dachs
	Faß	Fass-es	Faß	Faß
	Streß	Stress-es	Stress	Stress
Klasse 4	Apartment	Apartment-s	Apartment	Apartment
	Koffer	Koffer-s	Koffer	Koffer
	Nickel	Nickel-s	Nickel	Nickel
Klasse 5	Zirkus	Zirkus-ses	Zirkus	Zirkus
	Wagnis	Wagnis-ses	Wagnis	Wagnis
	As	As-ses	As	As
Klasse 6	Student	Student-en	Student-en	Student-en
	Akrobat	Akrobat-en	Akrobat-en	Akrobat-en
	Bär	Bär-en	Bär-en	Bär-en
Klasse 7	Affe	Affe-n	Affe-n	Affe-n
	Neurologe	Neurologe-n	Neurologe-n	Neurologe-n
	Falke	Falke-n	Falke-n	Falke-n
Klasse 8	Name	Name-ns	Name-n	Name-n
	Funke	Funke-ns	Funke-n	Funke-n
	Buchstabe	Buchstabe-ns	Buchstabe-n	Buchstabe-n
Klasse 9	Herz	Herz-ens	Herz-en	Herz

Tabelle 4.10: Klassifikation nach Singularflexion

7. Genitiv, Dativ und Akkusativ auf *-n*: *Affe, Falke, Neurologe*
8. Genitiv auf *-ns*, Dativ und Akkusativ auf *-n*: *Name, Funke, Buchstabe*
9. Genitiv auf *-ens*, Dativ auf *-en*, Akkusativ unmarkiert: *Herz*
10. keine Singularbildung möglich, z.B.: *Eltern*

Beim Plural werden werden 11 Bildungsklassen unterschieden; die Unterscheidung beruht auf dem Vorhandensein eines Pluralsuffixes und der Markiertheit des Dativ (vgl. Tabelle 4.11).

1. Gesamtes Paradigma unmarkiert
2. Kein Pluralmarker, Dativ auf *-n*
3. Pluralmarker *-er*, Dativ auf *-n*
4. Pluralmarker *-e*, Dativ auf *-n*
5. Pluralmarker *-se*, Dativ auf *-n*
6. Pluralmarker *-en*, Dativ unmarkiert
7. Pluralmarker *-n*, Dativ unmarkiert
8. Pluralmarker *-ien*, Dativ unmarkiert
9. Pluralmarker *-nen*, Dativ unmarkiert

	Nom. Sg.	Nominativ	Genitiv	Dativ	Akkusativ
Klasse 1	Ofen	Öfen	Öfen	Öfen	Öfen
	Galgen	Galgen	Galgen	Galgen	Galgen
	Segen	Segen	Segen	Segen	Segen
Klasse 2	Becher	Becher	Becher	Becher-n	Becher
	Segel	Segel	Segel	Segel-n	Segel
	Acker	Äcker	Äcker	Äcker-n	Äcker
Klasse 3	Amt	Ämt-er	Ämt-er	Ämt-ern	Ämt-er
	Weib	Weib-er	Weib-er	Weib-ern	Weib-ern
	Mann	Männ-er	Männ-er	Männ-ern	Männ-er
Klasse 4	Nuß	Nüss-e	Nüss-e	Nüss-en	Nüss-e
	Molch	Molch-e	Molch-e	Molch-en	Molch-e
	Akkord	Akkord-e	Akkord-e	Akkord-en	Akkord-e
Klasse 5	Bus	Bus-se	Bus-se	Bus-sen	Bus-se
	Ärgernis	Argernis-se	Ärgernis-se	Ärgernis-sen	Ärgernis-se
	Primas	Primas-se	Primas-se	Primas-sen	Primas-se
Klasse 6	Student	Student-en	Student-en	Student-en	Student-en
	Fracht	Fracht-en	Fracht-en	Fracht-en	Fracht-en
	Landschaft	Landschaft-en	Landschaft-en	Landschaft-en	Landschaft-en
Klasse 7	Granate	Granate-n	Granate-n	Granate-n	Granate-n
	Woche	Woche-n	Woche-n	Woche-n	Woche-n
	Affe	Affe-n	Affe-n	Affe-n	Affe-n
Klasse 8	Indiz	Indiz-ien	Indiz-ien	Indiz-ien	Indiz-ien
	Kleinod	Kleinod-ien	Kleinod-ien	Kleinod-ien	Kleinod-ien
	Reptil	Reptil-ien	Reptil-ien	Reptil-ien	Reptil-ien
Klasse 9	Rivalin	Rivalin-nen	Rivalin-nen	Rivalin-nen	Rivalin-nen
	Asiatin	Asiatin-nen	Asiatin-nen	Asiatin-nen	Asiatin-nen
	Sünderin	Sünderin-nen	Sünderin-nen	Sünderin-nen	Sünderin-nen
Klasse 10	Studio	Studio-s	Studio-s	Studio-s	Studio-s
	Dossier	Dossier-s	Dossier-s	Dossier-s	Dossier-s
	Jet	Jet-s	Jet-s	Jet-s	Jet-s

Tabelle 4.11: Klassifikation nach Pluralflexion

10. Pluralmarker -*s*, Dativ unmarkiert

11. Keine Pluralbildung möglich

Eine eventuelle (semi-irreguläre) Umlautung des Pluralallomorphs geht in diese Klassifikation nicht ein; sie wird als unabhängiges Klassifikationsmerkmal betrachtet. Zwar ist Pluralumlautung an die Klassen 1, 2, 3 und 4 gebunden; sie tritt jedoch nicht bei allen umlautfähigen Vertretern dieser Klassen auf, so daß sich aus der Zugehörigkeit zu einer dieser Klassen nicht zwangsläufig das Umlautverhalten schließen läßt.

4.2.3 Die Flexion von Adjektiven

Adjektive sind Wörter ohne konstantes Genus, die zwischen Determinativen und Substantiven stehen können (vgl. [Engel 1988], S.556).
Bei Adjektiven treten zwei Formen der Flexion auf, die sowohl kombiniert als auch getrennt vorkommen können:

- Die Deklination: Werden Adjektive attributiv gebraucht (wenn sie also Teil einer Nominalphrase sind), so werden sie dekliniert, wobei sie in

Genus, Kasus und Numerus mit dem voranstehenden Determinativ kongruieren müssen. In allen anderen Verwendungen wird das Adjektiv nicht flektiert; man bezeichnet es dann als Adverb.

- Die Komparation (Steigerung): Adjektive können in drei Steigerungsformen (Positiv, Komparativ, Superlativ) auftreten.

Die Deklination von Adjektiven

Es können nur attributiv gebrauchte Adjektive dekliniert werden. Die Flexionskategorien von deklinierten Adjektiven sind:

- Art des Artikels: definiter Artikel, indefiniter Artikel, kein Artikel
- Genus: wie bei Substantiven
- Kasus: wie bei Substantiven
- Numerus: wie bei Substantiven

Dabei treten fünf Flexionsendungen auf: *-e, -er, -es, -en* und *-em*, die sich auf $3 \cdot 3 \cdot 4 \cdot 2 = 72$ Kombinationen aus Artikelart (3), Genus (3), Kasus (4) und Numerus (2) verteilen.
Bei definitem Artikel (und bei *aller, derjenige, derselbe, dieser, irgendwelcher, jener, mancher, solcher, welcher* in allen drei Genera) treten nur die Flexionsendungen *-e* und *-en* auf, wobei *-e* bei Maskulina nur im Nominativ Singular und bei Feminina und Neutra in Nominativ und Akkusativ Singular steht. Bei allen anderen Formen steht *-en* (vgl. Tabelle 4.12).
Bei Adjektiven, deren Grundform auf *-er, -en* oder *-el* endet, existiert ein um *-e-* verkürztes Allomorph; diese Adjektive verhalten sich also semi-regulär. Die Allomorphe sind dabei verteilt wie folgt:

- Das verkürzte Allomorph wird bei der Deklination und der Bildung des Komparativ (s.u.) verwendet.
- Bei den Adjektiven die auf *-er* oder *-en* enden, kann das nicht verkürzte Allomorph adverbial gebraucht, dekliniert und zur Bildung von Komparativ und Superlativ verwendet werden. Bei den auf *-el* endenden Adjektiven kann das nicht verkürzte Allomorph nur adverbial und zur Bildung des Superlativ verwendet werden.

Bei Adjektiven, deren Grundform auf *-e* endet (z.B. *blöde*), existiert ein um das *-e* verkürztes Allomorph. Das verkürzte Allomorph kann hierbei adverbial gebraucht, dekliniert und zur Bildung von Komparativ und Superlativ verwendet werden, wohingegen das nicht verkürzte Allomorph nur adverbial und zur Bildung des Superlativ verwendet werden kann.
Bei indefinitem Artikel (und bei *manch ein, solch ein, irgendein, einige, etliche, mehrere* in allen drei Genera) treten vier Flexionsendungen auf, nämlich *-e, -er, -es* und *-en*. Die Distribution der Endungen ist somit komplexer als bei definitem Artikel (vgl. 4.13); jedoch sind auch hier die Endungen eines Pluralkasus unabhängig vom Genus.
Bei nicht vorhandenem Artikel (und bei *manch, solch, welch* in allen drei Genera und *deren, dessen*) treten fünf Flexionsendungen auf (vgl. Tabelle 4.14), nämlich *-e, -er, -es, -en* und *-em*. Wieder sind die Endungen eines Pluralkasus unabhängig vom Genus.

Numerus	Kasus	Maskulinum	Femininum	Neutrum
Singular	Nom.	der gute Wein	die gute Speise	das gute Brot
	Gen.	des guten Weines	der guten Speise	des guten Brotes
	Dat.	dem guten Wein	der guten Speise	dem guten Brot
	Akk.	den guten Wein	die gute Speise	das gute Brot
Plural	Nom.	die guten Weine	die guten Speisen	die guten Brote
	Gen.	der guten Weine	der guten Speisen	der guten Brote
	Dat.	den guten Weinen	der guten Speisen	den guten Broten
	Akk.	die guten Weine	die guten Speisen	die guten Brote

Tabelle 4.12: Deklination des Adjektivs nach definitem Artikel

Numerus	Kasus	Maskulinum	Femininum	Neutrum
Singular	Nom.	ein guter Wein	eine gute Speise	ein gutes Brot
	Gen.	eines guten Weines	einer guten Speise	eines guten Brotes
	Dat.	einem guten Wein	einer guten Speise	einem guten Brot
	Akk.	einen guten Wein	eine gute Speise	ein gutes Brot
Plural	Nom.	einige gute Weine	einige gute Speisen	einige gute Brote
	Gen.	einiger guten Weine	einiger guten Speisen	einiger guten Brote
	Dat.	einigen guten Weinen	einigen guten Speisen	einigen guten Broten
	Akk.	einige gute Weine	einige gute Speisen	einige gute Brote

Tabelle 4.13: Deklination des Adjektivs nach definitem Artikel

Numerus	Kasus	Maskulinum	Femininum	Neutrum
Singular	Nom.	guter Wein	gute Speise	gutes Brot
	Gen.	guten Weines	guter Speise	guten Brotes
	Dat.	gutem Wein	guter Speise	gutem Brot
	Akk.	guten Wein	gute Speise	gutes Brot
Plural	Nom.	gute Weine	gute Speisen	gute Brote
	Gen.	guter Weine	guter Speisen	guter Brote
	Dat.	guten Weinen	guten Speisen	guten Broten
	Akk.	gute Weine	gute Speisen	gute Brote

Tabelle 4.14: Deklination des Adjektivs nach definitem Artikel

Die Komparation von Adjektiven

Die zweite Art und Weise, wie Adjektive flektiert werden können, ist die Steigerung oder Komparation. Es existieren drei Komparationsgrade: Positiv, Komparativ und Superlativ.
Der Positiv ist die unmarkierte Form des Adjektivs; er ist gleichlautend mit der Grundform.
Der Komparativ wird durch Suffigierung der Grundform mit *-er* gebildet. Bei einigen umlautfähigen Adjektiven existiert ein zweites, semi-irregulär umgelautetes Komparativallomorph (z.B. *arm* – *ärm-er*). Bei Adjektiven, deren Grundform auf *-er, -en* oder *-el* endet, kann ein zweites, um *-e-* verkürztes Allomorph verwendet werden (zur Distribution siehe oben). Bei einigen Adjektiven wird im Komparativ eine (irreguläre) Suppletivform verwendet (z.B. *gut* – *bess-er*). In der Regel drückt der Komparativ aus, daß die durch das Adjektiv ausgedrückte Eigenschaft in stärkerem Maße vorhanden ist.[2]
Der Superlativ wird durch Suffigierung der Grundform mit *-st* gebildet; bei Stämmen, die auf einen Sibilanten außer /ʃ/ (<*sch*>) enden, wird mit *-est* suffigiert (vgl. [Drosdowski 1984], S.308). Bei Stämmen, die auf Dental oder /ʃ/ enden, kann fakultativ mit *-st* oder *-est* suffigiert werden (vgl. [Drosdowski 1984], S.308). Wenn ein umgelautetes Komparativallomorph existiert, so wird dies auch im Superlativ verwendet (z.B. *arm* – *ärm-st*) (vgl. [Engel 1988], S.562). Wird im Komparativ eine Suppletivform verwendet, so wird auch der Superlativ mit einem Suppletivallomorph gebildet; Komparativ- und Superlativallomorph müssen jedoch nicht identisch sein (z.B. *gut* – *be-st*). In der Regel drückt der Superlativ aus, daß die durch das Adjektiv ausgedrückte Eigenschaft in höchstmöglichem Maße vorhanden ist.
Adjektive können in allen drei Komparationsgraden zusätzlich dekliniert werden. Ist dies der Fall, so wird das Komparativ- bzw. Spuerlativsuffix immer vor dem Deklinationssuffix direkt an den Stamm geheftet.

4.2.4 Andere Wortarten

Außer Substantiven, Adjektiven und Verben werden nur noch Eigennamen flektiert. Alle anderen Wortklassen bleiben unflektiert.
Eigennamen (Nomina Propria) werden nur nach dem Kasus flektiert, wobei nur der Genitiv markiert ist:

- Eigennamen, die auf einen Sibilanten enden, werden im Genitiv (orthographisch) durch Anhängen eines Apostroph markiert (z.B. *Thomas' Hans'*, *Lorenz'*).
- Alle anderen Eigennamen werden im Genitiv durch Suffigierung mit *-s* markiert (z.B. *Peters, Meiers, Münchens*).

4.3 Die Derivation

Die Derivation kommt in beiden behandelten Varianten (Prä- und Suffigierung) bei allen drei Lexemwortarten vor. Die Wohlgeformtheit hängt dabei meist

[2] Zu Ausnahmen bei der Semantik der Komparation vgl. [Engel 1988], S.563f

sowohl von morphologisch-formalen wie auch von semantisch motivierten Kriterien ab. Während die Präfigierung in den meisten Fällen reihenbildend und unproduktiv ist, existiert bei der Suffigierung große Varianz bezüglich der Produktivität.
Derivationsphänomene werden in der vorliegenden Arbeit nur behandelt, wenn sie entweder reihenbildend sind und somit lexikalisiert werden können, oder wenn sie hochproduktiv sind und dabei aufgrund morphologischer oder syntaktischer Kriterien auf die formale Wohlgeformtheit der Derivation geschlossen werden kann. Da semantische Information in den vorliegenden Grundformlexika nicht vorhanden ist, können semantische Kriterien nicht zur Entscheidung der Wohlgeformtheit herangezogen werden. Zwar generiert die auf dieser Beschreibung basierende Grammatik über, was jedoch bei produktiven Derivationsphänomenen weniger ins Gewicht fällt und auch in Anbetracht der Tatsache, daß die Grammatik zur Wortformerkennung herangezogen werden soll, weniger problematisch ist.
Bei reihenbildenden Präfigierungen oder Suffigierungen wird die Möglichkeit der Affigierung als beim Grundwort lexikalisiert betrachtet; die lexikalische Beschreibung der derivierbaren Lexeme muß also Listen von möglichen Affixen enthalten. Eine Ausnahme hiervon bilden nicht abtrennbar präfigierte Verben (s.u.); da Präfix und Stamm immer direkt zusammenbleiben, wird das ganze präfigierte Verb als eigener Eintrag ins Lexikon aufgenommen.

4.3.1 Präfigierung

Präfigierung von Verben

Im Deutschen existieren zwei Arten von Verbpräfigierungen: Verben mit trennbarem Präfix und Verben mit nichttrennbarem Präfix.
Die Unterscheidung gründet dabei auf dem syntaktischen, dem morphologischen und dem suprasegmentalen Verhalten der Präfixverben.
Bei Verben mit trennbarem Präfix wird das Präfix im Satzgefüge vom Verb abgetrennt, wenn das Verb in finiter Form vorliegt und in Erst- oder Zweitstellung steht (also wenn es das Prädikat eines Hauptsatzes ist (vgl [Helbig und Buscha 1993], S.222)). Steht das Verb in Endstellung, so wird das Präfix nicht abgetrennt. Bei infiniten Verbformen wird das Präfix ebenfalls nicht abgtrennt; beim Infinitiv mit *zu* und beim Partizip II steht das Präfix vor den Präfixen *-zu-* bzw. *-ge-* (wie z.B. in *an-zu-rufen* bzw. *an-ge-rufen*). Bei allen trennbaren Präfixverben liegt der Hauptakzent auf dem Präfix.
Trennbar vorkommende Verbpräfixe sind hauptsächlich (vgl [Engel 1988], S.440f, [Helbig und Buscha 1993], S.222f):

> *ab, an, auf, aus, bei, durch, ein, empor, fehl, fort, her, hin, los, mit, nach, nieder, über, um, unter, vor, weiter, wider, wieder, zu, zurecht, zusammen*

Neben diesen Präfixen treten noch weitere Präfixe oder Präfixoide trennbar auf (vgl. [Mater 1966]).
Die Verbpräfixe können auch aus mehreren Präfixen zusammengesetzt sein, wobei das gesamte zusammengesetzte Präfix abgetrennt wird (vgl. [Helbig und Buscha 1993], S.223):

> *dabei, daher, darüber, darunter, davor, dahinter, daneben, darin, dagegen, dafür, dran (< daran), heraus, herauf, heran, herunter, herein (> rein), umher, vorher, hinauf, hinaus, hinweg, hinfort, hinüber, hinunter, daraufhin, zurecht, zuvor, zurück, zusammen, hinzu, durcheinander, miteinander, auseinander, aufeinander, übereinander, untereinander, voreinander, hintereinander, miteinander, nebeneinander, rückwärts, vorwärts*

Bei nichttrennbaren Präfixverben verbleibt das Präfix immer direkt am Stamm (vgl. [Engel 1988], S.439). So wird auch der Infinitiv mit *zu* nicht mit morphologischen Mitteln gebildet, sondern durch syntaktische Fügung. Das Partizip II von nichttrennbar präfigierten Verben wird immer ohne *ge-* gebildet. Der Hauptakzent liegt immer auf dem Verbstamm.
Nichttrennbar vorkommende Verbpräfixe sind:

> *be, ent, er, ge, hinter, miß, ver, zer*

Einige Präfixe können sowohl trennbar als auch nichttrennbar vorkommen (vgl. [Helbig und Buscha 1993], S.223):

> *durch, hinter, über, um, unter, wider, wieder*

Nichttrennbar präfigierte Verben (vgl. [Helbig und Buscha 1993], S.223) können zusätzlich noch trennbar präfigiert werden (z.B. *ab-be-rufen*); dabei wird nur das erste Präfix abgetrennt (z.B. *er beruft ab*). Das Partizip II wird ohne *ge-* gebildet (*abberufen*), beim Infinitiv mit *zu* steht *-zu-* zwischen den beiden Präfixen (*ab-zu-be-rufen*).
Die Präfigierung kann, muß jedoch nicht in einer Änderung des syntaktischen Verhaltens resultieren (vgl. [Erben 1983], S.74).
So kann sich zum einen der Valenzrahmen des Verbs ändern:

> *jemandem etwas geben – jemanden aufgeben*
> *arbeiten – jemanden bearbeiten*
> aber: *etwas bewegen – etwas umherbewegen*

Desweiteren kann sich das Hilfsverb-Komplement zum Partizip II ändern:

> *hat geblutet – ist ausgeblutet*
> *hat gedampft – ist verdampft*
> *ist gelaufen – hat (sich) verlaufen*

Die Änderung des syntaktischen Verhaltens ist in der Regel als Resultat der durch die Präfigierung stattfindenden semantischen Modifikation zu betrachten (vgl. [Erben 1983], S.75). Da die vorliegende Arbeit (wie bereits unter 4.1.3 festgestellt) auf die Beschreibung morphosemantischer Relationen verzichten muß, kann die aus der semantischen Modifikation resultierende Veränderung des syntaktischen Verhaltens nicht regelhaft abgeleitet werden. Daher muß die Präfigierung eines Verbs als reihenbildend lexikalisiert betrachtet werden, was auch in Anbetracht der Tatsache, daß die Präfigierungen meist nicht produktiv, sondern reihenbildend sind, angemessen erscheint.

Präfigierung von Substantiven

Bei der Präfigierung von Substantiven ist zwischen hochproduktiven und reihenbildenden Präfixen zu unterscheiden. Präfigierungen von geringerer Produktivität werden als Simplizia betrachtet.
Hochproduktiv sind die heimischen Präfixe *Haupt-* (vgl. [Fleischer und Barz 1982], S.200), *Nicht-* (vgl. [Fleischer und Barz 1982], S.202), *Ur-* (vgl. [Fleischer und Barz 1982], S.203) und *Hilfs-*. Ebenfalls hochproduktiv sind die Fremdpräfixe *Mini-, Super-, Ex-* und *Extra-* (vgl. [Erben 1983], S.81).
Reihenbildende Präfixe treten da auf, wo ein reihenbildend präfigierbarer Verbstamm durch implizite Derivation zum Substantiv wird, wie z.B. bei *Gabe (< geben): Auf-gabe, An-gabe, Wieder-gabe, Weiter-gabe, Weg-gabe, Vor-gabe, Zu-gabe*. Allerdings können meist nicht alle Verbpräfixe auch vor das implizit abgeleitete Substantiv gestellt werden; so kann *geben* zwar mit *acht* und *zufrieden* präfigiert werden; ∗*Achtgabe* und ∗*Zufriedengabe* sind jedoch nicht möglich. Aus diesem Grund und weil die implizite Derivation in der vorliegenden Arbeit nicht regelhaft abgeleitet sondern als lexikalisiert betrachtet wird, wird auch diese Art der Substantivpräfigierung als reihenbildend lexikalisiert behandelt.

Präfigierung von Adjektiven

Präfigierungen von Adjektiven sind im Deutschen selten (vgl. [Fleischer und Barz 1982], 269). Die einzige Ausnahme hiervon bildet das hochproduktive Präfix *un-*.

4.3.2 Suffigierung

Zwar existiert im Deutschen eine große Anzahl von Ableitungssuffixen; von diesen ist jedoch nur eine sehr kleine Anzahl in hohem Maße produktiv. Die Produktivität ist semantisch beschränkt. Transitivität und Reflexivität werden bei keinem der deverbalen Suffixe als Wohlgeformtheitskriterium verwendet, da bei allen sowohl transitive, intransitive und reflexive Stämme die Basis bilden können.
Hochproduktive Ableitungssuffixe sind:

deverbal:

- *-er* bildet maskuline Substantive. Das Suffix kann an Infinitvstämme starker und schwacher Verben angehängt werden (vgl. [Fleischer und Barz 1982], S.151). Bei semi-regulären Verben mit Infinitv *-eln* wird vorzugsweise der um *-e-* verkürzte Stamm verwendet (z.B. *Bettl-er*), bei den auf *-ern* endenen der unverkürzte Stamm (z.B. *Ruder-er*).
- *-erin* bildet analog zu *-er* feminine Substantive.
- *-ung* bildet feminine Substantive und ist das produktivste deverbale substantivbildende Suffix des Deutschen (vgl. [Fleischer und Barz 1982], S.172). Es kann an Infinitvstämme starker und schwacher Verben angehängt werden.
- *-bar* bildet Adjektive und stellt das produktivste deverbale adjektivbildende Suffix des Deutschen dar (vgl. [Fleischer und Barz 1982], S.252). Es kann an Infinitvstämme starker und schwacher Verben angehängt werden.

desubstantivisch:
- Ableitungen mit den Diminutivsuffixen *-chen* und *-lein* können von fast allen Substantiven gebildet werden (vgl. [Fleischer und Barz 1982], S.178ff). Bei umlautfähigem Stammvokal des Substantivs existiert ein umgelautetes Diminutivallomorph des Stamms. Die Ableitung bildet neutrale Substantive.
- *-mäßig* (vgl. [Fleischer und Barz 1982], S.265) bildet Adjektive und kann an Substantivstämme (auch mit Fuge, vgl. 4.4) angehängt werden.

deadjektivisch:
- Mit *-heit* und den kombinatorischen Varianten *-keit* und *-igkeit* können fast alle Adjektive suffigiert werden (vgl. [Fleischer und Barz 1982], S.158ff). Die Ableitung bildet feminine Substantive. Die Distribution der Varianten hängt von der Silbenstruktur der Adjektivstämme ab (vgl. [Drosdowski 1984], S.473): einsilbige Adjektive verwenden die Variante *heit*; Adjektive mit den Endungen *-ig, -bar, -isch, -lich* und *-sam* sowie Adjektive, deren Stamm auf Konsonanten gefolgt von *-er* enden, verwenden die Variante *-keit*; Adjektive, die auf *-haft* oder *-los* enden verwenden die Variante *-igkeit*; alle anderen verwenden ebenfalls die Variante *-heit*.

4.4 Die Komposition

Durch Komposition werden zwei Lexeme (also Substantive, Adjektive und Verben) zu einer komplexen Wortform kombiniert. Das erste Kompositionsglied wird dabei Bestimmungswort oder Erstglied, das zweite Grundwort oder Zweitglied genannt.
Während als Zweitglied potentiell alle Lexemallomorphe zugelassen werden können (da die Wortform noch fortgesetzt werden kann), kommen als Erstglied nur bestimmte Allomorphe in Frage:

- bei Substantiven können nur Allomorphe verwendet werden, die zumindest zur Bildung einer Singularwortform verwendet werden lönnen (außer bei Kombination mit Fuge, s.u.);
- bei Verben können nur Präsensallomorphe verwendet werden;
- bei Adjektiven können nur Allomorphe verwendet werden, die auch deklinierbar sind.

Sowohl Erst- als auch Zweitglied können, wenn möglich, sowohl präfigiert als auch suffigiert vorliegen.
Zwischen Erst- und Zweitglied kann noch ein sogenanntes Fugenzeichen treten, wenn das Erstglied ein Substantiv oder ein Verb ist. Dabei gelten für die Fugenzeichen folgende vereinfachte Regeln[3]:

[3]Tatsächlich hängt die Wahl des korrekten Fugenzeichens und ob überhaupt ein Fugenzeichen eingefügt wird nicht allein von der Flexionsklasse, sondern auch von einigen suprasegmentalen Eigenschaften von Erst- und Zweitglied ab (vgl. dazu [Erben 1983], S.59), über die in den Grundfromlexika keine Information vorliegt und die auch nicht immer in den Allomorphregeln abgeleitet werden können; die hier angegebenen Regeln liefern jedoch eine gute Approximation.

Erstglied / Zweitglied	Substantiv	Adjektiv	Verb
Substantiv	sehr häufig	häufig	selten
Adjektiv	häufig	mäßig häufig	selten
Verb	häufig	häufig	selten

Tabelle 4.15: Häufigkeit von Kompositionsmustern

- Ist das Erstglied ein Verbstamm, so kann das Fugenzeichen *-e-* zwischen Verbstamm und Zweitglied treten (vgl. [Engel 1988], S.520).
- Ist das Erstglied ein Substantivstamm, so muß das Fugenzeichen irgendeine für das Substantiv und den Numerus (oder die Numeri) des Stammes gültige Flexionsendung sein (mit Ausnahme der Dativflexionsendungen beider Numeri) (vgl. [Engel 1988], S.520). Auf diese Art und Weise können auch reine Pluralstämme an der Komposition teilnehmen.

Prinzipiell können alle Lexeme aller drei Wortklassen sowohl als Erst- als auch als Zweitglied auftreten; allerdings kommen einige Kompositionsmuster weitaus häufiger vor als andere. In [Drosdowski 1984] werden für die verschiedenen Kompositionsmuster die in der Tabelle 4.15 dargestellten Häufigkeiten angegeben (vgl. [Drosdowski 1984], S.407).

Kapitel 5

Das Grammatiksystem

Nachdem im vorigen Kapitel das in der Arbeit betrachtete Fragment des Deutschen vorgestellt wurde, soll jetzt die Umsetzung der im vorigen Kapitel dargestellten linguistischen Fakten dokumentiert werden.

5.1 Einführung

Die Umsetzung der linguistischen Fakten macht es erforderlich, die vorhandenen Phänomene in linguistisch adäquater Form zu gruppieren, um einen Überblick über die umzusetzenden Kernprobleme zu erhalten.
Im Idealfall resultieren diese identifizierten Kernprobleme bereits in einer grundsätzlichen Konzeption des Grammatiksystems. Bei der Morphologie des Deutschen erscheinen folgende Konzepte von grundsätzlicher Bedeutung:

Allomorphie. Die Allomorphie des Deutschen läßt sich weitestgehend regelhaft ableiten. Allerdings muß die Ableitung der Allomorphe in Abhängigkeit von der Wortklasse geschehen, da die Ursachen für die Allomorphie sich bei den verschiedenen Wortklassen unterscheidet.

Flexion. Die Flexion ist einer der wichtigsten kombinatorischen Prozesse des Deutschen. Auf abstraktestem Niveau läßt sich der Flexionsprozeß als Konkatenation eines Stammes mit einem Flexionssuffix auffassen. Eine prinzipielle Trennung der Lexem-Wortklassen hinsichtlich der Flexion scheint nicht erforderlich, wenn es gelingt, die Kombinatorik von Stämmen und Endungen so weit wie möglich in den Allomorpheinträgen und so wenig wie möglich explizit in den beteiligten Regeln zu kodieren.

Wortbildung. Obwohl die Wortbildung ein sehr komplexes Themengebiet ist, lassen sich doch in ähnlich abstrakter Weise wie bei der Flexion drei grundsätzliche kombinatorische Prozesse isolieren:

- ein Nicht-Lexem wird vorne an ein Lexem angeheftet (Präfigierung);
- ein Nicht-Lexem wird hinten an ein Lexem angeheftet (Suffigierung)[1];

[1]Im Grunde genommen stellt die Flexion einen Spezialfall dieses Prozesses dar. Es wurde bei der Entwicklung des Regelsystems auch erwogen, die Flexion in die Suffigierung zu integrieren. Die Überlegungen wurden jedoch verworfen; zum einen, da die Steuerung der

- zwei Lexeme werden aneinandergeheftet (Komposition).

Auch bei diesen Teilkonzepten wäre es wünschenswert, diese mit wortklassenunabhängigen Regeln auf hohem Abstraktionsniveau zu beschreiben. Dazu ist es, wie bei der Flexion, notwendig, die Kombinatorik der beteiligten Elemente soweit wie möglich bereits in den Allomorpheinträgen zu kodieren.

Diese Konzepte richten das Augenmerk des Grammatikentwicklers in hohem Maße auf die Allomorphregeln. Abgesehen von ihrer eigentlichen Hauptaufgabe, der Erzeugung verschiedener Allomorphe, müssen sie auch Information über die Kombinatorik der Allomorphe erzeugen, da diese Information im Grundformlexikon nur rudimentär bzw. implizit in anderen, sehr allgemeinen Kategorien vorhanden ist. Im Laufe der Entwicklung dieser Grammatik hat sich gezeigt, daß durch die Technik, kategoriale Information bereits in den Allomorphregeln regelhaft zu erzeugen, das Abstraktionsniveau der Kombinationsregeln stark erhöht werden konnte.

5.2 Die Lexika

5.2.1 Herkunft der Lexika

Alle im vorliegenden Grammatiksystem benutzten Lexika befinden sich im Verzeichnis `lex` und stammen aus folgenden Quellen:

- Das Verblexikon wurde im Rahmen einer Magisterarbeit an der Abteilung für Computerlinguistik der Universität Erlangen-Nürnberg von Irina Gutbrod erstellt (vgl. [Gutbrod 1995]). Es basiert auf Erich Maters *Deutsche Verben* [Mater 1966], von dem die Bände 6 („Rektionsarten"), 7 („Verhältnis zu Reflexivpronomina"), 8 („Präteritalbildung mit ‚haben' oder ‚sein'") und 9 („Präfixtrennung) durch Scannen in elektronische Form überführt und dann in Malaga-Format konvertiert wurden. Das daraus resultierende Lexikon enthält Informationen über Valenzrahmen, Partizip-II-Komplement *haben* oder *sein* sowie reihenbildende Präfigierung. Die Information über die Ablautklasse ablautender Verben und über Partizip-II-Bildung mit oder ohne *ge-* wurde manuell bzw. unter Zuhilfenahme anderer Quellen in das Lexikon eingetragen.

- Die Lexika für Substantive und Adjektive wurden vom Vorgängersystem LAP (vgl. [Schüller und Lorenz 1994]) übernommen und vom Autor der vorliegenden Arbeit in ein für Malaga taugliches Format überführt.

- Das Namenslexikon wurde von Marco Zierl mittels eines Flex-Skriptes aus der Frankfurter Rundschau extrahiert.

- Das Lexikon der Partikeln wurde vom Markus Schulze im Rahmen einer Malaga-Syntax erstellt. Das Lexikon der Abkürzungen wurde ebenfalls von Markus Schulze erstellt.

- Alle anderen Lexika wurden im Rahmen dieser Arbeit neu entwickelt.

Regelpfade dadurch komplexer geworden wäre; nicht zuletzt jedoch auch mit Rücksicht auf die Tradition der Sprachbeschreibung des Deutschen.

5.2.2 Verbeinträge

Alle Verbeinträge (mit Ausnahme der irregulären Verben) befinden sich in der Datei `lex/verben.lex` (siehe auch Anhang E.1). Jeder Verbeintrag muß folgende Attribute enthalten:

- Das Attribut `LemOFl` enthält die Oberfläche des Lemmas (die Grundform) als Zeichenkette. Lautet das Verb ab, so muß der ablautende Vokal in geschweifte Klammern eingeschlossen werden. Tritt ebenfalls Konsonantenwechsel auf, so müssen die wechselnden Konsonanten ebenfalls in die geschweiften Klammer aufgenommen werden.
- Das Attribut `Klasse` hat bei Verben immer den Wert `Verb`.
- Das Attribut `PII-Auxil` enthält das Hilfsverb, das das Komplement zum Partizip II enthält. Es ist eine Liste; mögliche Listenelemente sind `PII-haben` und `PII-sein`.
- Das Attribut `PII-Ge` enthält die Information darüber, ob das Partizip II mit oder ohne Präfigierung durch *ge-* gebildet wird. Mögliche Werte sind `Ge` (Partizipbildung mit *ge-*) oder `Ge-` (Partizipbildung ohne *ge-*).

Folgende Attribute können vorhanden sein:

- Das Attribut `AK` enthält den Verweis auf die Ablautreihe, die in der Datei `rules/i_dmm.ablaut` genauer spezifiziert wird.
- Das Attribut `Valenzen` enthält eine Liste von alternativen Valenzrahmen. Jeder Valenzrahmen ist wiederum eine Liste von Valenzen. Die möglichen Listenelemente (also Valenzen) sind in der Segmentatei `rules/dmm.seg` aufgeführt.
- Das Attribut `Präfixe` ist eine Liste von möglichen Präfigierungen. Jede mögliche Präfigierung ist ein Verbund, der die Attribute `PII-Auxil`, `PII-Ge` (beide siehe oben) und das Attribut `PxOFl` mit der Oberfläche des Präfixes als Zeichenkette enthalten muß. Zusätzlich kann der Verbund noch das Attribut `Valenzen` enthalten.

5.2.3 Substantiveinträge

Alle Lexikoneinträge von Substantiven befinden sich in der Datei `lex/substantive.lex` (siehe auch Anhang E.2). Jeder Substantiveintrag muß folgende Attribute enthalten:

- Das Attribut `LemOFl` enthält die Oberfläche des Lemmas (die Grundform) als Zeichenkette. Lautet der Stammvokal im Plural um, so muß er in geschweiften Klammern eingeschlossen werden. Bei Wörtern griechisch-lateinisch-italienischen Ursprungs, die ihre fremdsprachliche Flexion teilweise beibehalten oder die heimische Ersatzendungen verwenden, muß die fremdsprachliche Endung der Grundform in geschweifte Klammern eingeschlossen werden. Bei Substantiven, die auf einen kurzen Vokal gefolgt von `"ß"` enden, muß der kurze Vokal dadurch gekennzeichnet werden, daß ihm ein Apostroph `"'"` folgt.

- Das Attribut `Klasse` kann entweder den Wert `Substantiv` oder den Wert `OblPxSubstantiv` (Substantiv, das noch präfigiert werden muß) haben. Hat `Klasse` den Wert `OblPxSubstantiv`, so muß zusätzlich das Attribut `Präfixe` (s.u.) enthalten sein.
- Das Attribut `SgDekl` enthält das Singulardeklinationsverhalten des Substantivs.
- Das Attribut `PlDekl` enthält das Pluraldeklinationsverhalten des Substantivs.
- Das Attribut `Genus` enthält das Genus des Substantivs. Mögliche Werte sind `Maskulinum`, `Femininum` und `Neutrum`.

Desweiteren können folgende Attribute vorhanden sein:

- Das Attribut `Präfixe` enthält eine Liste von möglichen Präfigierungen. Jede mögliche Präfigierung ist ein Verbund, der nur das Attribut `PxOFl` mit der Oberfläche des Präfixes als Zeichenkette enthält.
- Das Attribut `Fuge` ist eine Liste von Sonderfugen, also solchen Fugen, die nicht mit gültigen Flexionsendungen des Substantivs zusammenfallen. Mögliche Listenelemente sind `Fuge_s` und `Fuge_en`.

5.2.4 Adjektiveinträge

Alle Adjektive (mit Ausnahme der irregulären Paradigmen) befinden sich in der Datei `lex/adjektive.lex` (siehe auch Anhang E.3). Es müssen in jedem Eintrag folgende Attribute vorhanden sein:

- Das Attribut `LemOFl` enthält die Lemmaoberfläche (die Grundform) als Zeichenkette. Lautet der Stammvokal bei der Komparation um, so muß er dadurch gekennzeichnet werden, daß er in geschweifte Klammern eingeschlossen wird. Bei Adjektiven, die auf einen kurzen Vokal gefolgt von `"ß"` enden, muß der kurze Vokal dadurch gekennzeichnet werden, daß ihm ein Apostroph `"'"` folgt.
- Das Attribut `Klasse` enthält bei allen Adjektiven den Wert `Adjektiv`.
- Das Attribut `AdjFlex` enthält Information darüber, ob ein Adjektiv adverbial gebraucht und dekliniert werden kann und ob die Bildung von Komparativ und Superlativ möglich ist.

Zusätzlich können noch folgende Attribute vorhanden sein:

- Das Attribut `PhonEnd` enthält Information darüber, mit welchem Suffix der Superlativ gebildet wird. Mögliche Werte sind die Segmente `e-Einfügung`, `s-Tilgung`, `unmarkiert` sowie das Multi-Segment `unmarkiert_e-Einfügung`. Das Attribut ist nur vorhanden, wenn die Superlativbildung nicht korrekt von den Allomorphregeln geschlossen werden kann.
- Das Attribut `HeitSx` ist vorbelegt, wenn die Variante(n) des Derivationssuffixes *-heit*, die an das Adjektiv angehängt werden können, von den Allomorphregeln nicht korrekt geschlossen werden kann. Das Attribut ist eine Liste; mögliche Listenelemente sind `heit`, `keit` und `igkeit`.

5.2.5 Lexikoneinträge von Flexionsendungen

Alle an der Flexion beteiligten Endungen befinden sich in der Datei `lex/flexend.lex` (siehe auch Anhang E.5). Eine Ausnahme hiervon bilden die Präteritalinfixe, das Komparativsuffix und die Superlativsuffixe, die nicht wie Flexionsendungen, sondern wie Suffixe behandelt werden.
Alle Flexionsendungen müssen folgende Attribute enthalten:

- Das Attribut `LemOFl` enthält die Oberfläche der Flexionsendung als Zeichenkette.
- Desweiteren muß das Attribut `FlexSx` enthalten sein, an dem Flexionsendungen erkannt werden. Das Attribut enthält eine Liste von möglichen Kombinatoriken der Flexionsendung (s.u.).

Jede mögliche Kombinatorik in der Liste `FlexSx` ist ein Verbund, der folgende Attribute enthält:

- Das Attribut `StammKlasse` gibt an, welche Wortklasse der Stamm haben muß, damit die Flexionsendung konkateniert werden darf.
- Das Attribut `Flexion` ist eine Liste von im Stamm zu überprüfenden Merkmalen. Jedes zu überprüfende Merkmal ist wiederum eine Liste, wobei das erste Listenelement der Name des im Stamm zu überprüfenden Attributs ist und das zweite Element der Wert, mit dem der Wert dieses Attributs kongruieren soll. Bei einigen zu überprüfenden Attribut-Werte-Paaren enthält diese Liste noch ein drittes Element; ist dies das Segment `exakt`, so wird nicht auf Kongruenz, sondern auf Identität geprüft; ist das dritte Element das Segment `in_Liste`, so wird geprüft, ob das zu überprüfende Attribut eine Liste ist und ob diese Liste den spezifizierten Wert enthält.
- Zusätzlich kann noch das Attribut `Blockieren` vorhanden sein. Es enthält eine Liste von Merkmalen, die nicht im Stamm vorhanden sein dürfen. Jedes zu blockierende Merkmal ist eine Liste, deren erstes Element der Name des im Stamm zu überprüfenden Attributs ist. Das zweite Element ist der Wert, mit dem der Wert des im Stamm zu überprüfenden Attributs nicht kongruieren darf.
- Bei Verben ist noch zusätzlich das Attribut `Konjugation` enthalten. Es ist eine Liste von möglichen Ergebnissen einer erfolgreichen Konkatenation. Jedes mögliche Ergebnis ist ein Verbund, der die Attribute `Subjekt` und `Modus` enthält.
- Bei Substantiven und Namen ist noch zusätzlich das Attribut `KasusNumerus` vorhanden. Der Wert dieses Attributs ist die Kasus-Numerus-Kombination, für die das flektierte Substantiv stehen kann.
- Bei Adjektiven ist noch das Attribut `Endung` enthalten, das angibt, welches die Endung des flektierten Adjektivs ist. Mögliche Werte sind `e, er, en, es` und `em`.

5.2.6 Lexikoneinträge von Suffixen

In der Datei `lex/suffixe.lex` sind verschiedene Suffixe eingetragen (siehe auch Anhang E.6). Es sind: das Komparativsuffix `"er"`, die Superlativsuffixe, alle Diminutivsuffixe, die produktiven Derivationssuffixe und -suffixoide, alle Fugen, die Präteritalinfixe sowie die Partizip-II Endungen. Folgende Attribute müssen vorhanden sein:

- Das Attribut `LemOfl` enthält die Oberfläche des Suffixes als Zeichenkette.
- Das Attribut `Sx` ist das Erkennungsmerkmal aller Suffixe. Es ist eine Liste und enthält alle möglichen Kombinatoriken des Suffixes (s.u.).

Jede mögliche Kombinatorik im Attribut `Sx` ist ein Verbund aus folgenden Attributen:

- Das Attribut `Suche` ist eine Liste von im Stamm zu überprüfenden Merkmalen. Jedes zu überprüfende Merkmal ist wiederum eine Liste; deren erstes Element ist das Attribut, das im Stamm enthalten sein muß; das zweite Element ist ein Zugriffsmarker und das dritte ist der Wert, mit dem der Wert des im Stamm zu überprüfenden Attributs verglichen werden soll. Ist der Zugriffsmarker das Segment `kongr`, so wird auf Kongruenz geprüft; ist der Zugriffsmaker das Segment `in_Liste`, so wird überprüft, ob das zu überprüfende Attribut eine Liste ist und eines deren Listelemente der Vergleichswert ist; ist der Zugriffsmarker das Segment `ist_in`, so muß der Vergleichswert (das dritte Element des zu überprüfenden Merkmals) eine Liste sein und der Wert des im Stamm zu überprüfenden Attributs muß in dieser Liste enthalten sein.
- Das Attribut `Lösche` ist eine Liste von Attributnamen, die bei erfolgreicher Konkatenation nicht aus dem Stamm ins Ergebnis übernommen werden sollen.
- Das Attribut `Blockieren` ist eine Liste von Merkmalen, die im Stamm nicht vorhanden sein dürfen. Syntax und Semantik dieser Liste sind identisch mit Syntax und Semantik des gleichnamigen Attributs bei Flexionsendungen (vgl. 5.2.5).
- Das Attribut `Ergebnis` ist ein Verbund, der alle Attribute enhält, die bei erfolgreicher Konkatenation zusätzlich ins Ergebnis geschrieben werden sollen.
- Zusätzlich kann noch ein Attribut `SxTyp` vorhanden sein, der den Typ der Suffigierung angibt.
- Desweiteren kann zusätzlich noch das Attribut `Gewicht` vorhanden sein. Gültige Werte sind Fließkommazahlen größer als Null und kleiner als Eins. Das Attribut ist Teil des Gewichtungsmechanismus.

5.2.7 Das Partikellexikon

Das Partikellexikon befindet sich in der Datei `lex/partikel.lex` (siehe auch Anhang E.7) und enthält alle morphologisch unveränderlichen Partikeln wie

Adverbiale, Pronomina, Konjunktionen sowie Präfixe und Präfixoide. Folgende Attribute sind bei Partikeleinträgen vorhanden:

- Das Attribut `LemOFl` enthält die Oberfläche der Partikel als Zeichenkette.
- Das Attribut `Klasse` enthält die Wortklasse der Partikel. Es kann bei Partikeln die Werte `Adverb` (Adverbial), `Kon` (Konjunktion), `Pron` (Pronomen), `Präfixoid` oder `Präfix` haben. Bei Partikeln, die als Präfixe verwendet werden können, kann das Attribut `Klasse` entfallen, wenn das Attribut `Px` vorhanden ist (s.u.).
- Das optionale Attribut `Ergebnis` enthält die syntaktische Funktion der Partikel. Dieses kann je nach Verwendung der Partikel unterschiedlichstes Aussehen haben.
- Das optionale Attribut `Struktur` enthält die Valenzen, die die Partikel in einem Syntagma eröffnet. Es enthält eine Liste von alternativ möglichen Valenzrahmen. Jeder Valenzrahmen ist ein Verbund.
- Das optionale Attribut `Px` ist eine Markierung, an der erkannt wird, daß ein Allomorph auch als Präfix verwendet werden kann. Es ist ein Verbund, der zumindest das Attribut `OFl` enthalten muß. Das Attribut `Px.OFl` hat den Wert `ja`, wenn die Entscheidung über die Wohlgeformtheit der Präfigierung auch über die Präfixoberfläche und einen Eintrag in einer Präfixliste im Stamm geschehen soll. Soll nur aufgrund rein kategorialer Information über die Konkatenation entschieden werden, so muß hier der Wert `nein` stehen. Weiterhin kann der Verbund `Px` noch das Attribut `Suche` enthalten, das eine Liste ist. Das erste Element der Liste enthält den Namen eines im Stamm zu prüfenden Attributs, das zweite ist ein Zugriffsmarker (wie bei Flexionsendungen) und das dritte ist der Wert, mit dem der Wert des Attributs im Stamm übereinstimmen soll. Das Attribut `Suche` muß vorhanden sein, wenn das Attribut `Px.OFl` den Wert nein hat. In diesem Fall muß auch das Attribut `Px.Ergebnis` vorhanden sein; es ist ein Verbund, der alle Attribut-Werte-Paare enthält, die in das Ergebnis der erfolgreichen Konkatenation geschrieben werden sollen.

5.2.8 Lexikoneinträge von Eigennamen

Alle Eigennamen befinden sich in der Datei `lex/namen.lex` (siehe auch E.4). Folgende Attribute sind bei Nameneinträgen vorhanden:

- Das Attribut `LemOFl` enthält den Namen als Zeichenkette.
- Das Attribut `Klasse` hat bei allen Eigennamen den Wert `Name`.
- Zusätzlich kann noch das Attribut `NamKl` vorhanden sein, das angibt, um welche Art von Namen es sich handelt. Mögliche Werte sind `VorName`, `NachName`, `OrtsName`, `StaatsName` und `TopolName` (topologischer Name, Landschaftsbezeichnung).
- Wenn das Attribut `NamKl` vorhanden ist und den Wert `VorName` hat, so muß noch ein Attribut `Genus` vorhanden sein. Mögliche Werte sind hier `Maskulinum` und `Neutrum`.

5.2.9 Lexikoneinträge von Zahlwörtern

Alle Zahlwörter befinden sich in der Datei `lex/num.lex` (siehe auch E.9). Folgende Attribute müssen vorhanden sein:

- Das Attribut `LemOFl` enthält die Oberfläche des Zahlworts als Zeichenkette.
- Das Attribut `Klasse` hat bei allen Zahlwörtern den Wert `Num`.

5.2.10 Lexikoneinträge von Akronymen und Abkürzungen

Alle Abkürzungen und Akronyme befinden sich in der Datei `lex/abk.lex` (siehe auch E.10). Folgende Attribute müssen vorhanden sein:

- Das Attribut `LemOFl` enthält die Abkürzung als Zeichenkette.
- Das Attribut `Klasse` hat bei allen Abkürzungen und Akronymen den Wert `Abk`.

5.2.11 Lexikoneinträge von Interpunktionszeichen

Alle Interpunktionszeichen befinden sich in der Datei `lex/interpunktion.lex` (siehe auch Anhang E.8). Folgende Attribute müssen vorhanden sein:

- Das Attribut `LemOFl` enthält das Interpunktionszeichen als Zeichenkette.
- Das Attribut `Klasse` enthält bei allen Interpunktionszeichen den Wert `Interpunkt`.
- Das Attribut `Ergebnis` enthält die Art des Interpunktionszeichens.

5.2.12 Lexikoneinträge irregulärer Paradigmen

Alle irregulären Paradigmen befinden sich in der Datei `lex/paradigma.lex` (siehe auch Anhang E.11. Folgende Attribute müssen vorhanden sein:

- Das Attribut `LemOFl` enthält die Grundform des Paradigmas als Zeichenkette.
- Das Attribut `Paradigma` enthält die Wortklasse des Paradigmas.
- Das Attribut `Wortform` ist eine Liste, die Verbunde enthält. Jeder Verbund steht für eine zu erzeugende Wortform; der Inhalt jedes Verbundes hängt von der Wortklasse des Paradigmas ab.

5.3 Die Allomorphregeln

Die Allomorphregeln dienen zwei Zwecken: der Erzeugung neuer Allomorphe und der Erzeugung neuer, im Grundformlexikon nicht vorhandener Information über alle Allomorphe. Die Regeldatei hat den Namen `dmm.all` und befindet sich im Verzeichnis `rules` (siehe auch Anhang B).

5.3.1 Die Startregel

Eine Allomorph-Regeldatei muß immer mit dem Schlüsselwort `initial-rules` beginnen; dadurch werden alle Regeln, die auf den Lexikoneintrag angewandt werden sollen, spezifiziert. In dieser Grammatik laufen alle Lexikoneinträge durch die Regel `Startregel`. Die Regel prüft anhand der kategorialen Information des Lexikoneintrags, um was für einen Eintrag es sich handelt.
Ist der Eintrag ein Verbeintrag, so wird noch unterschieden zwischen ablautenden und nichtablautenden Verben. Die ablautenden Verben werden anhand des Attributs `AK` erkannt und laufen dann in die Regel `Verb-Ablaut`; die Verben, bei denen das Attribut `"AK"` nicht enthalten ist, laufen in die Regel `Verb-Schwach`.
Ist der Eintrag ein Substantiv oder ein obligatorisch zu präfigierendes Substantiv (z.B. `"nahme"` wie in `"auf/nahme"`), so wird die Regel `Substantiv-Start` auf den Eintrag angewandt. Adjektive laufen in die Regel `Adjektiv-Start`, Namen in die Regel `Namen` und alle anderen Einträge durch die Regel `Default`.

5.3.2 Die Behandlung der Verben

Wie unter 4.2.1 beschrieben, gliedern sich die Verben des Deutschen hinsichtlich ihres Allomorphieverhaltens in vier Gruppen: reguläre, semi-reguläre, semi-irreguläre und irreguläre Verben.
Die Allomorphie der irregulären Verben ist nicht regelhaft und erfordert es, daß alle Allomorphe eines irregulären Verbs bereits im Grundformlexikon eingetragen werden. Die Erzeugung der Allomorphe geschieht durch die Regel `Paradigma`.
Die restlichen drei Verbklassen bilden ihre Allomorphe nach formalisierbaren Regeln. Die meisten der hierbei zu behandelnden Phänomene müssen im Prinzip bei allen drei Verbklassen behandelt werden; dementsprechend laufen alle Verballomorphe weitgehend durch den selben Regelstrang. Unterschiede existieren nur bezüglich der Frage, ob die Allomorphie eines Verbs aus dessen Grundformoberfläche geschlossen werden kann (wie bei den regulären und den semi-regulären) oder ob noch zusätzliche Information erforderlich ist (wie bei den semi-irregulären). Dem wird dadurch Rechnung getragen, daß die Regelstränge mit zwei unterschiedlichen Anfangsregeln beginnen; bei den regulären und semi-regulären ist dies die Regel `Verb-Regulär`, bei den semi-irregulären ist es die Regel `Verb-Ablaut`
Desweiteren muß noch zusätzliche Information erzeugt werden. Dabei werden folgende Attribute in erster Linie zur Beschreibung der Flexionskombinatorik benötigt:

- Das Attribut `FlexTyp` spezifiziert, ob ein Stamm stark oder schwach flektiert. Mögliche Werte sind `stark` und `schwach`.
- Das Attribut `SF` bezeichnet die Stammformen, für die der Stamm stehen kann. Mögliche Werte sind `SF_1, SF_2, SF_3, SF_4, SF_5, SF_el, SF_l, SF_er, SF_r` sowie Multisegment-Kombinationen aus diesen (vgl Datei `rules/dmm.seg`, A).
- Das Attribut `PhonEnd` enthält Information über die phonetische Endung des Stammes, die Einfluß auf die Flexionskombinatorik des Stammes hat. Gültige Werte sind `Dental, Sibilant` und `unmarkiert`.

- Das Attribut `NullFlex` gibt an, ob ein Verbstamm bereits ohne Konkatenation einer Flexionsendung eine fertige Wortform darstellt. Dieses Attribut ist eine Liste; mögliche Listenelemente sind `Imperativ`, `Präsens` und `Präteritum`.
- Das Attribut `Tempus` kennzeichnet das Tempus, für das der Stamm steht. Mögliche Werte sind `Präsens` und `Präteritum`.
- Das Attribut `Basis` enthält die zugehörige Grundform des Allomorphs. Gültige Werte sind Zeichenketten.

Gelöscht werden folgende Attribute:

- Die Attribute `PII-Auxil` und `PII-Ge` bei den Stämmen, die nicht zur Bildung des Partizip-II verwendet werden.
- Das Attribut `LemmaOberfläche` wird intern zu `T_OFl` (temporäre Oberfläche); am Ende der Allomorphregeln wird auch dieses gelöscht.

Die Regel `Verb-Ablaut`

Die Regel `Verb-Ablaut` erzeugt die Allomorphe der ablautenden (semiirregulären) Verben.
Dazu wird zuerst der ablautende Vokal identifiziert. Er muß im Grundformlexikon dadurch gekennzeichnet werden, daß er durch geschweifte Klammern eingegrenzt wird.
Die Ablautreihe ist im Attribut `AK` des Grundformeintrags vermerkt. Sie wird mit der Tabelle der Ablautreihen aus der Datei `rules/i_dmm.ablaut` verglichen, in der alle Ablautreihen beschrieben sind. Für jeden Vokal in der Ablautreihe wird nun ein eigener Pfad erzeugt. In jedem dieser Pfade wird der markierte Ablaut durch den Vokal aus der Ablautreihe ersetzt. In den Ablautreihen ist auch die Länge aller Vokale beschrieben, so daß eventuell wegen Wechsel der Vokallänge notwendige orthographische Korrekturen später noch erfolgen können.
Tritt zusätzlich noch Konsonantenwechsel auf, so müssen die wechselnden Konsonanten in der Oberfläche der Grundform mit in die geschweiften Klammern eingeschlossen werden. Auch die Beschreibung der Ablaute muß dann die Konsonanten mit berücksichtigen.
Jede Stammform läuft nun durch einen eigenen Pfad, wo ihr in dieser Regel noch die Attribute `FlexTyp`, `SF` und `Basis` zugeordnet werden.
Da die gemischt stark-schwachen Verben auch von dieser Regel bearbeitet werden, deren Präteritalstämme jedoch schwach flektieren, wird der Verbflexionstyp `FlexTyp` dieser Stämme auf schwach gesetzt. Dann wird bei allen Stämmen, die nicht als Partizip-II-Stamm auftreten können (erkennbar am nicht vorhandenen dritten Listenelement `PII` der Beschreibung des abgelauteten Vokals), die Partizip-II-Informationen gelöscht.
Zuletzt werden den Stämmen im Attribut `Tempus` die Tempora zugeordnet, für die sie stehen können. Dann wird die Ausgabe dem Regelmechanismus übergeben; als Nachfolgeregel wird die Regel `Verb-Orthographie` spezifiziert.

Die Regel `Verb-Regulär`

Die Regel `Verb-Regulär` behandelt die regulären Verben und erzeugt bei semiregulären Verben zwei Pfade mit unterschiedlichen Allomorphen.
Die Regel unterscheidet zwei Fälle: diejenigen Verben, deren Oberfläche auf `"e[lr]n"` endet (also die semi-regulären), und diejenigen, die auf `"en"` enden (also die regulären).
Im ersten Fall wird durch die Oberflächenabfrage gleichzeitig in `"ern"`- und `"eln"`-Verben unterteilt, indem der unterscheidende Konsonant (entweder `"r"` oder `"l"`) in der Variable `$Konsonant` abgelegt wird. Außerdem wird hier die Infinitivendung `"n"` abgetrennt. Abhängig von `$Konsonant` werden dann die Variablen `$SF1` und `$SF2` mit den Stammformen der beiden zu erzeugenden Allomorphe belegt. Dann werden die beiden Ausgaben definiert, wobei die oben belegten Stammformen dem Attribut `SF` zugeordnet werden und zur Erzeugung der korrekten Oberflächen die Variable `$Konsonant` benutzt wird. Beide Allomorphe erhalten den Verbflexionstyp `schwach`.
Im zweiten Fall, also bei den regulären Verben, wird nur die Infinitivendung abgetrennt und dem Allomorph der Verbflexionstyp `schwach` und die Stammform `SF_12` zugeordnet.

Die Regel `Verb-Orthographie`

Diese Regel nimmt Korrekturen vor, die Folge der nicht phonematischen Beziehung zwischen Laut und Zeichen der deutschen Orthographie sind. Dabei werden zum einen Doppelkonsonanten nach langen/kurzen Vokalen eliminiert/erzeugt, zum anderen ss/ß je nach Bedarf ineinander umgewandelt.
Bei kurzem Stammvokal gefolgt von `"ss"` müssen bei Stämmen, die die erste Stammform darstellen können, zwei Allomorphe erzeugt werden, eines das auf `"ß"` und eines das auf `"ss"` endet, da in diesem Fall sowohl auf `"e"` als auch auf `"t"` beginnende Flexionsendungen angehängt werden können. Um Oberflächenabfragen in den Kombinationsregeln zu vermeiden, wird die Information über die orthographische Endung im Attribut `Orthographie` erzeugt. Bei den anderen Stämmen, die kurzen Stammvokal gefolgt von `"ss"` aufweisen, können nur auf `"t"` beginnende Flexionsendungen konkateniert werden, so daß nur ein auf `"ß"` endendes Allomorph erzeugt werden muß.
Der Rest der Regel bezieht sich nur auf die Präteritalstämme der ablautenden Verben; bei diesen kann die Länge des Stammvokals sich gegenüber der Grundform geändert haben, was orthographisch auch Auswirkungen auf die folgenden Konsonanten haben kann.
Folgt einem kurzen Stammvokal ein `"f"` oder `"t"`, so war der Stammvokal der Grundform aufgrund der geltenden Rechtschreibkonvention lang. In diesem Fall wird der Konsonant am Ende verdoppelt.
Folgt einem langen Stammvokal ein Doppelkonsonant, so war der Stammvokal der Grundform kurz; der Doppelkonsnonant wird zu einem einfachen reduziert.
Folgt einem kurzen Stammvokal ein `"ß"`, so war der Stammvokal der Grundform lang; in diesem Fall müssen zwei Allomorphe (auf `"ß"` und `"ss"`) erzeugt werden, da sowohl mit `"e"` als auch mit `"t"` beginnende Flexionsendungen konkateniert werden können.
Folgt einem langen Stammvokal ein `"ss"`, so muß dieses in jedem Fall zu `"ß"` werden, da nach langen Vokalen grundsätzlich `"ß"` steht.

Ansonsten müssen nur eventuell vorhandene Längenzeichen gelöscht werden; diese Stämme sind in orthographischer Hinsicht unproblematisch.
Alle Stämme laufen dann weiter in die Regel `Verb-Phon-Info`.

Die Regel `Verb-Phon-Info`

Die Regel `Verb-Phon-Info` erzeugt Information über die Endung des Stammes, da diese Einfluß auf die Kombiantorik des Stammes hat (vgl. 4.2.1). Diese Information wird im Attribut `PhonEnd` abgelegt.
Stämme, die auf einen dentalen Plosiv enden oder auf einen Konsonantencluster, der aus aus einem nicht-liquiden Konsonanten (also nicht /l/ oder /r/) und einem nasalen Konsonanten (also /m/ oder /n/) besteht (nasale Doppelkonsonanz), werden mit `[PhonEnd: Dental]` gekennzeichnet. Stämme, die auf Sibilanten enden, werden mit `[PhonEnd: Sibilant]` gekennzeichnet. Alle anderen Stämme werden mit `[PhonEnd: unmarkiert]` gekennzeichnet.
Alle Stämme laufen weiter in die Regel `Verb-NullFlex`.

Die Regel `Verb-NullFlex`

In drei Fällen kann ein Verbstamm bereits ein fertig konjugierte Form sein, wobei die beiden ersten Möglichkeiten sich nicht gegenseitig ausschließen:

- Bei ablautenden Verben, bei denen e/i-Wechsel auftritt, ist die zweite Stammform ein fertiger Imperativ. Eine solche Stammform liegt vor, wenn das Verb einer bestimmten Ablautreihe angehört und das Attribut `SF` den Wert `SF_2` hat. Sie wird von der Endregel der Kombinationsregeln dadurch erkannt, daß das Segment `Imperativ` in der Liste `NullFlex` vorhanden ist, das in der Regel `Verb-NullFlex` erzeugt wird.

- Bei ablautenden Verben, bei denen die zweite Stammform eine umgelautete Variante der ersten Stammform ist oder bei der e/i-Wechsel auftritt, und wenn der Stamm auf einen Dental endet, ist die zweite Stammform bereits die 3. Person Singular Indikativ Präsens. Eine solche Stammform liegt vor, wenn das Verb einer bestimmten Ablautreihe angehört, das Attribut `SF` den Wert `SF_2` und das Attribut `PhonEnd` den Wert `Dental` hat. Ist dies der Fall, so wird das Segment `Präsens` in die Liste `NullFlex` aufgenommen.

- Stämme, die die dritte Stammform darstellen, sind bereits die 1. und 3. Person Singular Indikativ Präteritum. Ist dies der Fall, so wird das Segment `Präteritum` in die Liste `NullFlex` aufgenommen.

Tritt keines dieser Phänomene auf so wird die Liste `NullFlex` nicht erzeugt. Außerdem wird bei allen Stämmen das Attribut `AK` entfernt, da alle für die Kombinatorik notwendigen Informationen bereits erzeugt wurden und das Attribut nicht mehr benötigt wird.
Die Behandlung der Verballomorphe wird mit dieser Regel abgeschlossen; alle jetzt noch existenten Pfade laufen in die Regel `Erzeuge-Allo`.

5.3.3 Die Behandlung der Substantive

Bezüglich der Allomorphie müssen bei der Behandlung der Substantive zwei Phänomene behndelt werden: die Pluralumlautung und die Diminutivumlautung. Während Diminutivumlautung immer bei umlautfähigem Vokal eintritt, lauten nicht alle umlautfähigen Substantive auch im Plural um. Die Pluralumlautung muß daher im Lexikon erzeugt werden.
Neu erzeugt bzw. manipuliert werden müssen folgende Attribute:

- Das neu erzeugte Attribut `Diminutiv` gibt an, ob ein Stamm zur Bildung des Diminutiv verwendet wird. Mögliche Werte sind `ja` und `nein`; in den Allomorphregeln kann das Attribut intern auch den Wert `Plural` annehmen.

- Das Attribut `SgDekl` enthält das Singularflexionsverhalten des Substantivs. Der Wert dieses Attributs kommt aus dem Grundformlexikon; bei Allomorphen, die nicht zur Bildung des Singularparadigmas verwendet werden, wird der Wert dieses Attributs auf `nein` gesetzt.

- Das Attribut `PlDekl` enthält das Pluralflexionsverhalten des Substantivs. Der Wert dieses Attributs kommt aus dem Grundformlexikon; bei Allomorphen, die nicht zur Bildung des Pluralparadigmas verwendet werden, wird der Wert dieses Attributs auf `nein` gesetzt.

- Das Attribut `KasusNumerus` enthält die Kombinationen aus Kasus und Numerus, für die das Allomorph stehen kann, ohne daß eine Flexionsendung konkateniert wurde.

- Das Attribut `Basis` enthält die zugehörige Grundform des Allomorphs. Gültige Werte sind Zeichenketten.

Die Regel `Substantiv-Start`

Alle Substantive werden von der Startregel an die Regel `Substantiv-Start` übergeben. Dabei werden eventuelle Plural- und/oder Diminutivallomorphe erzeugt. Relevant sind die aus dem Lexikoneintrag stammenden Attribute `SgDekl` (Singulardeklinationsverhalten) und `PlDekl` (Pluraldeklinationsverhalten) und das von der Regel zu erzeugende Attribut `Diminutiv`. Kann ein Allomorph nicht für die Singularflexion herangezogen werden, so wird das Attribut `SgDekl` auf `nein` gesetzt; kann ein Allomorph nicht für die Pluralflexion herangezogen werden, so wird das Attribut `PlDekl` auf `nein` gesetzt; ansonsten bleiben die beiden Attribute so, wie sie im Lexikoneintrag spezifiziert werden. Pluralallomorphe, die auch zur Bildung des Diminutivs verwendet werden, werden mit `[Diminutiv: Plural]` gekennzeichnet; alle anderen Diminutivallomorphe werden mit `[Diminutiv: ja]` gekennzeichnet. Es sind dabei vier Fälle zu unterscheiden:

- Umlautung von heimischen Substantiven wird im Lexikon dadurch gekennzeichnet, daß der umlautende Stammvokal in der Lemmaoberfläche durch geschweifte Klammern gekennzeichnet wird. Bei diesen Substantiven muß ein umgelautetes Plural- und Diminutivallomorph erzeugt werden. Der umlautende Vokal wird durch Zerlegung der Lemmaoberfläche identifiziert und in Abhängigkeit von diesem umlautenden Vokal das zweite Allomorph erzeugt.

- Bei Fremdwörtern griechisch-lateinisch-italienischen Ursprungs, die sich ihre fremdsprachliche Pluralbildung bewahrt haben (wie z.B. *Modus – Modi*) oder die deutsche Ersatzendungen haben (wie z.B. *Arena - Arenen*), ist die ganze Endung durch geschweifte Klammern gekennzeichnet. Diese wird durch Zerlegung der Lemmaoberfläche identifiziert und dann zwei Allomorphe erzeugt, eines mit der Singularendung und eines mit der Pluralendung. Diese Substantive müssen im Lexikon mit `[PlDekl: Pl_0]` gekennzeichnet sein.

- Bei Substantiven ohne gekennzeichneten Pluralumlaut, die jedoch einen umlautfähigen Stammvokal besitzen, muß ein umgelautetes Allomorph erzeugt werden, das nur zur Diminutivbildung verwendet wird. Der umlautfähige Stammvokal wird durch Zerlegung der Lemmaoberfläche identifiziert und das umgelautete Diminutivallomorph erzeugt.

- Bei allen anderen Substantiven kann der Lexikoneintrag zur Singularflexion, zur Pluralflexion und zur Bildung des Diminutiv verwendet werden.

Alle Einträge laufen weiter in die Regel `Verkürze-Substantiv-e-en`.

Die Regel `Verkürze-Substantiv-e-en`

Der Diminutiv von Substantiven, die auf `"e"` oder `"en"` enden, wird durch Allomorphe gebildet, die um dieses `"e"` oder `"en"` verkürzt sind.
Handelt es sich um einen Substantivstamm, der sowohl für den Plural als auch den Diminutiv verwendet werden kann (erkennbar an `[Diminutiv: Plural]`, so muß dieses verkürzte Allomorph zusätzlich erzeugt werden. Das Pluralallomorph wird durch `[Diminutiv: nein]` als nicht diminutivfähig gekennzeichnet, das Diminutivallomorph als nicht flektierbar (durch `[SgDekl: nein, PlDekl: nein]`).
Handelt es sich um einen reinen Diminutivstamm, so genügt es, diesen Stamm zu verkürzen und kennzuzeichnen durch `[Diminutiv: ja, SgDekl: nein, PlDekl: nein]`.
Bei Stämmen, die nicht auf `"e"` oder `"en"` enden, wird, wenn gegeben, `[Diminutiv: Plural]` durch `[Diminutiv: ja]` ersetzt, da die Information, ob ein Stamm gleichzeitig Plural- und Diminutivallomorph ist, nicht mehr benötigt wird. Ansonsten bleiben diese Stämme unverändert.
Alle Stämme laufen weiter in die Regel `Substantiv-Orthographie`.

Die Regel `Substantiv-Orthographie`

Bei Singularallomorphen nichtfemininer Substantive, die auf kurzen Vokal gefolgt von `"ß"` enden, muß ein zweites, auf `"ss"` endendes Allomorph erzeugt werden, da Flexionsendungen, die mit `"e"` beginnen, konkateniert werden können. Die Vokalkürze muß in diesem Fall in der Oberfläche des Grundformeintrags durch ein Apostroph nach dem Vokal gekennzeichnet sein. Feminine Substantive sind davon nicht betroffen, da sie in Singular immer unflektiert bleiben.
Bei Pluralallomorphen aller Genera muß finales `"ß"` nach kurzen Vokalen immer zu `"ss"` werden, da in jedem Fall eine Flexionsendung konkateniert wird.

Alle anderen Stämme bleiben unverändert. Alle Stämme laufen weiter in die Regel `Substantiv-Kasus`.

Die Regel `Substantiv-Kasus`

Einige Stämme können bereits ohne Flexionsendung eine fertig flektierte Wortform darstellen. Da Kasus und Numerus diesen Wortformen nicht durch Konkatenation zugeordnet wird, muß diese Information bereits an den Allomorphen gekennzeichnet werden. Prinzipiell ausgenommen hiervon sind die Allomorphe, die nur zur Diminutivbildung verwendet werden.
Kasus und Numerus des Allomorphs wird über die Tabelle in der Datei `rules/i_dmm.kasus` zugeordnet. In der Datei stehen alle möglichen Kombinationen von Singular- und Pluralflexion mit den jeweiligen Kasus-Numerus-Kombinationen. Diese Kasus-Numerus-Information wird im Allomorph in dem Attribut `KasusNumerus` vermerkt.
Die Bearbeitung der Substantive ist dann abgeschlossen; alle Allomorphe laufen in die Regel `Erzeuge-Allo`.

5.3.4 Die Behandlung der Adjektive

Bei Adjektiven ist hinsichtlich der Allomorphie nur das Phänomen der Komparativumlautung zu beachten. Die Umlautung muß im Lexikon markiert sein, da nicht alle umlautfähigen Adjektive umlauten.
Neu erzeugt bzw. manipuliert werden müssen folgende Attribute:

- Das Attribut `AdjFlex` enthält das Flexionsverhalten des Adjektivs. Der Wert dieses Attributs kommt aus dem Lexikon und kann die Werte `AdjFlex_a` (adverbialer Gebrauch möglich), `AdjFlex_d` (deklinierbar), `AdjFlex_k` (Komparativbildung möglich) und `AdjFlex_s` (Superlativbildung möglich) bzw. aus diesen zusammengesetzte Multisegmente. Bei Allomorphen, bei denen eine oder mehrere der genannten Verwendungen nicht möglich sind, muß diese Verwendungsmöglichkeit gelöscht werden.

- Das Attribut `Komparation` muß bei den Allomorphen, die zur Bildung des Positivs verwendet werden können, vorhanden sein und den Wert `Positiv` haben. Bei den anderen Allomorphen wird dieses Attribut erst in den Kombinationsregeln durch Konkatenation mit dem Komparativ- oder Superlativsuffix erzeugt.

- Das Attribut `PhonEnd` enthält Information darüber, ob eventuell bei Superlativsuffix e-Einfügung oder s-Tilgung auftritt. Mögliche Werte sind `unmarkiert, e-Einfügung` und `s-Tilgung`.

- Das Attribut `HeitSx` enthält Information über die Variante des deadjektivischen Derivationssuffixes *-heit*, das an den Stamm angeheftet wird. Das Attribut ist eine Liste; mögliche Werte der Listenelemente sind `heit, keit` und `igkeit`.

- Das Attribut `Basis` enthält die zugehörige Grundform des Allomorphs. Gültige Werte sind Zeichenketten.

Die Regel `Adjektiv-Start`

Die Regel unterscheidet zwischen umzulautenden Einträgen und nicht umzulautenden Einträgen. Die umzulautenden Einträge werden daran erkennt, daß der umzulautende Stammvokal in der Lemmaoberfläche in geschweifte Klammern eingeschlossen ist.
Bei umlautendem Adjektiv wird der Stammvokal identifiziert und in Abhängigkeit von diesem werden zwei Allomorphe erzeugt, eines ohne und eines mit Umlaut. Dem nicht umgelauteten Positivallomorph wird `[Komparation: Positiv]` zugeordnet. Das Attribut `AdjFlex` erhält beim Positivallomorph den Wert `AdjFlex_ad` (adverbialer Gebrauch möglich und deklinierbar), bei anderen Allomorph den Wert `AdjFlex_ks` (Komparativ- und Superlativbildung möglich).
Bei nicht umlautenden Adjektiven wird dem einzigen Allomorph nur `[Komparation: Positiv]` zugeordnet.
Alle Allomorphe laufen weiter in die Regel `Adjektiv-Endung`.

Die Regel `Adjektiv-Endung`

Bei einigen Adjektiven existieren neben eventueller Umlautung noch weitere Allomorphiephänomene. Die Regel `Adjektiv-Endung` erzeugt die zusätzlichen Allomorphe in folgenden Fällen:

- Bei Adjektiven, deren Grundform auf `"e[nr]"` endet, existiert ein weiteres Allomorph, bei dem das Ende des Stammes um `"e"` verkürzt ist. Während das normale Allomorph sowohl adverbial, dekliniert, beim Komparativ und beim Superlativ verwendet wird, kann das verkürzte Allomorph nur dekliniert und im Komparativ verwendet werden. Diese Regel gilt nicht für Adjektive griechisch-lateinisch-italienischen Ursprungs die auf der Endsilbe unbetont sind, nicht für Stämme, bei denen das `"e"` Teil eines Vokals ist (wie in *schier, leer*) und bei den Adjektiven *schwer* und *quer*. Bei diesen existiert kein verkürzter Stamm.

- Bei Adjektiven, deren Grundform auf `"el"` endet, existiert ein weiteres Allomorph, bei dem das Ende des Stammes um `"e"` verkürzt ist. Die Allomorphe sind komplementär verteilt; das verkürzte Allomorph wird bei der Deklination und für die Bildung des Komparativ verwendet, während das andere Allomorph bei adverbialem Gebrauch und bei der Bildung des Superlativ verwendet wird.

- Bei Adjektiven, deren Grundform auf kurzen Vokal gefolgt von `"ß"` endet, muß ein weiteres Allomorph erzeugt werden, bei dem das `"ß"` durch `"ss"` ersetzt wird. Das auf `"ß"` endende Allomorph wird nur bei adverbialem Gebrauch verwendet; das auf `"ss"` endende Allomorph wird bei der Deklination und zur Bildung von Komparativ und Superlativ verwendet.

- Bei Adjektiven, deren Grundform auf `"e"` endet, existiert ein zweites, um das finale `"e"` verkürztes Allomorph. Das verkürzte Allomorph wird bei adverbialem Gebrauch, bei der Deklination und zur Bildung von Komparativ und Superlativ verwendet, das nicht verkürzte fakultativ bei adverbialem Gebrauch und zur Bildung des Superlativs.

- Alle anderen Stämme verfügen in dieser Hinischt nur über ein Allomorph.

Alle Stämme laufen weiter in die Regel `Adjektiv-Phon-Info`.

Die Regel `Adjektiv-Phon-Info`

Bei einigen Stämmen wird bei der Bildung des Superlativ nicht `"st"`, sondern `"est"`, gelegentlich auch `"t"` verwendet. Die Information darüber wird im Attribut `PhonEnd` markiert. Dabei gilt es folgende Fälle zu unterscheiden:

- Selten wird bei auf Sibilanten endenden Stämmen der Superlativ durch das Suffix `"t"` gebildet (z.B. `"größ/t/e"`). Bei solchen Fällen muß das Attribut `PhonEnd` bereits im Grundformlexikon vorhanden sein und dort den Wert `s-Tilgung` haben. In diesem Fall wird das Allomorph unverändert an die nächste Regel weitergegeben.
- Stämme, die auf einen Sibilanten außer `"sch"` enden oder auf `"haft"`, bilden den Superlativ auf `"est"`. Das Attribut `PhonEnd` erhält in diesem Fall den Wert `e-Einfügung`.
- Endet der Stamm auf einen Dental, auf `"sch"`, auf einen Diphtong oder einen durch `"h"` gelängten Vokal, so kann der Superlativ sowohl mit *-st* als auch mit *-est* bilden. Das Attribut `PhonEnd` erhält in diesem Fall den Wert `unmarkiert_e-Einfügung`, der ein Multi-Segment ist und mit `unmarkiert` und `e-Einfügung` kongruiert.
- In allen anderen Fällen wird der Superlativ mit `"st"` gebildet und das Attribut `PhonEnd` erhält den Wert `unmarkiert`.

Alle Allomorphe laufen weiter in die Regel `Adjektiv-heit`

Die Regel `Adjektiv-heit`

Aus der lautlichen Gestalt des Adjektivstamms kann mit hoher Sicherheit geschlossen werden, mit welcher der *-heit*-Varianten suffigiert werden kann. Dabei können manche Stämme mit mehr als einer Variante suffigiert werden; in diesem Fall muß die Liste `HeitSx` bereits im Grundformlexikon vorhanden sein und mit den richtigen Listenelementen belegt sein. Auch bei Ausnahmen von den im folgenden angegebenen Regeln muß `HeitSx` bereits im Lexikon vorbelegt sein. Die Regel verändert nur Allomorphe, die derivierbar sind.
Das Attribut `HeitSx` wird von der Regel belegt wie folgt:

- Ist das Attribut bereits vorhanden, so wird es nicht verändert.
- Einsilbige Adjektive verwenden die Variante `"heit"`.
- Auf bestimmte Silben endende, zumeist abgeleitete Adjektive sowie Adjektive, die auf Konsonanten und `"er"` enden, verwenden die Variante `"keit"`.
- Adjektive, die auf `"haft"` oder `"los"` enden, verwenden die Variante `"igkeit"`.
- In allen anderen Fällen wird die Variante `"heit"` zugeordnet.

Mit dieser Regel ist die Behandlung der Adjektive abgeschlossen. Alle Allomorphe laufen in die Regel `Erzeuge-Allo`.

5.3.5 Die Behandlung der Eigennamen

Bei der Behandlung von Eigennamen ist in morphologischer Hinsicht nur von Interesse, ob der Name auf einen Sibilanten endet; diese Namen bilden ihren Genitiv durch Suffigierung mit einem Apostroph `"'"`, während alle anderen Namen ihren Genitiv durch Suffigierung mit `"s"` bilden. Diese Information wird im Attribut `PhonEnd` markiert. Bei auf Sibilanten endenden Namen erhält dieses den Wert `Sibilant`, ansonsten den Wert `unmarkiert`. Desweiteren wird noch das Attribut Basis mit der Grundform belegt. Alle Namen laufen dann weiter in die Regel `Erzeuge-Allo`.

5.3.6 Die Behandlung der restlichen Lexikoneinträge

Die restlichen Lexikoneinträge sind entweder morphologisch unveränderbare Morpheme oder Einträge für irreguläre Paradigmen.

Die Regel `Paradigma`

Irreguläre Paradigmen werden von der Regel `Paradigma` bearbeitet. Diese irregulären Paradigmen müssen anstatt des Attributs `Klasse` das Attribut `Paradigma` mit der Wortklasse als Wert aufweisen. Die Einträge enthalten eine Liste `Wortform`, in der alle zu der Grundform gehörenden Wortformen aufgezählt werden. Die Listenelemente (also die einzelnen Wortformen) müssen in ausgabefähigem Format vorliegen (d.h., die Endregel der Kombinationsregeln muß sie korrekt interpretieren können). Attribute und Werte, die allen Wortformen gemein sind, werden nicht in jedes Listenelement eingetragen, sondern auf oberster Ebene in den Lexikoneintrag.
Die Regel erzeugt für jedes Element der Liste `Wortform` einen eigenen Pfad. In diesem Pfad werden die Attribute `Klasse` mit der Wortklasse des Eintrags und `Basis` mit der Grundform belegt. Außerdem werden alle Attribute, die im Lexikoneintrag auf oberster Ebene angegeben sind mit Ausnahme der Attribute `LemOFl, Paradigma` und `Wortform`, in das auszugebende Allomorph übernommen.
Alle Allomorphe laufen in die Regel `Erzeuge-Allo`.

Die Regel `Default`

Alle Lexikoneinträge, die nicht durch eine der bisher behandelten Regeln gelaufen ist, sind morphologisch unveränderbare Morpheme. Es kann sich hierbei zum einen um diverse Partikeln handeln oder um Affixe.
Partikel müssen das Attribut `Klasse` aufweisen; das Attribut muß dabei einen der folgenden Werte haben: `Artikel, Präp, Pron, Adverb, Interpunkt, Präfix, Präfixoid, Abk, Num, VorgPass` oder `Kon`.
Affixe können entweder Präfixe oder Suffixe sein. Bei reihenbildenden Präfixen ist ein Attribut `Px` vorhanden; desweiteren existiert ein Attribut `Px.OFl`, das den Wert `ja` hat. In diesem Fall muß noch ein Attribut `PxOfl` erzeugt werden, das als Wert die Oberfläche des Präfixes hat.
Bei allen Lexikoneinträgen muß noch das Attribut `Basis` erzeugt werden. Ist es bereits im Lexikon vorhanden, so wird es von dort übernommen; ansonsten wird die Lemmaoberfläche `LemOFl` als Grundform eingesetzt.
Alle Allomorphe laufen dann in die Regel `Erzeuge-Allo`.

5.3.7 Die Erzeugung der Allomorphe

Alle bisher erfolgreichen Pfade laufen wieder in der Regel `Erzeuge-Allo` zusammen. Die Regel geht davon aus, daß die vorangegangenen Regeln das Attribut `T_OFl` erzeugt haben; ist dies nicht der Fall, so kann kein Allomorph erzeugt werden und der Allomorphgenerator bricht mit einer Fehlermeldung ab. In diesem Fall steckt ein syntaktischer Fehler im Lexikon und/oder ein logischer Fehler in den Allomorphregeln.
Das Attribut `T_Ofl` wird dem Regelmechanismus als Allomorphoberfläche und das Attribut `Basis` als Grundform übergeben. Als kategoriale Information wird dem Regelmechanismus die gesamte Regeleingabe übergeben, mit Ausnahme der Attribute `LemOFl, T_Ofl` und `Mater` (dieses enthält derzeit ungenutzte Information aus [Mater 1966], vgl. Gutbrod 1995). Zusätzlich wird das Attribut `Allomorph` erzeugt, das die Oberfläche des Allomorphs enthält.
Damit wurden alle Allomorphe erzeugt und im Allomorphlexikon angespeichert.

5.4 Die Kombinationsregeln

Die Kombinationsregeln befinden sich in der Datei `dmm.mor` im Verzeichnis `rules`; siehe auch Anhang C.1.

5.4.1 Vorbemerkung

In der nun folgenden Beschreibung der Kombinationsregeln wird die Behandlung zweier Mechanismen nicht im einzelnen bei jeder Regel erklärt, sondern soll hier umrissen werden. Diese Mechanismen sind der Gewichtungsmechanismus und der Mechanismus zur Dokumentation der Wortbildungsprozesse.
Der Gewichtungsmechanismus soll es ermöglichen, bei ambigen Wortformen aus den Analysen schließen zu können, welche der Analysen „korrekter" sind. Zwar sind alle Analysen im Sinne des Grammatiksystems wohlgeformt; da sich aufgrund lexikalischer Ambiguitäten und Ambiguitäten hinsichtlich der möglichen Segmentierungen dennoch Analysen ergeben, die nicht sinnvoll sind, läßt sich mit Hilfe des Gewichtungsmechanismus zumindest eine Hypothese darüber bilden, welche der ambigen Analysen mit hoher Wahrscheinlichkeit korrekt sind und welche nicht.
Dazu werden in Abhängigkeit vom morphologischen Prozeß und den beteiligten Allomorphen einer Regelanwendung ein Gewicht zwischen 0 und 1 zugeordnet. Dieses Gewicht wird mit dem Gewicht des analysierten Wortanfangs (beim leeren Wortanfang ist dies 1) multipliziert, so daß das Endgewicht immer zwischen 0 und 1 liegt. Das Gewicht einer Wortform ist immer in dem Attribut `Gewicht` angegeben.
Der Mechanismus zur Dokumentation der Wortbildungsprozesse dient dazu, aus dem Analyseresultat einer Wortform die wichtigsten Wortbildungsprozesse ersehen zu können und welche Morpheme (Grundformen) an der Wortform beteiligt sind. Diese Information ist im Attribut `WortStruktur` angegeben; es ist eine Liste, deren Listenelemente Verbunde mit folgenden Attributen sind:

- `Klasse` gibt die Wortklasse des beteiligten Allomorphs an;
- `Basis` ist ein Verweis auf das Morphem (die Grundform), zu der das Allomorph gehört;

- `Allomorph` ist das Allomorph selbst;
- `Typ` ist die Art der Kombination, mit der das Allomorph mit dem analysierten Wortanfang konkateniert wurde.

5.4.2 Der Startzustand

Am Beginn einer Malaga-Kombinationsregeldatei muß der Startzustand der Grammatik spezifiziert werden. Dazu muß zuerst die Kategorie des leeren Wortanfangs hinter dem Schlüsselwort `initial-cat` angegeben werden. Bei dieser Grammatik wird als leerer Wortanfang der leere Verbund angegeben.
Desweiteren muß ein initiales Regelpaket hinter dem Schlüsselwort `initial-rules` angegeben werden; bei dieser Grammatik besteht dieses Regelpaket nur aus der Regel `Startregel`. Alle Pfade beginnen also in der Startregel.

5.4.3 Die Startregel

In der Startregel wird das erste Segment der Eingabe gelesen.
Im allgemeinen wird in dieser Kombinationsregeldatei der analysierte Wortanfang immer in die Variable `$Links` gelesen, wohingegen das zu konkatenierende neue Allomorph immer in die Variable `$Rechts` gelesen wird.
Bevor in der Startregel diese beiden Variablen belegt werden, werden mit dem Schlüsselwort `generate` die Allomorphe angegeben, die zur Generierung mit dem `paradigm`-Befehl benutzt werden sollen. Wird `base` als Argument übergeben, wird zur Generierung jedes Allomorph mit der Grundform des als nächstes gelesenen Allomorphs benutzt.
Nachdem leerer Wortanfang und neues Allomorph gelesen wurden, überprüft die Regel, ob es sich bei dem neuen Allomorph nicht um eine Flexionsendung oder ein Suffix handelt, da keine Wortform mit Flexionsendungen oder Suffixen beginnen kann.
In der Variable `$Ausgabe` wird ein Standard-Verbund für alle Wortformen spezifiziert. Die hier angegebenen Attribute sind in jeder Ausgabe vorhanden.
Dann wird in Abhängigkeit von kategorialer Information des neuen Allomorphs in Nachfolgeregeln verzweigt:

- Lexeme (Substantive, Adjektive und Verben) können entweder flektiert werden (Regel `Flexion`), Teil eines Kompositums sein (Regel `Komposition`), suffigiert werden (Regel `Suffigierung`) oder bereits fertige Wortformen darstellen.
- Zahlen und Zahlwörter können entweder zu komplexen numerischen Ausdrücken zusammengesetzt werden (Regel `Zahl`) oder fertige Wortformen darstellen (Regel `Endregel`).
- Namen können entweder flektiert werden (Regel `Flexion`), suffigiert werden (Regel `Suffigierung`) oder eine fertige Wortform darstellen (Regel `Endregel`).
- Abkürzungen können enweder eine fertige Wortform darstellen (Regel `Endregel`) oder Teil eines Bindestrich-Kompositums sein (Regel `Komposition`).

- Präfixe können Teil einer Präfigierung sein (Regel `Präfigierung`) oder fertige Wortformen darstellen (Regel `Endregel`).
- Alle anderen Allomorphe sind morphologisch unveränderliche Partikel und laufen direkt in die Regel `Endregel`.

5.4.4 Die Flexion

Die Regel `Flexion` bearbeitet die gesamte Flexion mit Flexionssuffix mit Ausnahme des Anhängens des Präteritalinfixes, des Komparationssuffixes und des Superlativsuffixes; diese werden durch die Regel `Suffigierung` bearbeitet. Auch infinite Verbformen werden von anderen Regeln bearbeitet.
In der Regel werden zuerst hinter dem Schlüsselwort `generate` alle Allomorphe angegeben, die bei der Generierung als neues Allomorph konkateniert werden sollen.
Nachdem Wortanfang und neues Wort eingelesen wurden, wird überprüft, ob der Wortanfang nich für die Flexion blockiert ist. Danach wird geprüft, ob das neue Allomorph eine Flexionsendung ist.
Im Folgenden wird für jede mögliche Kombinatorik der Flexionsendung (vgl. 5.2.5) ein eigener Pfad erzeugt. In jedem Pfad befindet sich die Kombinatorik in der Variable `$FlexEnd`.
Nachdem geprüft wurde, ob die Klassen von Stamm und Flexionsendung übereinstimmen, wird eine Standardausgabe in der Variable `$Ausgabe` definiert. Dann wird für jedes in der Kombinatorik `$FlexEnd.Flexion` angegebene Merkmal (vgl. 5.2.5) überprüft, ob das Attribut vorhanden ist. Ist dies der Fall, so wird in Abhängigkeit vom Zugriffsmarker getestet, ob Stamm und Flexionsendung hinsichtlich dieses Merkmals zusammenpassen. Wenn die Überprüfung aller Merkmale erfolgreich ist, werden die im Stamm untersuchten Attribute aus der Ausgabe gelöscht. Danach wird auf ähnliche Weise getestet, ob die Flexionsendung nicht aufgrund der in `$FlexEnd.Blockieren` angegebenen Merkmale blockiert ist.
Alle Pfade, die bis jetzt erfolgreich verlaufen sind, stellen korrekte Flexionsformen dar.
Alle flektierten Wortformen können mit keinen weiteren Allomorphen konkateniert werden und laufen daher in die Regel `Endregel`.

5.4.5 Die Komposition

Die Komposition von Lexemen wird von der Regel `Komposition` bearbeitet. Die Entscheidung, ob eine Komposition zulässig ist, wird aufgrund morphologischer Kriterien getroffen, da in den Grundformlexika keine semantische Information vorhanden ist.
Die Regel prüft zuerst, ob sowohl Wortanfang als auch neues Allomorph potentiell kompositionsfähig sind. Potentiell kompositionsfähig als Erstglied sind:

- Substantive, Adjektive und Verben;
- Substantive und Adjektive, die durch Anhängen von Fugen zwingend komponiert werden müssen.

Potentiell kompositionsfähig als Zweitglied sind:

- Substantive, Adjektive und Verben;
- Präfixe. Ist dies der Fall, so findet die eigentliche Komposition erst in der Regel `Präfigierung` statt, wenn das Präfix an einen Stamm geheftet wird.

Ist das Zweitglied ein Präfix, so wird geprüft, ob die Präfigerung über die Oberfläche des Präfixes gesteuert wird; in diesem Fall enthält das Attribut `Px.PxOFl` im Präfix den Wert `ja`. Andere Präfixe dürfen nicht zur Komposition zugelassen werden. Wenn das Präfix zugelassen wird, werden seine Attribute in die Ausgabe übernommen; der Verbund, der dann der Regel `Präfigierung` übergeben wird, verhält sich also auch wie ein Präfix.
Ist das Zweitglied ein Lexem, so wird anhand der Kompositionsmuster aus der Datei `rules/i_dmm.kompo` geprüft, ob das Erstglied die Kriterien für eine erfolgreiche Komposition erfüllt. Dabei wird wie bei der Flexionsregel Vorhandensein und Inhalt bestimmter Attribute bzw. zu blockierender Attribute überprüft. Dabei gelten folgende Kriterien:

- Substantiv als Zweitglied:
  - Ist das Erstglied ein Substantiv, so darf es kein Allomorph sein, das einzig zur Bildung des Diminutiv verwendet wird. Substantiv-Substantiv-Komposita sind sehr häufig; die Komposition wird daher mit dem Wert 0,9 gewichtet.
  - Ist das Erstglied ein Adjektiv, so muß es deklinierbar sein. Die Adjektiv-Substantiv-Komposition wird mit 0,7 gewichtet.
  - Ist das Erstglied ein Verb, so muß es eine Präsensstammform sein. Verb-Substantiv-Komposita werden mit 0,3 gewichtet.
- Adjektiv als Zweitglied:
  - Ist das Erstglied ein Adjektiv, so muß es deklinierbar sein. Die Adjektiv-Adjektiv-Komposition wird mit 0,5 gewichtet.
  - Ist das Erstglied ein Substantiv, so darf es kein Allomorph sein, das einzig zur Bildung des Diminutiv verwendet wird. Substantiv-Adjektiv-Komposita werden mit 0,2 gewichtet.
- Verb als Zweitglied:
  - Ist das Erstglied ein Adjektiv, so muß es deklinierbar sein. Adjektiv-Verb-Komposita werden mit 0,1 gewichtet.
  - Ist das Erstglied ein Substantiv, so darf es kein Allomorph sein, das einzig zur Bildung des Diminutiv verwendet wird. Substantiv-Verb-Komposita werden mit 0,2 gewichtet.

Sind alle Kriterien hinsichtlich Erst- und Zweitglied erfüllt, so ist die Komposition im Rahmen der Arbeit als wohlgeformt zu betrachten. Je nach Häufigkeit eines Kompositionsmusters wird das Gewicht der Komposition in der Ausgabe manipuliert.
Komposita können entweder weiter komponiert werden (Nachfolgeregel `Komposition`), flektiert werden (Nachfolgeregel `Flexion`), suffigiert werden (Nachfolgeregel `Suffigierung`) oder fertige Wortformen darstellen (Nachfolgeregel `Endregel`).

5.4.6 Die Suffigierung

Alle Suffigierungsprozesse mit Ausnahme der Flexion werden von der Regel `Suffigierung` bearbeitet. Dies beinhaltet die Derivation mit Derivationssuffix, Anhängen von Fugenzeichen, Präteritalinfix und Partizip-II-Endung.
Als Suffixe werden nur Allomorphe zugelassen, die das Attribut `Sx` aufweisen, also im Sinne dieses Regelsystems als Suffixe gelten. Als Stämme werden nur Wortanfänge zugelassen, die entweder Lexeme oder Namen sind.
Alle Suffixeinträge enthalten das Attribut `Sx`; es ist eine Liste und enthält alle möglichen Kombinatoriken des Suffixes. Für jede Kombinatorik wird ein eigener Pfad erzeugt und geprüft, ob alle von der Kombinatorik geforderten Attribute vorhanden sind und die korrekten Werte haben, und ob die Suffigierung nicht aufgrund der im Suffix angegebenen Blockierungsmerkmale unerlaubt ist. Diese Überprüfung erfolgt analog zu dem in der Flexionsregel verwendeten Überprüfungsmechanismus.
Sind alle Überprüfungen erfolgreich verlaufen, so wird die Wortform als wohlgeformt anzusehen.
Suffigierte Wortformen können entweder weiter suffigiert werden (Nachfolgeregel `Suffigierung`), flektiert werden (Nachfolgeregel `Flexion`), komponiert werden (Nachfolgeregel `Komposition`) oder fertige Wortformen darstellen (Nachfolgeregel `Endregel`).

5.4.7 Die Präfigierung

Die Präfigierungsregel bearbeitet reihenbildene Präfigierungen, Präfigierung mit Präfixoid, Präfigierung von geeigneten Verbstämmen mit dem Partizip-II-Präfix `"ge"` und Infigierung von `"zu"` bei präfigierten Verben im Infinitiv mit *zu*.
Die Regel prüft zuerst, ob der Wortanfang ein Präfix ist (erkennbar am Attribut `Px`). Ist das neue Allomorph ebenfalls ein Präfix, so muß es sich dabei um das Partizip-II-Präfix `"ge"` oder den Infinitiv-mit-*zu*-Marker `"zu"` handeln. In diesem Fall muß der Wortanfang ein reihenbildendes Präfix sein (erkennbar am Attribut `Px.OFl` mit dem Wert `ja`), und als Nachfolgeregel muß wieder die Präfigierungsregel angegeben werden, da das komplexe Präfix im nächsten Schritt in jedem Fall mit einem Lexemstamm kombiniert werden muß.
Ist das neue Allomorph ein Lexemstamm, so muß geprüft werden, ob der Wortanfang ein reihenbildendes Präfix ist und dies in der Präfixliste des Stammes vorhanden ist. Ist dies der Fall, so muß noch zusätzlich getestet werden, ob das Präfix im Wortanfang eventuell `"ge"` oder `"zu"` oder ein komplexes Präfix mit `"ge"` oder `"zu"` ist, das noch das Vorhandensein bestimmter, im Präfix spezifizierter kategorialer Information erwartet. Ist dies der Fall, so werden die erwarteten Merkmale im Stamm geprüft. Da ein Lexemstamm konkateniert wurde, werden als Nachfolgeregeln `Komposition`, `Suffigierung`, `Flexion` und `Endregel` angegeben.

5.4.8 Zahlenregeln

Die beiden Zahlenregeln konkatenieren Ziffern und die Trennzeichen Punkt und Komma. Die Regel `Zahl` konkateniert dabei solange Ziffern, bis ein Komma oder ein Punkt eingelesen wird. Wird ein Komma gelesen, so wird dies als Trennzeichen zwischen der 10^0-Stelle und der 10^{-1}-Stelle interpretiert. Punkte können rein optische Trennzeichen sein oder aber (besonders im angelsächsichen Raum und zunehmend auch im deutschen Sprachraum) dieselbe Bedeutung wie das Komma haben. Die Nachkommastellen werden von der Regel `Nachkommazahl` einfach weiter an den Wortanfang konkateniert.

5.4.9 Die Endregel

Die Endregel akzeptiert die Eingabe als wohlgeformt, wenn die konkatenierte Wortform einigen Bedingungen entspricht und die gesamte Eingabe gelesen wurde.
Während bei Wortformen, die flektiert oder suffigiert wurden, in der Regel nur kosmetische Veränderungen der kategorialen Information vorgenommen werden müssen und ansonsten davon ausgegangen werden kann, daß die Wortformen wohlgeformt sind, muß bei unsuffigierten Stämmen erst noch geprüft werden, ob der Stamm auch ohne Flexionssuffix eine fertige Wortform darstellen kann. Dies geschieht in Abhängigkeit von der kategorialen Beschreibung des Stammes. Kann ein Stamm auch ohne Flexionsendung eine Wortform darstellen, so wird er auf dasselbe Ausgabeformat gebracht, das auch flektierte Formen haben.
Bei allen anderen Analysen werden nur Kategoriesegmente umformatiert oder gelöscht.

5.5 Die Projektdatei

Jedes in Malaga geschriebene Grammatiksystem besteht aus mehreren Komponenten. Welche Dateien den verschiedenen Komponenten zugeordnet sind, wird in der Projektdatei spezifiziert.
Die Projektdatei der vorliegenden Arbeit befindet sich in der Datei `dmm.pro` (siehe auch Anhang D) und umfaßt die folgenden Komponenten:

- In der Segmentdatei `rules/dmm.seg` müssen alle in der Grammatik verwendeten Segmente und Multisegmente deklariert werden.
- In der Allomorphregeldatei `rules/dmm.all` befinden sich die Regeln zur Generierung des Allomorphlexikons aus den Grundformlexika. Die Dateien `rules/i_dmm.kasus` und `rules/i_dmm.ablaut` werden durch eine `include`-Anweisung in die Allomorph-Regel-Datei eingelesen.
- Die Lexikondatei `rules/dmm.lex` enthält das Grundformlexikon. Alle Teillexika (s.u.) werden per `include`-Anweisung eingelesen.
- Die Kombinationsregeldatei `rules/dmm.mor` enthält die Kombinationsregeln zur Konkatenation der Allomorphe. Die Datei `rules/i_dmm.kombi` wird per `include`-Anweisung eingelesen.

5.6 Das Format der Ausgabe

Analysen mit dem vorliegenden Grammatiksystem werden in folgendem Format ausgegeben:

- Partikel, Abkürzungen und Interpunktionszeichen sind morphologisch unveränderlich und werden im selben Format wie deren Lexikoneinträge ausgegeben.
- Verben weisen immer `[Klasse: Verb]` auf sowie das Attribut `Konjugation`. Dieses ist eine Liste von Subjekt-Tempus-Modus-Kombinationen; jede Subjekt-Tempus-Modus-Kombination enthält die Attribute `Subjekt` (Person und Numerus des Subjekts), `Tempus` und `Modus`. Zusätzlich kann noch das Attribut `Präfixe` vorhanden sein, wenn dies bereits im Grundformlexikon vorhanden ist.
- Substantive weisen neben `[Klasse: Substantiv]` immer noch das Attribut `Genus` und `KasusNumerus` (Kasus-Numerus-Kombination) auf.
- Adjektive weisen neben `[Klasse: Adjektiv]` immer noch die Attribute `Endung` (Endung des Adjektivs, mögliche Werte `e, es, er, en` und `em`) und `Komparation` (Komparationsgrad) auf.
- Adverbien weisen neben `[Klasse: Adverb]` immer noch das Attribut `Komparation` (Komparationsgrad) auf.
- Namen weisen neben `[Klasse: Name]` immer noch das Attribut `Genus` auf. Ist im Grundformlexikon noch das Attribut `NamKl` (Namensart) vorhanden, so ist es auch in der Ausgabe vorhanden.

Kapitel 6

Ergebnis und Ausblick

Ziel der Arbeit war es, auf der Grundlage bestehender Lexika mithilfe des Programmapakets Malaga ein System zur automatischen Wortformerkennung zu entwickeln. Weiterhin sollte geprüft werden, inwieweit sich Malaga für die Implementation komplexer Morphologiegrammatiken eignet.

6.1 Die entwickelte Grammatik

Die entwickelte Morphologiegrammatik liefert beim Parsen großer Korpora von Zeitungs- und Gebrauchstexten recht gute Ergebnisse. So wurden von 691.000 laufenden Wortformen aus einem Korpus der Frankfurter Rundschau 92% analysiert; bei der Liste der Unique-Wortformen wurden 70% erkannt.
Bei einem weiteren Korpus aus Trivialliteratur wurden 94% der knapp 295.000 laufenden Wortformen und 80% der Unique-Wortformen analysiert.
Das vorliegende Grammatiksystem liefert also zuverlässige Analyseergebnisse. Es wird bereits als Morphologiekomponente für syntaktische und semantische Analysesysteme in Malaga verwendet.

6.2 Eignung von Malaga für morphologische Anwendungen

Nach der Meinung des Autors der vorliegenden Arbeit eignet sich Malaga sehr gut für die Erstellung von linksassoziativen Morphologiegrammatiken. Dies ist auf folgende Eigenschaften von Malaga zurückzuführen:

- Die Verwendung von Attribut-Werte-Paaren, die sich zu komplexen Strukturen kombinieren lassen, erlauben im Gegensatz zu den Vorgängerprogrammen von Malaga eine elegantere und auch linguistisch fundiertere Beschreibung;

- Die Möglichkeit, in den Allomorphregeln auch Nachfolgeregeln verwenden zu können (was bei den Vorgängerprogrammen ebenfalls nicht der Fall war), macht einfacher strukturierte und dennoch linguistisch adäquatere Allomorphregeln möglich. Auch die regelhafte Erzeugung zusätzlicher Information in den Allomorphregeln wird so erst sinnvoll möglich;

- Das Programmpaket Malaga stellt eine Zusammenstellung von Software dar, die dem Zweck angemessen ist und sehr zuverlässig läuft;

- Die guten Möglichkeiten zur Analysedarstellung und für das Debugging ermöglichen eine effiziente und ergonomische Grammatikentwicklung.

6.3 Ausblick

Zwar wurde versucht, ein möglichst breites Spektrum an Phänomenen im vorliegenden Grammatiksystem zu behandeln; dennoch existieren noch viele unebhandelte Details, die es in die Grammatik zu integrieren gilt. Lohnend wäre dabei die Aufnahme von reihenbildenden Suffixen und Suffixoiden in die Lexemlexika. Desweiteren erscheint es notwendig, die Lexika für Namen, Adjektive und Substantive so bald wie möglich durch Lexika zu ersetzen, die einen größeren Umfang haben und auch weniger fehlerträchtig als die momentan verwendeten Lexika. Für die Qualität der Analysen wäre es von Vorteil, wenn die Zahlen für den Gewichtungsmechanismus, die im Moment noch experimantelle Werte darstellen, durch korpusbasiert gewonnene Daten ersetzt würden.

Literatur

Beutel, B. (1995). *Programmierung Linksassoziativer Grammatiken mit MALAGA*. Friedrich-Alexander-Universität Erlangen-Nürnberg.

Bußmann, H. (1990). *Lexikon der Sprachwissenschaft* (Zweite, völlig neu bearbeitete Auflage ed.). Alfred Kröner Stuttgart.

Drosdowski, G. e. a. (1984). *Duden „Grammatik der Deutschen Gegenwartssprache“* (4., völlig neu bearbeitete Auflage ed.). Bibliographisches Institut, Mannheim; Wien; Zürich.

Engel, U. (1988). *Deutsche Grammatik*. Julius Groos Verlag Heidelberg.

Erben, J. (1983). *Einführung in die deutsche Wortbildungslehre*. Erich Schmidt Verlag Berlin.

Fleischer, W. (1982). *Wortbildung der deutschen Gegenwartssprache*. Max Niemeyer Verlag Tübingen.

Fleischer, W. und I. Barz (1982). *Wortbildung der deutschen Gegenwartssprache*. Max Niemeyer Verlag Tübingen.

Gutbrod, I. (1995). Eine korpusbasierte Evaluation der Verbanalysen von Mater. Magisterarbeit, Friedrich–Alexander–Universität Erlangen-Nürnberg.

Hausser, R. (1985). *Left-Associative Grammar and the Parser NEWCAT*. Center for the Study of Language and Information, Stanford University.

Hausser, R. (1988). Algebraic Definitions of Left-Associative Grammars. *Computers and Translation, Vol. 3.2*.

Hausser, R. (1989). *Principles of Computational Morphology*. Laboratory of Computational Linguistics, Carnegie-Mellon-University.

hausser, r. (1992). Complexity in Left-Associative Grammar. *Theoretical Computer Science, Vol. 103*.

Hausser, R. (1996). *Grundlagen der Computerlinguistik in der Linksassoziativen Grammatik (Vorlesungsskript)*. Friedrich-Alexander-Universität Erlangen Nürnberg.

Helbig, G. und J. Buscha (1993). *Deutsche Grammatik*. Langenscheidt Verlag Enzyklopädie Leipzig, Berlin, München.

Mater, E. (1966). *Deutsche Verben*. VEB Bibliographisches Institut Leipzig.

Schüller, G. und O. Lorenz (1994). LA-MORPH – ein linksassoziatives Morphologiesystem. *Linguistische Verifikation – Dokumentation zur ersten Morpholympics*.

Schreiber, J. (1995). Formalgrammatische Bschreibung und automatische Analyse von Verbalkomplexen der deutschen Sprache. Magisterarbeit, Friedrich-Alexander-Universität Erlangen-Nürnberg.

Oliver Lorenz
Drausnickstr. 32
91052 Erlangen

Lebenslauf

geboren am	18.06.1970 in Bayreuth
1976-80	Besuch der St.-Georgen-Grundschule in Bayreuth
1980-81	Besuch des Markgräfin-Wilhelmine-Gymnasiums in Bayreuth
1981-89	Besuch des Clavius-Gymnasiums in Bamberg
Mai 1989	Erreichen der Hochschulreife
Juni 1989 - August 1990	Grundwehrdienst
WS 1990/91 - SS 1991	Studium an der Otto-Friedrich-Universität Bamberg (Romanistik/Anglistik/Psychologie)
seit WS 1991	Studium an der Friedrich-Alexander-Universität Erlangen-Nürnberg (Linguistische Informatik/Anglistik/Romanistik)

WAHRHEITSGEMÄSSE ERKLÄRUNG

Ich erkläre hiermit wahrheitsgemäß, daß ich

1. die eingereichte Arbeit selbständig und ohne unerlaubte Hilfe angefertigt habe,

2. außer den im Schrifftumsverzeichnis angegebenen Hilfsmitteln keine weiteren benutzt habe und alle Stellen, die aus dem Schrifttum ganz oder annähernd entnommen sind, als solche kenntlich gemacht und einzeln nach ihrer Herkunft unter Bezeichnung der Ausgabe (Auflage und Jahr des Erscheinens), des Bandes und der Seite des benutzten Werkes in der Abhandlung nachgewiesen habe,

3. alle Stellen und Personen, welche mich bei der Vorbereitung und Abfertigung der Abhandlung unterstützten, genannt habe,

4. die Abhandlung noch keiner anderen Stelle zur Prüfung vorgelegt habe und daß dieselbe noch nicht anderen Zwecken - auch nicht teilweise - gedient hat.

Erlangen, den

Vor- und Zuname

Automatische Wortformerkennung für das Deutsche im Rahmen von MALAGA

(Anhänge)

Magisterarbeit

in der Philosophischen Fakultät II

(Sprach- und Literaturwissenschaften)

der Friedrich-Alexander-Universität

Erlangen-Nürnberg

vorgelegt von
Oliver Lorenz
geb. am 18.06.1970 in Bayreuth

Anhang A

Die Segmentdatei

```
#----------------------------------------------------------------------#
#                                                                      #
# dmm.seg                                                              #
# =======                                                              #
#                                                                      #
# Deutsche Malaga-Morphologie, Version 1.0                             #
#                                                                      #
# (c) 1996 Abteilung für Computerlinguistik der Universität            #
# Erlangen-Nürnberg                                                    #
#                                                                      #
# Segmentdatei                                                         #
#                                                                      #
# Oliver Lorenz, 29.02.1996                                            #
#                                                                      #
# orlorenz@linguistik.uni-erlangen.de                                  #
#                                                                      #
#----------------------------------------------------------------------#

# 1 Ausgabesegmente

Klasse;                 # Wortklasse

        Verb;
        Adjektiv;
        Substantiv;
        Adverb;
        Name;
        Artikel;
        PII;
        Präp;
        Abk;
        Num;
        Interpunkt;
        VorgPass;
        Kon;

Infinitiv;

Konjugation;            # Subjekt/Tempus/Modus bei Verben
        Subjekt;        # Subjekt bei Verben

                Sg1;
                Sg2;
                Sg3;
                Pl1;
                Pl2;
                Pl3;
                * Sg13     := < Sg1, Sg3 >;
                * Sg23     := < Sg2, Sg3 >;
                * Sg2+Pl1  := < Sg2, Pl1 >;
                * Sg2+Pl2  := < Sg2, Pl2 >;
                * Sg23+Pl2 := < Sg2, Sg3, Pl2 >;
                * Sg3+Pl2  := < Sg3, Pl2 >;
                * Pl13     := < Pl1, Pl3 >;



        Tempus;
                Präsens;
                Präteritum;
        Modus;
                Indikativ;
                Konjunktiv;
                Imperativ;
                * Indikativ+Konjunktiv := <Indikativ, Konjunktiv>;
                * Indikativ+Konjunktiv+Imperativ
                  := <Indikativ, Konjunktiv, Imperativ>;

Valenzen;               # Liste von Valenzlisten (alternative Valenzrahmen)
        AkkRefl;
```

```
        DatRefl;
        Inf;
        InfZu;
        InfZu_Subj;
        Inf_Subj;
        Akk_InfZu_Subj;
        AkkOpt_InfZu_Subj;
        DatOpt_InfZu_Subj;
        AkkRefl_InfZu_Subj;
        AkkReflOpt_InfZu_Subj;
        NomGS;
        AdjGS;
        Nebs;
        PräpErg;
        DatBen;

PII-Auxil;              # Hilfsverb, das zum PartizipII gehört
        PII-haben;
        PII-sein;

Präfixe;                # Liste von abtrennbaren Präfixen
        PxOFl;          # Präfixoberfläche

Genus;                  # Genus bei Substantiven
        Maskulinum;
        Neutrum;
        Femininum;
        * Kein_Genus         := <Maskulinum, Femininum, Neutrum>;
        * Maskulinum+Neutrum := <Maskulinum, Neutrum>;

KasusNumerus;           # Kasus und Numerus bei Substantiven
        * NomSg
         := < Sg1, Sg2, Sg3 >;
        * NomPl
         := < Pl1, Pl2, Pl3 >;
        GenSg;
        GenPl;
        DatSg;
        DatPl;
        AkkSg;
        AkkPl;
        * Nom
         := < Sg1, Sg2, Sg3, Pl1, Pl2, Pl3 >;
        * Gen
         := < GenSg, GenPl >;
        * Dat
         := < DatSg, DatPl >;
        * Akk
         := < AkkSg, AkkPl >;
        * NomSg+DatSg+AkkSg
         :=  < Sg1, Sg2, Sg3, DatSg, AkkSg >;
        * NomSg+DatSg+AkkSg+NomPl+GenPl+AkkPl
         := < Sg1, Sg2, Sg3, DatSg, AkkSg, Pl1, Pl2, Pl3, GenPl, AkkPl >;
        * NomSg+DatSg+AkkSg+Plural
         := < Sg1, Sg2, Sg3, DatSg, AkkSg, Pl1, Pl2, Pl3, GenPl, DatPl, AkkPl >;
        * NomSg+AkkSg
         := < Sg1, Sg2, Sg3, AkkSg >;
        * GenSg+DatSg
         := < GenSg, DatSg >;
        * GenSg+Plural
         := < GenSg, Pl1, Pl2, Pl3, GenPl, DatPl, AkkPl >;
        * DatSg+NomPl+GenPl+AkkPl
         := < DatSg, Pl1, Pl2, Pl3, GenPl, AkkPl >;
        * GenSg+DatSg+AkkSg
         := < GenSg, DatSg, AkkSg >;
        * DatSg+AkkSg+Plural
         := < DatSg, AkkSg, Pl1, Pl2, Pl3, GenPl, DatPl, AkkPl >;
        * DatSg+Plural
         := < DatSg, Pl1, Pl2, Pl3, GenPl, DatPl, AkkPl >;
        * DatSg+AkkSg
         := < DatSg, AkkSg >;
        * DatSg+Pl2
         := < DatSg, Pl2 >;
        * GenSg+DatSg+AkkSg+Plural
         := < GenSg, DatSg, AkkSg, Pl1, Pl2, Pl3, GenPl, DatPl, AkkPl >;
        * NomPl+GenPl+AkkPl
         := < Pl1, Pl2, Pl3, GenPl, AkkPl >;
        * NomPl+AkkPl
         := < Pl1, Pl2, Pl3, AkkPl >;
        * Singular+NomPl+GenPl+AkkPl
         := < Sg1, Sg2, Sg3, GenSg, DatSg, AkkSg, Pl1, Pl2, Pl3, GenPl, AkkPl>;
        * Singular+Plural
         := < Sg1, Sg2, Sg3, GenSg, DatSg, AkkSg, Pl1, Pl2, Pl3, GenPl, DatPl, AkkPl>;
        * Singular
         := < Sg1, Sg2, Sg3, GenSg, DatSg, AkkSg >;
        * Plural
         := < Pl1, Pl2, Pl3, GenPl, DatPl, AkkPl >;
        * Sg3+Gen+DatSg
         := < Sg3, GenSg, GenPl, DatSg >;
        * Sg3+AkkSg
         := < Sg3, AkkSg>;
        * Pl3+AkkPl
         := < Pl3, AkkPl>;

Endung;                 # Adjektivendung
        e;
        en;
        er;
        es;
```

```
        em;
        * ersnm := < e, en, er, es, em >;

Komparation;

        Positiv;
        Komparativ;
        Superlativ;

NamKl;
        OrtsName;
        TopolName;
        StaatsName;
        VorName;
        NachName;
        InPos;
        PräPos;

Typ;

Struktur;
        Syntax;
        Val;
        Ergebnis;               # Ergebnisliste
                Interrog;

                KoKon;
                SubKon;
                DassNebs;

                * Sg3+Dat
                 := < Sg3, DatSg, DatPl >;
                * Sg3+Pl3+Akk
                 := < Sg3, Pl3, AkkSg, AkkPl >;
                * Sg3+Dat+Akk
                 := < Sg3, DatSg, DatPl, AkkSg, AkkPl >;
                * Sg3+Gen+Dat
                 := < Sg3, GenSg, GenPl, DatSg, DatPl >;
                * Sg3+Gen
                 := < Sg3, GenSg, GenPl >;
                * Sg3+Akk
                 := < Sg3, AkkSg, AkkPl >;
                * Pl3+Gen+Dat+Akk
                 := < Pl3, GenSg, GenPl, DatSg, DatPl, AkkSg, AkkPl >;
                * Gen+Dat+Akk
                 := < GenSg, GenPl, DatSg, DatPl, AkkSg, AkkPl>;
                * Pl3+Akk
                 := < Pl3, AkkSg, AkkPl >;
                * Pl2+Dat
                 := < Pl2, DatSg, DatPl >;
                * Dat+Akk
                 := < DatSg, DatPl, AkkSg, AkkPl >;
                * Gen+Dat
                 := < GenSg, GenPl, DatSg, DatPl >;

                Punkt;
                Fragezeichen;
                Komma;
                Semikolon;
                Doppelpunkt;
                Ausrufezeichen;
                Schrägstrich;
                Klammer_auf;
                Klammer_zu;
                Bindestrich;
                Anführungszeichen;
                Kaufmund;
                Multzeichen;
                Pluszeichen;
                Zeichen;
                Unterstrich;

                Adv;
                AdvL;
                AdvD;
                AdvM;
                AdvK;
                AdvT;

WortStruktur;
        Basis;
        Allomorph;
#       Typ;
                WortStart;
                Subst-Subst-Komp;
                Adj-Subst-Komp;
                Verb-Subst-Komp;
                Adj-Adj-Komp;
                Subst-Adj-Komp;
                Adj-Verb-Komp;
                Subst-Verb-Komp;
                Diminutiv;
                Verb-Subst-Deriv;
                Verb-Adj-Deriv;
                Adj-Subst-Deriv;
                Subst-Adj-Deriv;
                Name-Adj-Deriv;
                PartizipII;
                InfinitivZu;
```

```
                Präfigierung;

Pron;


Grundform;

Wortform;

Gewicht;

# 2 interne Segmente

# 2.1 Allgemein

ja;
nein;
LemOFl;                 # Lemmaoberfläche
StammKlasse;            # Klasse, zu der eine Flexionsendung paßt
T_OFl;                  # Temporäres Oberflächenfeature

# 2.2 Substantive

Komp_Substantiv;        # Substantiv mit Fuge kombiniert
OblPxSubstantiv;        # Substantiv, das obligatorisch mit einem Präfix
                        # kombiniert werden muß

SgDekl;                 # Singulardeklinationsverhalten

        KeinSg;
        Sg_0;
        Sg_e_s;
        Sg_en;
        Sg_en_s;
        Sg_es_e;
        Sg_n;
        Sg_n_s;
        Sg_s;
        Sg_se_s;

PlDekl;                 # Pluraldeklinationsverhalten

        KeinPl;
        Pl_0;
        Pl_0_n;
        Pl_e_n;
        Pl_en;
        Pl_er_n;
        Pl_ien;
        Pl_n;
        Pl_nen;
        Pl_s;
        Pl_se_n;
        Pl_sen;
        Pl_ten;

        * ANYSUB   := < Sg_0, Sg_e_s, Sg_en, Sg_n, Sg_n_s, Sg_s, Sg_es_e, Sg_se_s,
                     nein, Pl_0, Pl_0_n, Pl_e_n, Pl_en, Pl_er_n, Pl_ien, Pl_n,
                     Pl_s, Pl_se_n, Pl_ten, Pl_nen, KeinPl >;
        * ANYSUB-s := < Sg_0, Sg_e_s, Sg_en, Sg_n, Sg_n_s, Sg_es_e, Sg_se_s,
                     nein, Pl_0, Pl_0_n, Pl_e_n, Pl_en, Pl_er_n, Pl_ien, Pl_n,
                     Pl_se_n, Pl_ten, Pl_nen, KeinPl >;
        * ANYP-E_N := < Pl_0, Pl_0_n, Pl_en, Pl_er_n, Pl_ien, Pl_n, Pl_s, Pl_se_n,
                     Pl_ten, Pl_nen, nein>;
        * ANYP-EN  := < Pl_0, Pl_0_n, Pl_e_n, Pl_er_n, Pl_ien, Pl_n, Pl_s, Pl_se_n,
                     Pl_ten, Pl_nen, nein>;
        * ANYS-EN  := < Sg_0, Sg_e_s, Sg_es_e, Sg_s, nein >;
        * SG_D_E   := < Sg_e_s, Sg_es_e >;
        * SG_D_0   := < Sg_0, Sg_s, nein >;

# 2.3 Verben

AK;                     # Ablautklasse
        Abl_Norm_a1;
        Abl_Norm_a2;
        Abl_Norm_a3;
        Abl_Norm_a4;
        Abl_Norm_a5;
        Abl_Norm_au1;
        Abl_Norm_au2;
        Abl_Norm_au3;
        Abl_Norm_e1;
        Abl_Norm_e2;
        Abl_Norm_e3;
        Abl_Norm_e4;
        Abl_Norm_e5;
        Abl_Norm_e6;
        Abl_Norm_e7;
        Abl_Norm_e8;
        Abl_Norm_e9;
        Abl_Norm_e10;
        Abl_Norm_e11;
        Abl_Norm_e12;
        Abl_Norm_ei1;
        Abl_Norm_ei2;
        Abl_Norm_ei3;
        Abl_Norm_i1;
        Abl_Norm_i2;
        Abl_Norm_i3;
```

```
        Abl_Norm_i4;
        Abl_Norm_ie1;
        Abl_Norm_ie2;
        Abl_Norm_ie3;
        Abl_Norm_o1;
        Abl_Norm_o2;
        Abl_Norm_u;
        Abl_Norm_ä1;
        Abl_Norm_ä2;
        Abl_Norm_ö;
        Abl_Norm_ü;
        Abl_Kons_echt;
        Abl_Kons_eh1;
        Abl_Kons_eh2;
        Abl_Kons_ehm;
        Abl_Kons_eid;
        Abl_Kons_itz;
        Abl_Kons_ieh;
        Abl_Kons_ess;
        Abl_tFlx_enk;
        Abl_tFlx_ing;
        Abl_tFlx_enn;
        Abl_tFlx_ind;

AV;                     # Vokale der Ablautreihe
BW;                     # Beschreibung des Bestimmungswortes
                        # bei den Kompositionsmustern

PII-Ge;                 # Partizipbildung mit/ohne "ge"
        Ge;
        Ge-;

FlexTyp;

        schwach;
        stark;
        * schwach+stark := < schwach, stark >;

SF;
        SF_1;           # Stammformen
        SF_2;
        SF_3;
        SF_4;
        SF_5;
        SF_el;
        SF_l;
        SF_er;
        SF_r;

        * SF_el_er      := <    SF_el,  SF_er   >;
        * SF_1_el_er    := <    SF_1,   SF_el,  SF_er   >;
        * SF_1_l_r_er   := <    SF_1,   SF_l,   SF_r,   SF_er>;
        * SF_1_l_er     := <    SF_1,   SF_l,   SF_er>;
        * SF_12         := <    SF_1,   SF_2    >;
        * SF_12_l_er    := <    SF_1,   SF_2,   SF_l,   SF_er   >;
        * SF_12_el_er   := <    SF_1,   SF_2,   SF_el,  SF_er   >;
        * SF_13         := <    SF_1,   SF_3>;
        * SF_2_el_er    := <    SF_2,   SF_el,  SF_er   >;
        * SF_34         := <    SF_3,   SF_4    >;

PhonEnd;                # Auslaut von Stämmen

        Dental;
        Sibilant;
        unmarkiert;

        * allePhonEnd   := < Dental, Sibilant, unmarkiert>;
        * nichtSibilant := < Dental, unmarkiert >;
        * nichtDental   := < Sibilant, unmarkiert >;

# 2.3 Adjektive

Komp_Adjektiv;                  # Adjektiv mit (Bindestrich)Fuge,
                                # zu Attribut Klasse

AdjFlex;                                # Flexionsverhalten des Adjektivs

        AdjFlex_a;
        AdjFlex_d;
        AdjFlex_k;
        AdjFlex_s;

        * AdjFlex_ad    := < AdjFlex_a, AdjFlex_d>;
        * AdjFlex_adk   := < AdjFlex_a, AdjFlex_d, AdjFlex_k>;
        * AdjFlex_adks  := < AdjFlex_a, AdjFlex_d, AdjFlex_k, AdjFlex_s>;
        * AdjFlex_aks   := < AdjFlex_a, AdjFlex_k, AdjFlex_s>;
        * AdjFlex_as    := < AdjFlex_a, AdjFlex_s>;
        * AdjFlex_dk    := < AdjFlex_d, AdjFlex_k>;
        * AdjFlex_dks   := < AdjFlex_d, AdjFlex_k, AdjFlex_s>;
        * AdjFlex_ks    := < AdjFlex_k, AdjFlex_s>;

        * Ad_Deriv      := < AdjFlex_a, AdjFlex_d>;

# PhonEnd;
        e-Einfügung;
        s-Tilgung;
        * unmarkiert_e-Einfügung := <unmarkiert, e-Einfügung>;

HeitSx;
```

```
        heit;
        keit;
        igkeit;

Nullflex;

# 2.4 Flexionsendungen

FlexSx;                         # Marker für Flexionsendung
Flexion;                        # Beschreibung der Flexionsendung

ErgKlasse;                      # Ergebnis-Wortklasse


# 2.5 Suffixe, Präfixe etc.

Fuge;

        Fuge_s;         # Sonderfuge -s-
        Fuge_en;        # Sonderfuge -en-
        Fuge_Kolon;     # Bindestrich-"Fuge"
Suche;
Lösche;
Komb;
Sx;
Px;
OFl;
in_Liste;
kongr;
exakt;
ist_in;
SxTyp;
Präteritalinfix;
Präfix;
Präfixoid;

# 3 Sonstiges

Paradigma;
Blockieren;
Freigeben;
Homonym;
unbekannt;

Orthographie;
        Ortho-ß;
        Ortho-ss;
        FlexIndex;

# momentan ungenutzte Information aus dem Mater

Mater;
        h;
        VerbOPx;
        alt;
        poet;
        umg;
        wieder;
        unfl;
```

Anhang B

Die Allomorph-Regeln

B.1 Die Datei `dmm.all`

```
#----------------------------------------------------------------------#
#                                                                      #
# dmm.all                                                              #
# =======                                                              #
#                                                                      #
# Deutsche Malaga-Morphologie, Version 1.0                             #
#                                                                      #
# (c) 1996 Abteilung für Computerlinguistik der Universität            #
# Erlangen-Nürnberg                                                    #
#                                                                      #
# Allomorph-Regeln                                                     #
#                                                                      #
# Oliver Lorenz, 29.02.1996                                            #
#                                                                      #
# orlorenz@linguistik.uni-erlangen.de                                  #
#                                                                      #
#----------------------------------------------------------------------#

initial-rules Startregel;

# Hinter dem Schlüsselwort initial-rules folgen die Regeln, die als erste
# aufgerufen werden können.

#----------------------------------------------------------------------#
#                                                                      #
#                              Startregel                              #
#                                                                      #
#----------------------------------------------------------------------#

allo-rule Startregel;

# Die Startregel prüft nach dem Einlesen eines Lemmas, um was für eine Art
# von Lemma es sich handelt und verzweigt in die entsprechende Regel;

  start $Lemma;

  case

  of Klasse in $Lemma:
  # Das Attribut Klasse ist nur bei solchen Einträgen vorhanden, die Lexeme
  # darstellen können

    case

    of  $Lemma.Klasse = Verb:
    # Verben

      case

      of AK in $Lemma:
      # ablautende (semi-irreguläre) Verben,
      # erkennbar an dem Attribut AK

        result $Lemma, rules Verb-Ablaut;

      else result $Lemma, rules Verb-Regulär;
      # reguläre und semi-reguläre Verben

      end case;

    of $Lemma.Klasse in <Substantiv, OblPxSubstantiv>:
    # Substantive und solche, die obligatorisch präfigiert
    # werden müssen

      result $Lemma, rules Substantiv-Start;
```

```
    of $Lemma.Klasse = Adjektiv:
    # Adjektive

      result $Lemma, rules Adjektiv-Start;

    of $Lemma.Klasse = Name:
    # Namen

      result $Lemma, rules Namen;

    else   result $Lemma, rules Default;
    # Partikeln

    end case;

  of Paradigma in $Lemma:
  # irreguläre Verben und Adjektive

    result $Lemma, rules Paradigma;

  else     result $Lemma, rules Default;
  # alles sonstige (Flexionsendungen, Suffixe)

  end case;

end allo-rule;

#----------------------------------------------------------------------#
#                                                                      #
#                         Regeln für Verben                            #
#                                                                      #
#----------------------------------------------------------------------#

allo-rule Verb-Ablaut;

  # Regelbeschreibung:

  # Diese Regel erzeugt die Allomorphe der ablautenden (semi-irregulären)
  # Verben. Dabei wird im Lexikon nur die Ablautreihe des Verbs vermerkt
  # sowie der Ablautende Vokal vermerkt. Bei Ablautreihen, bei denen auch
  # ein Konsonantenwechsel auftritt (z.B. "gehen - ging - gegangen"), muß
  # die in der Beschreibung der Ablautreihe (die in der include-Datei
  # "i_dmm.ablaut" enthalten ist) auch der Konsonantenwechsel mit
  # beschrieben werden. Nicht jedoch muß auf Konsonanten*verdoppelung* wie
  # in "treten - tritt" oder Reduktion auf Einfachkonsonanten wie in
  # "kommen - kam" beschrieben werden, da dies in der nachfolgenden Regel
  # Verb-Orthographie regelhaft aufgrund der Vokallänge vorgenommen wird.

  start $Eingabe;
  # liest Lexikoneintrag ein

  assert match $Eingabe.LemOFl = $Of1:"[^{]*" & "{" & $Of2:"[^}]*" & "}" & $Of3:".*" & "en";
  # zerlegt Grundformoberfläche

  include "i_dmm.ablaut";
  # definiert die Variable $Ablauttabelle

  choose $Ablautreihe in $Ablauttabelle;
  # holt Ablautreihe aus der Ablauttabelle

  ? $Eingabe.AK = $Ablautreihe.AK;
  # prüft, ob richtige Ablautreihe

  choose $Ablautbeschreibung in $Ablautreihe.AV;
  # holt Ablautbeschreibung aus der Ablautreihe

  $Ausgabe := $Eingabe/LemOFl & [T_OFl: $Of1 & $Ablautbeschreibung.1L & $Of3,
                                FlexTyp: stark,
                                SF: $Ablautbeschreibung.2L,
                                Basis: $Of1 & $Of2 & $Of3 & "en"];
  # erzeugt Allomorphoberfläche, Flexionstyp und Stammform

  if $Eingabe.AK in <Abl_tFlx_enk, Abl_tFlx_ing, Abl_tFlx_enn, Abl_tFlx_ind>
     and $Ausgabe.SF ~ SF_34 then
  # Die Präteritalstämme der gemischt stark-schwachen Verben werden
  # schwach flektiert

    set $Ausgabe := $Ausgabe & [FlexTyp: schwach];

  end if;

  if not $Ablautbeschreibung.3L = PII then
  # An den Attributen PII-Auxil und PII-Ge wird von den Kombiregeln
  # die Partizip-II-Stammform erkannt. Bei den Stammformen, die dafür
  # nicht in Frage kommen, werden diese Attribute gelöscht. Partizip-II
  # Stämme sind diejenigen, die in der Ablauttabelle als solche gekenn-
  # zeichnet sind.

    set $Ausgabe := $Ausgabe/PII-Auxil/PII-Ge;

  end if;

  if $Ausgabe.SF ~ SF_12 then

    set $Ausgabe := $Ausgabe & [Tempus: Präsens];

  elseif $Ausgabe.SF ~ SF_34 then

    set $Ausgabe := $Ausgabe & [Tempus: Präteritum];
```

```
  end if;
  # Hier werden den Stämmen die zugehörigen Tempora zugeordnet

  result $Ausgabe, rules Verb-Orthographie;
  # erzeugt Ergebnis

end allo-rule; # Verb-Ablaut

#-----------------------------------------------------------------------#

allo-rule Verb-Regulär;

  # Diese Regel erzeugt die Allomorphe der nicht ablautenden (regulären
  # und semi-regulären) Verben. Dabei werden nur die Oberflächen der
  # Verben, die auf "eln" oder "ern" enden (semi-reguläre Verben),
  # bearbeitet. Die regulären Verben laufen unverändert durch die Regel.

  start $Eingabe;
  # liest Lexikoneintrag ein;

  case

  of match $Eingabe.LemOFl = $Of1:".*" & "e" & $Konsonant:"[lr]" & "n":
  # Die semi-regulären Verben enden im Infinitv auf -eln oder -ern
  # lächeln =-> "lächel", "lächl"
  # meistern =-> "meister", "meistr"

    $SF1 := nil;
    $SF2 := nil;

    if     $Konsonant = "l" then
    # -eln-Verben

      set $SF1 := SF_el;
      set $SF2 := SF_l;


    elseif $Konsonant = "r" then
    # -ern-Verben

      set $SF1 := SF_er;
      set $SF2 := SF_r;

    else error;

    end if;

    $Ausgabe1 := $Eingabe/LemOFl & [T_OFl: $Of1 & "e" & $Konsonant,
                                    FlexTyp: schwach,
                                    Tempus: Präsens,
                                    SF: $SF1,
                                    Basis: $Eingabe.LemOFl];

    $Ausgabe2 := $Eingabe/LemOFl/PII-Auxil/PII-Ge
                                  & [T_OFl: $Of1 & $Konsonant ,
                                    FlexTyp: schwach,
                                    Tempus: Präsens,
                                    SF: $SF2,
                                    Basis: $Eingabe.LemOFl];

    # normale und verkürzte Stammform erzeugen

    result $Ausgabe1, rules Verb-Orthographie;
    result $Ausgabe2, rules Verb-Orthographie;
    # erzeuge zwei Ergebnisse

   of match $Eingabe.LemOFl = $Of1:".*" & "en|n":
   # reguläre Verben

     $Ausgabe := $Eingabe/LemOFl & [T_OFl: $Of1,
                                   FlexTyp: schwach,
                                   SF: SF_12,
                                   Tempus: Präsens,
                                   Basis: $Eingabe.LemOFl];

    result $Ausgabe, rules Verb-Orthographie;
    #erzeuge Ergebnis

  end case;

end allo-rule; # Verb-Regulär

#-----------------------------------------------------------------------#

allo-rule Verb-Orthographie;

  # Diese Regel nimmt Korrekturen vor, die Folge der nicht phonematischen
  # Beziehung zwischen Laut und Zeichen der deutschen Orthographie
  # sind. Dabei werden zum einen Doppelkonsonanten nach langen/kurzen
  # Vokalen eliminiert/erzeugt, zum anderen ss/ß je nach Bedarf ineinander
  # umgewandelt.

  start $Eingabe;
  # Lies Allomorph von der Deque

  case

  of match $Eingabe.T_OFl = $Of1:"[^:]*" & $Vok:"[aäeioöuü]" & "ss":
```

```
  # kurzer Vokal vor ss, z.B. "lassen"; Regel Verb-Ablaut erzeugt "lass"
  # gilt sowohl für ablautende als auch nicht ablautende Verben

  case

  of  $Eingabe.SF ~ SF_1:               # Stammform 1 ?

    $Ausgabe1  := $Eingabe/T_OFl & [T_OFl: $Of1 & $Vok & "ß",
                                    Orthographie: Ortho-ß];
    # "ihr laßt"

    $Ausgabe2  := $Eingabe/T_OFl & [T_OFl: $Of1 & $Vok & "ss",
                                    Orthographie: Ortho-ss];
    # "ich lasse"

    result $Ausgabe1, rules Verb-Phon-Info;
    result $Ausgabe2, rules Verb-Phon-Info;
    # erzeuge Ergebnisse

  else # andere Stammformen

    $Ausgabe  := $Eingabe/T_OFl & [T_OFl: $Of1 & $Vok & "ß" ];
    # "du läßt"

    result $Ausgabe, rules Verb-Phon-Info;
    # Erzeuge Ergebnis

  end case;

of $Eingabe.FlexTyp = stark:
# Rest der Regel bezieht sich nur auf ablautende Verben

  case

  of match $Eingabe.T_OFl =
    $Of1:"[^:]*" & $Vok:"[aäeioöuü]" & $Kons:"[ft]" :
    # Kurzer Vokal gefolgt von einfachem Konsonanten
    # z.B. "saufen"; Regel Verb-Ablaut erzeugt "sof"

    $Ausgabe  := $Eingabe/T_OFl &
    [ T_OFl: $Of1 & $Vok & $Kons & $Kons ];

    # verdopple Konsonanten: "soff"

    result $Ausgabe, rules Verb-Phon-Info;

    # erzeuge Ergebnis

  of match $Eingabe.T_OFl =
    $Of1:"[^:]*" & ":" & "ff|ll|mm|tt":

    assert match $Eingabe.T_OFl = ".*" & $Kons:".";

    # langer Vokal gefolgt von Doppelkonsonanten
    # z.B. "fallen"; Regel Verb-Ablaut erzeugt "fie:ll"

    $Ausgabe  := $Eingabe & [T_OFl: $Of1 & $Kons];
    # reduziere zu einfachem Konsonanten: "fiel"

    result $Ausgabe, rules Verb-Phon-Info;

    # Erzeuge Allomorph

  of match $Eingabe.T_OFl = $Of1:"[^:]*" & $Vok:"[aäeioöuü]" & "ß" :

  # kurzer Vokal vor ß, z.B. "schmeißen"
  # Regel Verb-Ablaut erzeugt "schmiß"

    $Ausgabe1  := $Eingabe/T_OFl & [T_OFl: $Of1 & $Vok & "ß",
                                    Orthographie: Ortho-ß];

    # "er schmiß"

    $Ausgabe2  := $Eingabe/T_OFl & [T_OFl: $Of1 & $Vok & "ss",
                                    Orthographie: Ortho-ss];

    # "sie schmissen"

    result $Ausgabe1, rules Verb-Phon-Info;
    result $Ausgabe2, rules Verb-Phon-Info;

  of match $Eingabe.T_OFl =  $Of1:"[^:]*" & ":" & "ss" :

    # langer Vokal vor ss, z.B. "lassen"
    # Regel Verb-Ablaut erzeugt "lie:ss"

    $Ausgabe  := $Eingabe/T_OFl & [T_OFl: $Of1 & "ß" ];

    result $Ausgabe, rules Verb-Phon-Info;

  else

    case

    of match $Eingabe.T_OFl = $Of1:"[^:]*" & ":" & $Of2:".*":
    # wenn unproblematische Endung, dann Längenzeichen einfach
    # löschen

      $Ausgabe  := $Eingabe/T_OFl & [T_OFl: $Of1 & $Of2 ];
```

```
        result $Ausgabe, rules Verb-Phon-Info;

      else result $Eingabe, rules Verb-Phon-Info;
      # keine Auffälligkeit, keine Längenzeichen

      end case;

    end case;

  else result $Eingabe, rules Verb-Phon-Info;

  end case;

end allo-rule; # Verb-Orthographie

#----------------------------------------------------------------------#

allo-rule Verb-Phon-Info;

  # Diese Regel erzeugt die Information über die Endung des Stammes; sie
  # kann entweder unmarkiert sein (= unmarkiert), auf "[dt][mn]?" oder
  # "[gk][mn]" oder "ch[mn]" (= Dental) oder auf "[sßxz]" (= Sibilant) enden.

  start $Eingabe;
  # Lies Allomorph von der Deque

  case

  # Selbsterklärend

  of match $Eingabe.T_OFl = $OFl:".*" & "[dt]|[^rlmnaeiouäöüh][mn]|chn":
  # e-Erweiterung bei Stammendung, die auf dentalen Plosiv oder nasalische
  # Doppelkonsonanz (Frikativ/Plosiv + Nasal) endet

    $Ausgabe := $Eingabe & [PhonEnd: Dental];

    result $Ausgabe, rules Verb-NullFlex;

  of match $Eingabe.T_OFl = $OFl:".*" & "[sßxz]":
  # s-Tilgung nach Sibilanten

    $Ausgabe := $Eingabe & [PhonEnd: Sibilant];

    result $Ausgabe, rules Verb-NullFlex;

  else

    $Ausgabe := $Eingabe & [PhonEnd: unmarkiert];

    result $Ausgabe, rules Verb-NullFlex;

  end case;

end allo-rule; # Verb-Phon-Info

#----------------------------------------------------------------------#

allo-rule Verb-NullFlex;

# In drei Fällen kann ein Verbstamm bereits ein fertig konjugierte Form
# sein:
# 1 Bei ablautenden Verben, bei denen e/i-Wechsel auftritt, ist die
#   zweite Stammform ein fertiger Imperativ ("gib")
# 2 Bei ablautenden Verben, bei denen die zweite Stammform eine umgelautete
#   Variante der ersten Stammform ist oder bei der e/i-Wechsel auftritt,
#   und wenn der Stamm auf einen Dental endet, ist die zweite Stammform
#   bereits die 3.Pers.Sing.Ind.Präs. ("hält")
# 3 Stämme, die die dritte Stammform darstellen, sind bereits die 1. und 3.
#   Pers.Sg.Ind.Prät.
# Die Möglichkeiten 1 und 2 können beide zutreffen ("tritt").
# Die Endregel erkennt diese Stämme am Inhalt der Liste NullFlex und gibt
# sie in geeigneter Form aus.

  start $Links;

  $Ausgabe := $Links;

  $Nullflex := <>;

  if {$Links.AK in <Abl_Norm_e1, Abl_Norm_e2, Abl_Norm_e3, Abl_Norm_e4,
                    Abl_Norm_e5, Abl_Norm_e7, Abl_Norm_e8, Abl_Norm_e11,
                    Abl_Norm_e12, Abl_Kons_ehm, Abl_Kons_ess>
                   # Ablautreihen mit e/i-Wechsel

      and $Links.SF = SF_2} then

    set $Nullflex := $Nullflex & <Imperativ>;

  end if;

  if {$Links.AK in <Abl_Norm_e1,  Abl_Norm_e2,  Abl_Norm_e3,  Abl_Norm_e4,
                    Abl_Norm_e5,  Abl_Norm_e7,  Abl_Norm_e8,  Abl_Norm_e11,
                    Abl_Norm_e12, Abl_Norm_a1,  Abl_Norm_a2,  Abl_Norm_a3,
                    Abl_Norm_a4,  Abl_Norm_a5>
                   # Ablautreihen mit umgelauteter zweiter Stammform
                   # oder e/i-Wechsel

      and $Links.SF = SF_2 and $Links.PhonEnd = Dental} then
```

```
    set $Nullflex := $Nullflex & <Präsens>;

  end if;

  if $Links.SF ~ SF_3 and
    {not $Links.AK in <Abl_tFlx_enk, Abl_tFlx_ing, Abl_tFlx_enn>} then
    # dritte Stammform, nicht gemischt stark-schwache Verben

    set $Nullflex := $Nullflex & <Präteritum>;

  end if;

  if not $Nullflex = <> then
  # Nullflexion mit diesem Stamm möglich, Attribut NullFlex
  # erzeugen

    set $Ausgabe := $Ausgabe & [NullFlex: $NullFlex];

  end if;

  set $Ausgabe := $Ausgabe/AK;
  # Ablautklasse löschen, da nicht mehr notwendig

  result $Ausgabe, rules Erzeuge-Allo;

end allo-rule;

#----------------------------------------------------------------------#
#                                                                      #
#                         Regeln für Substantive                       #
#                                                                      #
#----------------------------------------------------------------------#

allo-rule Substantiv-Start;

  # Diese Regel wird von allen Substantiv-Lexikoneinträgen als erste
  # durchlaufen. Sie erzeugt Plural- und Diminutivablaute und
  # die Flexionsendungen bei einigen Wörtern griech.-lat.-ital.
  # Ursprungs.
  # Die Streichung von e/en am Allomorphende für die Diminutivbildung
  # sowie die Behandlung von ß am Allomorphende erfolgt erst in
  # den Nachfolgeregeln.

  start $Eingabe;
  # Lies Allomorph ein


  ? $Eingabe.Klasse in <Substantiv, OblPxSubstantiv>;
  # Substantivlemma?

  case

  of match $Eingabe.LemOFl = $OFl1:".*" & "{" & $StVok:"[aou]" & "}" & $OFl2:".+" :

    # hier werden alle "markierten" Umlaute (d.h. die, die im Lexikon
    # mit einem großen Vokal eingetragen sind) bearbeitet; es werden
    # zwei Allomorphe erzeugt.

    $Uml1 := "";
    $Uml2 := "";

    if     $StVok = "a" then
      set $Uml1  := "a" ;
      set $Uml2  := "ä" ;

    elseif $StVok = "o" then
      set $Uml1  := "o" ;
      set $Uml2  := "ö" ;

    elseif $StVok = "u" then
      set $Uml1  := "u" ;
      set $Uml2  := "ü" ;

    end if;

    $Ausgabe1 := $Eingabe/LemOFl &

    [T_OFl: $Ofl1 & $Uml1 & $OFl2,
     Basis: $Ofl1 & $Uml1 & $OFl2,
     Diminutiv: nein,
     PlDekl: nein];

    # Singularallomorph;

    $Ausgabe2 := $Eingabe/LemOFl &

    [T_OFl: $Ofl1 & $Uml2 & $OFl2,
     Basis: $Ofl1 & $Uml1 & $OFl2,
     Diminutiv: Plural,
     SgDekl: nein];

    # Plural-/Diminutivallomorph

    result $Ausgabe1, rules Verkürze-Substantiv-e-en;
    result $Ausgabe2, rules Verkürze-Substantiv-e-en;

  of match $Eingabe.LemOFl = $OFl1:".*" & "{" & $StVok:".+" & "}":
```

```
  ? $Eingabe.PlDekl = Pl_0;

  $Uml1 := nil;
  $Uml2 := nil;

  # Fremdwörter griech.-lat.-ital. Ursprungs mit deutscher
  # Ersatzendung -en oder fremder Endung (vgl. Duden 4, S.251)
  # (Diminutivbildung ?)


  if          $StVok = "a"    then    set $Uml1 := "a";      set $Uml2 := "en";
  elseif      $StVok = "o"    then    set $Uml1 := "o";      set $Uml2 := "en";
  elseif      $StVok = "us" then      set $Uml1 := "us";     set $Uml2 := "en";
  elseif      $StVok = "um" then      set $Uml1 := "um";     set $Uml2 := "en";
  elseif      $StVok = "dus" then     set $Uml1 := "dus";    set $Uml2 := "di";
  elseif      $StVok = "eus" then     set $Uml1 := "eus";    set $Uml2 := "ei";
  end if;

  $Ausgabe1 := $Eingabe/LemOFl &

  [T_OFl: $Ofl1 & $Uml1,
   Basis: $Ofl1 & $Uml1,
   Diminutiv: nein,
   PlDekl: nein];

  # Singularallomorph

  $Ausgabe2 := $Eingabe/LemOFl &

  [T_OFl: $Ofl1 & $Uml2,
   Basis: $Ofl1 & $Uml1,
   Diminutiv: nein,
   SgDekl: nein];

  # Pluralallomorph

  result $Ausgabe1, rules Verkürze-Substantiv-e-en;
  result $Ausgabe2, rules Verkürze-Substantiv-e-en;


of match $Eingabe.LemOFl =
  $OFl1:".*[^e]?" & $StVok:"[aou]|aa|oo|au"
  & $OFl2:"[bcdfghjklmnpqrstvwxyz]+|er" & $OFl3:"e?n?" :

  # hier werden die Lexikoneinträge bearbeitet, bei denen keine
  # Umlautung im Lexikon markiert ist -> Umlautung nur beim
  # Diminutivallomorph

  $Uml1 := nil;
  $Uml2 := nil;

  if     $StVok = "a" then

    set $Uml1 := "a";
    set $Uml2 := "ä";

  elseif $StVok = "aa" then

    set $Uml1 := "aa";
    set $Uml2 := "ä";

  elseif $StVok = "o" then

    set $Uml1 := "o";
    set $Uml2 := "ö";

  elseif $StVok = "oo" then

    set $Uml1 := "oo";
    set $Uml2 := "ö";

  elseif $StVok = "u" then

    set $Uml1 := "u";
    set $Uml2 := "ü";

  else

    set $Uml1 := "au";
    set $Uml2 := "äu";

  end if;

  $Ausgabe1 := $Eingabe/LemOFl &

  [T_OFl: $Ofl1 & $Uml1 & $OFl2 & $OFl3,
   Basis: $Ofl1 & $Uml1 & $OFl2 & $OFl3,
   Diminutiv: nein];

  # Singular-/Pluralallomorph

  $Ausgabe2 := $Eingabe/LemOFl &

  [T_OFl: $Ofl1 & $Uml2 & $OFl2,
   Basis: $Ofl1 & $Uml1 & $Ofl2,
   Diminutiv: ja,
   SgDekl: nein,
   PlDekl: nein,
   Genus: nein];
```

```
    # Diminutivallomorph

    result $Ausgabe1, rules Verkürze-Substantiv-e-en;
    result $Ausgabe2, rules Verkürze-Substantiv-e-en;

  else

    # Default: keine Umlautung


    $Ausgabe := $Eingabe/LemOFl & [T_OFl: $Eingabe.LemOFl,
                                   Basis: $Eingabe.LemOFl,
                                   Diminutiv: Plural];
    # Singular-/Plural-/Diminutivallomorph

    result $Ausgabe, rules Verkürze-Substantiv-e-en;

  end case;

end allo-rule; # Substantiv-Umlaut


#----------------------------------------------------------------------#


allo-rule Verkürze-Substantiv-e-en;

# Diese Regel verkürzt bei Bedarf Substantivallomorphe mit finalem e/en:
#
# > bei Plural/Diminutivallomorph wird ein weiteres, um e/en ver-
#   kürztes Allmorph erzeugt (z.B. ofen - öfen - öf/chen); erkennbar
#   am Attribut [ Diminutiv: Plural ]
# > bei Diminutivallomorphen wird e/en abgetrennt; erkennbar am
#   Attribut [ Diminutiv: ja ]

  start $Eingabe;
  # lies Allomorph von der Deque;

  case

  of match $Eingabe.T_Ofl = $OFl1:"....+" & $OFl2:"e|en" :

    # Allomorph endet auf "e/en"

    case

    of $Eingabe.Diminutiv = Plural :

      # Plural und Diminutiv

      $Ausgabe1 := $Eingabe/T_OFl & [T_OFl: $OFl1 & $OFl2,
                                     Diminutiv: nein ];

      $Ausgabe2 := $Eingabe/T_OFl & [T_OFl: $OFl1,
                                     SgDekl: nein,
                                     PlDekl: nein,
                                     Genus: nein,
                                     Diminutiv: ja ];

      result $Ausgabe1, rules Substantiv-Orthographie;
      result $Ausgabe2, rules Substantiv-Orthographie;

    of $Eingabe.Diminutiv = ja :

      # nur Diminutiv

      $Ausgabe  := $Eingabe/T_OFl & [T_OFl: $OFl1,
                                     SgDekl: nein,
                                     PlDekl: nein,
                                     Genus: nein,
                                     Diminutiv: ja ];

      result $Ausgabe, rules Substantiv-Orthographie;

    else result $Eingabe, rules Substantiv-Orthographie;

    end case;

  else

    # endet nicht auf "e/en"

    $Ausgabe := nil;

    if $Eingabe.Diminutiv = Plural then

      # Die Information, ob der Diminutivstamm gleichzeitig Pluralstamm
      # ist, wird nicht mehr benötigt; daher wird bei solchen Stämmen
      # Diminutiv auf ja gesetzt.

      set $Ausgabe := $Eingabe & [ Diminutiv: ja ];

    else

      set $Ausgabe := $Eingabe;

      # keine Verkürzung notwendig
```

```
    end if;

    result $Ausgabe, rules Substantiv-Orthographie;

  end case;

end allo-rule; # Verkürze-Substantiv-e-en;


#----------------------------------------------------------------------#

allo-rule Substantiv-Orthographie;

# Diese Regel erzeugt zusätzliche Allomorphe auf ss, wenn ein Substantiv
# auf kurzen Vokal gefolgt von ß endet.
# NB: Diese Regel wird durch die Orthographiereform hinfällig

  start $Eingabe;

  # lies Eingabe ein

  case

  of match $Eingabe.T_OFl = $OFl1:".*" & $OFl2:"'ß" :

    assert match $Eingabe.Basis = $OFl3:".*" & "'" & $OFl4:".*";

    case

    of not $Eingabe.SgDekl = nein:

      case

      of $Eingabe.Genus = Femininum:
      # Beispiel: Nuß. Feminina sind im Sg. i.d.R. unflektiert;
      # es muß also kein weiteres Allomorph erzeugt werden

        $Ausgabe := $Eingabe & [T_OFl: $OFl1 & "ß",
#                               NF: ja,
                                Basis: $OFl3 & $OFl4,
                                PlDekl: nein];

        result $Ausgabe, rules Substantiv-Kasus;

      else

        # erzeuge beide Allomorphe
        $Ausgabe1 := $Eingabe & [T_OFl: $OFl1 & "ß",
#                                NF: ja,
                                 Basis: $OFl3 & $OFl4,
                                 SgDekl: nein,
                                 PlDekl: nein];

        $Ausgabe2 := $Eingabe & [T_OFl: $OFl1 & "ss",
                                 Basis: $OFl3 & $OFl4 ];

        result $Ausgabe1, rules Substantiv-Kasus;
        result $Ausgabe2, rules Substantiv-Kasus;

      end case;

    else        $Ausgabe := $Eingabe & [T_OFl: $OFl1 & "ss",
                                        Basis: $OFl3 & $OFl4 ];

      result $Ausgabe, rules Substantiv-Kasus;

    end case;

  else result $Eingabe, rules Substantiv-Kasus;

  end case;

end allo-rule; # Substantiv-Orthographie

#----------------------------------------------------------------------#

allo-rule Substantiv-Kasus;

  # Diese Regel erzeugt bei Allomorphen, die bereits eine fertig
  # flektierte Wortform darstellen, aus der Genusinformation eine
  # Genus-Kasus-Markierung.

  start $Eingabe;

  $Ausgabe := $Eingabe;

  if not {$Eingabe.SgDekl    = nein and
          $Eingabe.PlDekl    = nein and
          $Eingabe.Diminutiv = ja} then

    include "i_dmm.kasus";
    # Hier wird die Variable $SubstKat definiert. Diese ist eine Liste von
    # Verbunden. Jeder dieser Verbunde besteht aus einem Attribut Stamm (vgl.
    # oben) und einem Attribut KasGen, das die Kasus-Genus-Kombination dar-
    # stellt, für die das Allomorph unflektiert stehen kann.

    choose $StBesch in $SubstKat;
    # für jedes Listenelement in $SubstKat:
```

```
    ? $StBesch.SgDekl ~ $Eingabe.SgDekl;
    ? $Eingabe.PlDekl in $StBesch.PlDekl;

    set $Ausgabe := $Ausgabe & [KasusNumerus: $StBesch.KasusNumerus];

  end if;

  result $Ausgabe, rules Erzeuge-Allo;

end allo-rule; # Substantiv-Kasus


#----------------------------------------------------------------------#
#                                                                      #
#                          Regeln für Adjektive                        #
#                                                                      #
#----------------------------------------------------------------------#

# Diese Regel erzeugt die umgelauteten Adjektivallomorphe. Die umlautenden
# Formen sind bereits im Lexikon markiert (durch Großschreiben des um-
# lautenden Vokals).

# größtenteils selbsterklärend

allo-rule Adjektiv-Start;

  start $Eingabe;

  ? $Eingabe.Klasse = Adjektiv;

  $Umlaut1 := ""; $Umlaut2 := "";
  # Variable vorbelegen;

  case

  of match $Eingabe.LemOFl = $OFl1:".*" & "{" & $StVok:"[aou]" & "}" & $OFl2:".*" :
  # Umlautmarkierungen vorhanden ?

    if  $StVok = "a" then

      set $Umlaut1 := "a"; set $Umlaut2 := "ä";

    elseif $StVok = "o" then

      set $Umlaut1 := "o"; set $Umlaut2 := "ö";

    elseif $StVok = "u" then

      set $Umlaut1 := "u"; set $Umlaut2 := "ü";

    end if;


    $Ausgabe1 := $Eingabe/LemOFl & [T_OFl: $OFl1 & $Umlaut1 & $OFl2,
                                    Basis: $OFl1 & $Umlaut1 & $OFl2,
                                    Komparation: Positiv,
                                    AdjFlex: AdjFlex_ad];
    # AdjFlex_ad steht für _a_dverbiale Verwendung und _d_eklinierbar.
    # Diese Formen sind nicht umgelautet.

    $Ausgabe2 := $Eingabe/LemOFl & [T_OFl: $OFl1 & $Umlaut2 & $OFl2,
                                    Basis: $OFl1 & $Umlaut1 & $OFl2,
                                    AdjFlex: AdjFlex_ks];
    # AdjFlex_ks steht für _K_omparativallo und _S_upelativallo.
    # Diese sind umgelautet.

    result $Ausgabe1, rules Adjektiv-Endung;
    result $Ausgabe2, rules Adjektiv-Endung;

  # keine Umlautung:
  else  $Ausgabe := $Eingabe/LemOFl & [T_OFl: $Eingabe.LemOFl,
                                       Komparation: Positiv,
                                       Basis: $Eingabe.LemOFl ];

    result $Ausgabe, rules Adjektiv-Endung;

  end case;

end allo-rule; # Adjektiv-Umlaut

#----------------------------------------------------------------------#

# Diese Regel nimmt orthographische Korrekturen vor bzw. erzeugt weitere
# Allomorphe, die nicht durch Umlautung entstehen (z.B. heikel - heikle).

allo-rule Adjektiv-Endung;

  start $Eingabe;

  case

  of match $Eingabe.T_OFl = ".*e[nr]" :
  # Bei einem Großteil der Verben auf ".*e[nr]" kann dekliniert und im
  # Komparativ das "e" ausgelassen werden.

    case

    of match $Eingabe.T_OFl = $OFl1:".*" & $OFl2:"qu|ph|[eiw]" & "er" :
    # Adjektive lat.-griech. Ursprungs, die auf der Endsilbe betont sind
```

```
    # (wie z.B. "peripher") sowie heimische Adjektive, bei denen
    # das "e" der Endung zu "ie" oder "ee" gehört, sowie "schwer" und "quer".

    # Bei diesen tritt keine e-Auslassung auf.

      result $Eingabe, rules Erzeuge-Allo;

    of match $Eingabe.T_OFl = $OFl1:".*" & "e" & $Kons:"[nr]" :

      # Beispiel: heiter, trocken -> heit(e)rer, trock(e)ner
      $Ausgabe1 := $Eingabe & [T_OFl: $OFl1 & "e" & $Kons];

      # Die Form "heitre" und "trockne" findet man allerdings nur selten.
      $Ausgabe2 := $Eingabe & [T_OFl: $OFl1 & $Kons,
                               AdjFlex: AdjFlex_dk];

      result $Ausgabe1, rules Adjektiv-Phon-Info;
      result $Ausgabe2, rules Adjektiv-Phon-Info;

    end case;

  of match $Eingabe.T_OFl = $OFl1:".*" & "el":
  # Beispiel: heikel. Hier ist die e-Auslassung beim der Deklination
  # und dem Komparativ der Standardfall

    $Ausgabe1 := $Eingabe & [T_OFl: $OFl1 & "el",
                             AdjFlex: AdjFlex_as];

    $Ausgabe2 := $Eingabe & [T_OFl: $OFl1 & "l",
                             AdjFlex: AdjFlex_dk];

    result $Ausgabe1, rules Adjektiv-Phon-Info;
    result $Ausgabe2, rules Adjektiv-Phon-Info;

  of match $Eingabe.T_OFl = $OFl1:".*" & "'ß" :
  # kurzer Vokal (erkennbar am ') gefolgt von ß; Beispiel:
  # naß, nasses, nasser, nassester

    $Ausgabe1 := $Eingabe & [T_OFl: $OFl1 & "ß",
                             AdjFlex: AdjFlex_a];

    $Ausgabe2 := $Eingabe & [T_OFl: $OFl1 & "ss",
                             AdjFlex: AdjFlex_dks];

    result $Ausgabe1, rules Adjektiv-Phon-Info;
    result $Ausgabe2, rules Adjektiv-Phon-Info;

  of match $Eingabe.T_OFl = $OFl1:".*" & "e" :
  # blöde, feige etc.

    $Ausgabe1 := $Eingabe/HeitSx & [T_OFl: $OFl1 & "e",
                                    AdjFlex: AdjFlex_as];

    $Ausgabe2 := $Eingabe & [T_OFl: $OFl1,
                             AdjFlex: AdjFlex_adks];

    result $Ausgabe1, rules Adjektiv-Phon-Info;
    result $Ausgabe2, rules Adjektiv-Phon-Info;

  else result $Eingabe, rules Adjektiv-Phon-Info;

  end case;

end allo-rule; # Adjektiv-Endung

#-----------------------------------------------------------------------#

allo-rule Adjektiv-Phon-Info;

# Diese Regel identifiziert, welches Superlativsuffix verwendet wird.

start $Eingabe;

  $Ausgabe := $Eingabe;
  $PhonEnd := nil;

  if not PhonEnd in $Eingabe then
  # wenn PhonEnd bereits im Lexikon ist, überspringe diese Regel

    if     match $Eingabe.T_Ofl = ".*"  & "[ßsxz]|haft" then
    # Adjektive, die auf Sibilanten oder auf -haft enden
    # verwenden -est

      set $PhonEnd := e-Einfügung;

    elseif match $Eingabe.T_Ofl = ".*"  & "[dt]|sch|[aeiou][iuh]+" then
    # bei finalem Dental, finalem -sch, Diphtong oder durch h gedehntem
    # Konsonant können sowohl -st als auch -est verwendet werden

      set $PhonEnd := unmarkiert_e-Einfügung;

    else
    # ansonsten wird -st verwendet

      set $PhonEnd := unmarkiert;

    end if;

  else set $PhonEnd := $Eingabe.PhonEnd;
```

```
  end if;

  set $Ausgabe :=  $Ausgabe & [PhonEnd: $PhonEnd];

  result $Ausgabe, rules Adjektiv-heit;

end allo-rule;

#----------------------------------------------------------------------#

# Praktisch alle Adjektive lassen sich mit -heit suffigieren, wobei die
# Allomorphe -heit, -keit und -igkeit existieren (die bis auf wenige
# Ausnahmen, die bereits im Lexikon vermerkt sind, komplementär
# distribuiert sind). Die Wahl des -heit-Suffixes hängt nur von der
# lautlichen Gestalt des Stammes ab.

allo-rule Adjektiv-heit;

  start $Eingabe;

  $HeitSx := <>;

  case

  of {$Eingabe.AdjFlex ~ Ad_Deriv} and {not HeitSx in $Eingabe}:

    if     match $Eingabe.T_Ofl = "[^aeiuoäöü][aeiouäöü][iu][^aeiuoäöü]" then
    # einsilbige Adjektive

      set $HeitSx := <keit>;

    # bestimmte abgeleitete Adjektive
    elseif match $Eingabe.T_Ofl = ".*ig"   then set $HeitSx := <keit>;
    elseif match $Eingabe.T_Ofl = ".*bar"  then set $HeitSx := <keit>;
    elseif match $Eingabe.T_Ofl = ".*isch" then set $HeitSx := <keit>;
    elseif match $Eingabe.T_Ofl = ".*lich" then set $HeitSx := <keit>;
    elseif match $Eingabe.T_Ofl = ".*sam"  then set $HeitSx := <keit>;

    # auf Konsonant + "er" endende Adjektive
    elseif match $Eingabe.T_Ofl = ".*[^eiu]er"   then set $HeitSx := <keit>;

    # -los und -haft immer mit -igkeit
    elseif match $Eingabe.T_Ofl = ".*haft" then set $HeitSx := <igkeit>;
    elseif match $Eingabe.T_Ofl = ".*los"  then set $HeitSx := <igkeit>;

    # andere Fälle
    else                                        set $HeitSx := <heit>;

    end if;

    result $Eingabe & [HeitSx: $HeitSx], rules Erzeuge-Allo;

  else result $Eingabe, rules Erzeuge-Allo;

  end case;

end allo-rule;

#----------------------------------------------------------------------#

# Diese Regel stellt fest, ob ein Name auf einen Sibilanten endet. Diese
# Namen bilden ihren Genitiv durch Suffigierung mit einem Apostroph. Alle
# anderen Namen bilden den Genitiv durch Suffigierung mit -s.

allo-rule Namen;

  start $Eingabe;

  ? $Eingabe.Klasse = Name;

  case

  # Namen, die auf einen Sibilanten enden, bilden den Genitiv durch
  # Anhängen eines Apostrophs (z.B. Lorenz')
  of match $Eingabe.LemOFl = $OFl1:".*" & $OFl2:"[sxzß]":

    $Ausgabe := $Eingabe &
               [T_OFl: $Eingabe.LemOFl,
                Basis: $Eingabe.LemOFl,
                PhonEnd: Sibilant];

    result $Ausgabe, rules Erzeuge-Allo;

  # alle anderen Namen
  else

    $Ausgabe := $Eingabe &
               [T_OFl: $Eingabe.LemOFl,
                Basis: $Eingabe.LemOFl,
                PhonEnd: unmarkiert];

    result $Ausgabe, rules Erzeuge-Allo;

  end case;

end allo-rule; # Namen

#----------------------------------------------------------------------#
```

```
allo-rule Paradigma;

# Bei Morphemen, deren Allomorphbildung unregelmäßig ist (z.B.
# Suppletivformen oder andere Unregelmäßigkeiten), ist das volle
# Paradigma ins Lexikon eingetragen (Lexikondateien pdgm_verben.lex
# und pdgm_adjektive.lex). Ihnen fehlt das Attribut Klasse; stattdessen
# haben sie das Attribut Paradigma, dessen Wert die Wortklasse des
# Morphems ist. Die folgende Regel erzeugt aus diesem Lexikoneintrag
# dessen Allomorphe.

  start $Eingabe;

  ? Paradigma in $Eingabe;

  choose $Wortform in $Eingabe.Wortform ;
  # $Eingabe.Wortform ist die Liste von Allomorphen; erzeuge einen Pfad
  # für jedes Listenelement

  $Ausgabe := [Klasse: $Eingabe.Paradigma,
               T_Ofl: $Wortform.LemOFl,
               Basis: $Eingabe.LemOFl] &
               $Wortform/LemOFl & $Eingabe/LemOFl/Paradigma/Wortform;

  # Erzeugen des Allomorpheintrags; die Attribute der obersten Ebene werden
  # übernommen; die einzelnen Listenelemente enthalten nur die distinktiven
  # Merkmale

  result $Ausgabe, rules Erzeuge-Allo;

end allo-rule;

#-----------------------------------------------------------------------#

# Regel für alle Lexikoneinträge, die nicht durch eine der anderen Regeln
# laufen sollen.

allo-rule Default;

  start $Eingabe;

  if Klasse in $Eingabe then
  # wenn Klasse vorhanden, dann eine der folgenden:
  # (soll fehlerhafte Lexikoneinträge ausfiltern; falls die
  # Wortklasse keine der folgenden ist, kann kein Allomorph
  # erzeugt werden und der Lexikoncompiler gibt eine Fehler-
  # meldung aus

    ? $Eingabe.Klasse in <Artikel, Präp, Pron, Adverb, Interpunkt, Präfix,
                          Präfixoid, Abk, Num, VorgPass, Kon>;

  end if;

  # Wenn Klasse nicht vorhanden ist, muß der Eintrag entweder ein
  # Präfix oder ein Suffix sein; bei Präfixen muß noch das Attribut
  # PxOFl erzeugt werden

  case  of Px in $Eingabe and $Eingabe.Px.OFl = ja:

    $Ausgabe := $Eingabe & [T_OFl: $Eingabe.LemOFl,
                            PxOFl: $Eingabe.LemOFl,
                            Basis: $Eingabe.LemOFl];

    result $Ausgabe, rules Erzeuge-Allo;

  else

    $Basis := $Eingabe.LemOFl;

    if Basis in $Eingabe then set $Basis := $Eingabe.Basis; end if;

    $Ausgabe := $Eingabe & [T_OFl: $Eingabe.LemOFl,
                            Basis: $Basis ];

    result $Ausgabe, rules Erzeuge-Allo;

  end case;

end allo-rule;

#-----------------------------------------------------------------------#

allo-rule Erzeuge-Allo;

  # Dies ist die "End-Regel", in der die Allomorphe erzeugt werden

  start $Allomorph;

  # Lies Allomorph ein

  ? T_OFl in $Allomorph;

  $Oberfläche := $Allomorph.T_OFl;
  $Kategorie  := $Allomorph/LemOFl/T_Ofl/Mater &
                 [Allomorph: $Allomorph.T_OFl];
  $Basis      := $Allomorph.Basis;

  allo $Oberfläche, cat $Kategorie, base $Basis;
```

```
  # erzeuge Allomorph

end allo-rule; # Erzeuge-Allo
```

B.2 Die Datei `i_dmm.ablaut`

```
#----------------------------------------------------------------------#
#                                                                      #
# i_dmm.ablaut                                                         #
# ============                                                         #
#                                                                      #
# Deutsche Malaga-Morphologie, Version 0.1                             #
#                                                                      #
# (c) 1996 Abteilung für Computerlinguistik der Universität            #
# Erlangen-Nürnberg                                                    #
#                                                                      #
# Tabelle der Ablautreihen                                             #
#                                                                      #
# Oliver Lorenz, 29.02.1996                                            #
#                                                                      #
#----------------------------------------------------------------------#

$Ablauttabelle :=
<
[AK: Abl_Norm_a1,AV: <<"a", SF_1, PII>,
<"ä", SF_2>,
<"i", SF_34>>],
[AK: Abl_Norm_a2,AV: <<"a", SF_1, PII>,
<"ä", SF_2>,
<"ie:", SF_34>>],
[AK: Abl_Norm_a3,AV: <<"a", SF_1, PII>,
<"ä", SF_2>,
<"u:", SF_3>,
<"ü:", SF_4>>],
[AK: Abl_Norm_a4,AV: <<"a:", SF_1, PII>,
<"ä:", SF_2>,
<"ie:", SF_34>>],
[AK: Abl_Norm_a5,AV: <<"a:", SF_1, PII>,
<"ä:", SF_2>,
<"u:", SF_3>,
<"ü:", SF_4>>],
[AK: Abl_Norm_au1,AV: <<"au:", SF_1>,
<"äu:", SF_2>,
<"o", SF_3, PII>,
<"ö", SF_4>>],
[AK: Abl_Norm_au2,AV: <<"au:", SF_12>,
<"o:", SF_3, PII>,
<"ö:", SF_4>>],
[AK: Abl_Norm_au3,AV: <<"au:", SF_1, PII>,
<"äu:", SF_2>,
<"ie:", SF_34>>],
[AK: Abl_Norm_e1,AV: <<"e", SF_1>,
<"i", SF_2>,
<"a", SF_3>,
<"ä", SF_4>,
<"o", SF_5, PII>>],
[AK: Abl_Norm_e2,AV: <<"e", SF_1>,
<"i", SF_2>,
<"a", SF_3>,
<"ö", SF_4>,
<"o", SF_5, PII>>],
[AK: Abl_Norm_e3,AV: <<"e", SF_1>,
<"i", SF_2>,
<"a", SF_3>,
<"ü", SF_4>,
<"o", SF_5, PII>>],
[AK: Abl_Norm_e4,AV: <<"e", SF_1>,
<"i", SF_2>,
<"a:", SF_3>,
<"ä:", SF_4>,
<"o", SF_5, PII>>],
[AK: Abl_Norm_e5,AV: <<"e", SF_1>,
<"i", SF_2>,
<"o", SF_3, PII>,
<"ö", SF_4>>],
[AK: Abl_Norm_e6,AV: <<"e", SF_12>,
<"o", SF_3, PII>,
<"ö", SF_4>>],
[AK: Abl_Norm_e7,AV: <<"e", SF_1, PII>,
<"i", SF_2>,
<"a:", SF_3>,
<"ä:", SF_4>>],
[AK: Abl_Norm_e8,AV: <<"e:", SF_1>,
<"ie:", SF_2>,
<"a:", SF_3>,
<"ä:", SF_4>,
<"o:", SF_5, PII>>],
[AK: Abl_Norm_e9,AV: <<"e:", SF_12>,
<"o:", SF_3, PII>,
<"ö:", SF_4>>],
[AK: Abl_Norm_e10,AV: <<"e:", SF_12, PII>,
<"a:", SF_3>,
<"ä:", SF_4>>],
[AK: Abl_Norm_e11,AV: <<"e:", SF_1, PII>,
<"i", SF_2>,
<"a:", SF_3>,
```

```
<"ä:", SF_4>>],
[AK: Abl_Norm_e12,AV: <<"e:", SF_1, PII>,
<"ie:", SF_2>,
<"a:", SF_3>,
<"ä:", SF_4>>],
[AK: Abl_Norm_ei1,AV: <<"ei:", SF_12>,
<"i", SF_34, PII>>],
[AK: Abl_Norm_ei2,AV: <<"ei:", SF_12>,
<"ie:", SF_34, PII>>],
[AK: Abl_Norm_ei3,AV: <<"ei:", SF_12, PII>,
<"ie:", SF_34>>],
[AK: Abl_Norm_i1,AV: <<"i", SF_12>,
<"a", SF_3>,
<"ä", SF_4>,
<"o", SF_5, PII>>],
[AK: Abl_Norm_i2,AV: <<"i", SF_12>,
<"a", SF_3>,
<"ä", SF_4>,
<"u", SF_5, PII>>],
[AK: Abl_Norm_i3,AV: <<"i", SF_12>,
<"a", SF_3>,
<"ö", SF_4>,
<"o", SF_5, PII>>],
[AK: Abl_Norm_i4,AV: <<"i", SF_12>,
<"a:", SF_3>,
<"ä:", SF_4>,
<"e:", SF_5, PII>>],
[AK: Abl_Norm_ie1,AV: <<"ie:", SF_12>,
<"a:", SF_3>,
<"ä:", SF_4>,
<"e:", SF_5, PII>>],
[AK: Abl_Norm_ie2,AV: <<"ie:", SF_12>,
<"o", SF_3, PII>,
<"ö", SF_4>>],
[AK: Abl_Norm_ie3,AV: <<"ie:", SF_12>,
<"o:", SF_3, PII>,
<"ö:", SF_4>>],
[AK: Abl_Norm_o1,AV: <<"o", SF_12, PII>,
<"a:", SF_3>,
<"ä:", SF_4>>],
[AK: Abl_Norm_o2,AV: <<"o:", SF_1, PII>,
<"ö:", SF_2>,
<"ie:", SF_34>>],
[AK: Abl_Norm_u,AV: <<"u:", SF_12, PII>,
<"ie:", SF_34>>],
[AK: Abl_Norm_ä1,AV: <<"ä", SF_12>,
<"i", SF_34>,
<"a", SF_5, PII>>],
[AK: Abl_Norm_ä2,AV: <<"ä:", SF_12>,
<"o:", SF_3, PII>,
<"ö:", SF_4>>],
[AK: Abl_Norm_ö,AV: <<"ö:", SF_12>,
<"o:", SF_3, PII>,
<"ü:", SF_4>>],
[AK: Abl_Norm_ü,AV: <<"ü:", SF_12>,
<"o:", SF_3, PII>,
<"ö:", SF_4>>],
[AK: Abl_Kons_echt,AV: <<"echt", SF_1>,
<"ich", SF_2>,
<"ocht", SF_3, PII>,
<"öcht", SF_4>>],
[AK: Abl_Kons_eh1,AV: <<"e:h", SF_12>,
<"ing", SF_34>,
<"ang", SF_5, PII>>],
[AK: Abl_Kons_eh2,AV: <<"e:h", SF_12>,
<"and", SF_3, PII>,
<"ünd", SF_4>>],
[AK: Abl_Kons_ehm,AV: <<"e:hm", SF_1>,
<"imm", SF_2>,
<"a:hm", SF_3>,
<"ä:hm", SF_4>,
<"omm", SF_5, PII>>],
[AK: Abl_Kons_eid,AV: <<"eid", SF_12>,
<"itt", SF_34, PII>>],
[AK: Abl_Kons_itz,AV: <<"itz", SF_12>,
<"a:ß", SF_3>,
<"ä:ß", SF_4>,
<"ess", SF_5, PII>>],
[AK: Abl_Kons_ieh,AV: <<"ie:h", SF_12>,
<"o:g", SF_3, PII>,
<"ö:g", SF_4>>],
[AK: Abl_Kons_ess,AV: <<"e", SF_1>,
<"i", SF_2>,
<"a:", SF_3>,
<"ä:", SF_4>,
<"ge", SF_5, PII>>],
[AK: Abl_tFlx_enk,AV: <<"enk", SF_12>,
<"ach", SF_34, PII>>],
[AK: Abl_tFlx_ing,AV: <<"ing", SF_12>,
<"ach", SF_34, PII>>],
[AK: Abl_tFlx_enn,AV: <<"enn", SF_12>,
<"ann", SF_34, PII>>],
[AK: Abl_tFlx_ind,AV: <<"ind", SF_12>,
<"und", SF_5, PII>>]
>;
```

B.3 Die Datei `i_dmm.kasus`

```
#                                                                          #
# (c) 1996 Abteilung für Computerlinguistik der Universität                #
# Erlangen-Nürnberg                                                        #
#                                                                          #

$SubstKat := <
[SgDekl: Sg_s,
 PlDekl: <nein, KeinPl, Pl_e_n, Pl_en, Pl_er_n, Pl_n, Pl_s, Pl_ien>,
 KasusNumerus: NomSg+DatSg+AkkSg],
[SgDekl: Sg_s,
 PlDekl: <Pl_0_n>,
 KasusNumerus: NomSg+DatSg+AkkSg+NomPl+GenPl+AkkPl],
[SgDekl: Sg_s,
 PlDekl: <Pl_0>,
 KasusNumerus: NomSg+DatSg+AkkSg+Plural],

[SgDekl: Sg_0,
 PlDekl: <Pl_n, Pl_en, Pl_nen, Pl_s, Pl_e_n, Pl_sen, Pl_se_n, nein, KeinPl>,
 KasusNumerus: Singular],
[SgDekl: Sg_0,
 PlDekl: <Pl_0_n>,
 KasusNumerus: Singular+NomPl+GenPl+AkkPl],
[SgDekl: Sg_0,
 PlDekl: <Pl_0>,
 KasusNumerus: Singular+Plural],

[SgDekl: Sg_e_s,
 PlDekl: <Pl_e_n, Pl_en, Pl_er_n, Pl_ien, Pl_s, Pl_ten, KeinPl, nein>,
 KasusNumerus: NomSg+DatSg+AkkSg],
[SgDekl: Sg_e_s,
 PlDekl: <Pl_0>,
 KasusNumerus: NomSg+DatSg+AkkSg+Plural],

[SgDekl: Sg_es_e,
 PlDekl: <Pl_e_n, Pl_en, Pl_er_n, Pl_se_n, nein, KeinPl>,
 KasusNumerus: Singular],

[SgDekl: Sg_n_s,
 PlDekl: <Pl_n>,
 KasusNumerus: NomSg],

[SgDekl: Sg_n,
 PlDekl: <Pl_n>,
 KasusNumerus: NomSg],

[SgDekl: Sg_en,
 PlDekl: <Pl_en>,
 KasusNumerus: NomSg],

[SgDekl: Sg_en_s,
 PlDekl: <Pl_en>,
 KasusNumerus: NomSg+AkkSg],

[SgDekl: Sg_se_s,
 PlDekl: <Pl_se_n>,
 KasusNumerus: NomSg+DatSg+AkkSg],

[SgDekl: nein,
 PlDekl: <Pl_0_n>,
 KasusNumerus: NomPl+GenPl+AkkPl],
[SgDekl: nein,
 PlDekl: <Pl_0>,
 KasusNumerus: Plural],
[SgDekl: nein,
 PlDekl: <Pl_er_n, Pl_e_n>,
 KasusNumerus: nein],
[SgDekl: KeinSg,
 PlDekl: <Pl_0>,
 KasusNumerus: Plural]
>;
```

Anhang C

Die Kombinationsregeln

C.1 Die Datei dmm.mor

```
#----------------------------------------------------------------------#
#                                                                      #
# dmm.mor                                                              #
# =======                                                              #
#                                                                      #
# Deutsche Malaga-Morphologie, Version 1.0                             #
#                                                                      #
# (c) 1996 Abteilung für Computerlinguistik der Universität            #
# Erlangen-Nürnberg                                                    #
#                                                                      #
# Kombinationsregeln                                                   #
#                                                                      #
# Oliver Lorenz, 29.02.1996                                            #
#                                                                      #
# orlorenz@linguistik.uni-erlangen.de                                  #
#                                                                      #
#----------------------------------------------------------------------#

initial-cat [];
# hinter dem Schlüsselwort "initial-cat" wird die Verbund des leeren
# Satzanfangs angegeben.

initial-rules Startregel;
# hinter dem Schlüsselwort "initial-rules" werden die Regeln angegeben,
# mit denen der leere Satzanfang mit dem ersten Segment verknüpft werden
# soll.

#----------------------------------------------------------------------#
#                                                                      #
#                     Startregel: liest alles ein                      #
#                                                                      #
#----------------------------------------------------------------------#

# Die Startregel liest das erste Segment der Eingabe ein.

combi-rule Startregel;

  generate base;
  # hinter dem Schlüsselwort "generate" werden die Allomorphe angegeben,
  # die zur Generienrung mit dem "paradigm"-Befehl benutzt werden sollen.
  # Wird "base" als Argument übergeben, wird zur Generierung ein Allomorph
  # mit der Grundform des als nächstes gelesenen Segments benutzt.

  start $Links, next $Rechts;
  # Der analysierte Satzanfang wird immer in die Variable $Links gelesen,
  # das zu konkatenierende neue Segment immer in die Variable $Rechts.

  ? not FlexSx in $Rechts;
  # Das Attribut Flex ist nur in Flexionsendungen vorhanden.
  # Eine Flexionsendung kann nicht das erste gelesene Segment sein

  ? not Sx in $Rechts;
  # Das Attribut Sx ist nur in Suffixen vorhanden.
  # Ein Suffix kann nicht das erste gelesene Segment sein

  $Gewicht := 1.0;

  if Gewicht in $Rechts then

    set $Gewicht := $Rechts.Gewicht;

  end if;

  # Wenn kein Gewicht in der Eingabe vorhanden, erzeuge Standardwert 1

  $Ausgabe := $Links & $Rechts &
```

```
             [Gewicht: $Gewicht,
              Wortform: $Rechts.Allomorph,
              Grundform: $Rechts.Basis,
              WortStruktur: <[Typ: WortStart,
                              Basis: $Rechts.Basis,
                              Allomorph: $Rechts.Allomorph,
                              Klasse: $Rechts.Klasse]>];
  # Standard-Verbund einer Wortform

  case

  # Die folgende Fallunterscheidung akzeptiert das erste Segment und verzweigt
  # je nach dessen Wortklasse in Nachfolgeregeln

  of $Rechts.Klasse in < Substantiv, Adjektiv, Verb > :
  # Lexeme

    result $Ausgabe, rules Flexion, Komposition, Suffigierung, Endregel;

  of $Rechts.Klasse = Num:
  # Zahlen

    result $Ausgabe, rules Zahl, Endregel;

  of $Rechts.Klasse = Name:
  # Namen

    result $Ausgabe, rules Flexion, Suffigierung, Endregel;

  of $Rechts.Klasse = Abk:
  # Abkürzungen. Es werden zwei Pfade erzeugt; einer läuft direkt in die
  # Endregel (d.h. es liegt eine fertige Wortform vor); der andere erzeugt
  # Bindestrich-Komposita wie z.B. "ARD-Intendant"

    result $Links & $Rechts, rules Endregel;

    result $Ausgabe/Klasse &
    [ Klasse: Substantiv, Diminutiv: nein, Fuge: <> ],

    rules Suffigierung;

  of Px in $Rechts:
  # Präfixe. Präfixe können verschiedene Wortklassen haben; daher werden
  # sie am Attribut Px erkannt.

    result $Ausgabe, rules Präfigierung, Endregel;

  else
  # alle anderen Segmente sind als unflektierbare, umkomponierbare und
  # underivierbare Partikel zu behandeln und laufen direkt in die
  # Endregel.

    result $Ausgabe, rules Endregel;

  end case;

end combi-rule;

#---------------------------------------------------------------------#
#                                                                     #
#            Flexionsregel: erledigt Flexion mit Flexionssuffix        #
#                                                                     #
#---------------------------------------------------------------------#

# Diese Regel bearbeitet die gesamte Flexion mit Flexionsendungen;
# Flexion ohne Flexionsendung wird in der Endregel bearbeitet.
# Die Regel überprüft, ob die Beschreibung des Stammes ( $Links.Stamm )
# elementweise mit der Beschreibung der Flexionsendung ( $Rechts.FlexSx.Flexion )
# übereinstimmt. Im Moment wedren die Flexionsendungen noch in einem
# Schritt angehängt; in einer späteren Version soll dies eventuell in
# mehreren Schritten geschehen (z.B. "bad/e/t/en").

combi-rule Flexion;

  generate "e","em","en","er","ern","es","est","et",
           "ien","n","nen","ns","s","'","se","sen",
           "ses","st","t";
  # diese Allomorphe sollen zur Generierung benutzt werden.

  start $Links, next $Rechts;

  ? not {Blockieren in $Links and $Links.Blockieren = Flexion};
  # bestimmte Satzanfänge werden für die Flexion blockiert (z.B. der
  # Infinitiv mit zu von abtrennbar-präfigierten Verben wie z.B.
  # "austragen": "auszutrag")

  ? FlexSx in $Rechts;

  # Alle Flexionsendungen enthalten das Attribut FlexSx; ist dies nicht
  # vorhanden, wird dieser Pfad abgebrochen

  choose $FlexEnd in $Rechts.FlexSx;

  # Das Attribut Flex ist eine Liste von kategorialen Interpretationen
  # der Flexionsendung; choose erzeugt für jedes Element dieser Liste
  # einen Pfad. In jedem dieser Pfade liegt jenes Element in der
  # Variable $FlexEnd

  ? $Links.Klasse        = $FlexEnd.StammKlasse;
```

```
# überprüft, ob die Wortklassen von Stamm und Flexionsendung identisch
# sind. Diese Überprüfung ist eigentlich nicht notwendig, da die
# korrekten Kombinationen (und nur diese) auch durch das folgende
# foreach-Statement erkannt wird. Um die Effizienz zu steigern, kann der
# Pfad durch obiges Statement aber schon vorher abgebrochen werden,
# wenn die Klassen von Stamm und Flexion nicht kongruieren.

$Ausgabe := $Links & $FlexEnd/StammKlasse/Flexion;

# Im der foreach-Anweisung wird überprüft, ob die in der Flexionsendung
# spezifizierten Attribute im Stamm vorhanden sind und ob sie mit den
# ebenfalls in der Flexionsendung angegebenen Werten kongruieren. Das
# Ergebnis der Konkatenation ist die Verschmelzung von kategorialer
# Beschreibung des Stammes und der Flexionsendung mit Ausnahme der
# Attribute StammKlasse und Flexion.

foreach $FlexBeschr in $FlexEnd.Flexion:

  $Suche           := $FlexBeschr.1L;
  $Was             := $FlexBeschr.2L;
  $Zugriffsmarker  := $FlexBeschr.3L;

  # $Suche ist das Attribut, das im Stamm vorhanden sein und
  # den Wert $Was haben muß. Bei einigen Flexionsendungen enthält
  # $FlexBeschr noch als drittes Element das Segment exakt. Bei
  # diesen darf nicht auf Kongruenz geprüft werden, sondern es
  # muß auf Identität geprüft werden. Ist das dritte Segment nicht
  # vorhanden, wird $FlexBeschr.3L zu nil.

  ? $Suche in $Links;
  # Attribut vorhanden?

  if $Zugriffsmarker = nil then
  # Kongruenz prüfen

    ? $Links.$Suche ~ $Was;

  elseif $Zugriffsmarker = exakt then
  # Identität prüfen

    ? $Links.$Suche = $Was;


  elseif $Zugriffsmarker = in_Liste then
  # ist $Was ein Element der Liste $Suche ?

    ? $Was in $Links.$Suche;

  else error;
  # Wenn $FlexBeschr ein drittes Element enthält, dies aber nicht
  # das Segment exakt ist, so ist der Lexikoneintrag fehlerhaft.
  # Die Anweisung error bricht die Analyse ab und gibt einen
  # Laufzeitfehler aus.

  end if;

  set $Ausgabe := $Ausgabe/$Suche;
  # Lösche überprüftes Attribut aus der Ausgabe

end foreach;
# Alle von der Flexionsendung geforderten Attribute sind vorhanden
# und haben den richtigen Wert.

# Einige Flexionsendungen müssen blockiert werden, auch wenn obige
# Prüfung erfolgreich war. Diese haben das Attribut Blockieren, das
# eine Liste ist. Diese enthält wiederum Listen, deren erstes
# Element das im Stamm zu suchende Attribut ist; das zweite Element
# ist der Wert, den dieses Attribut *nicht* haben darf. Ist das
# zu untersuchende Attribut eine Liste, muß noch ein Zugriffsmarker
# in_Liste vorhanden sein.

if Blockieren in $FlexEnd then

  foreach $BlockBeschr in $FlexEnd.Blockieren:

    $Suche          := $BlockBeschr.1L;
    $Was            := $BlockBeschr.2L;
    $Zugriffsmarker := $BlockBeschr.3L;

    if $Zugriffsmarker = nil then

      ? not {$Suche in $Links and $Links.$Suche = $Was};

      # wenn Attribut vorhanden und mit bestimmtem Wert belegt,
      # dann Abbruch des Pfades

    elseif $Zugriffsmarker = in_Liste then

      ? not {$Suche in $Links and $Was in $Links.$Suche};

      # wenn Attribut als Liste vorhanden und als Listenelement
      # $Was enthält, dann Abbruch des Pfades

    else error;
    # anderer Zugriffsmarker -> Fehlerhafte Lexikonsyntax

    end if;
```

```
    end foreach;

  end if;

  # alle Pfade, die bis hierher erfolgreich waren, sind korrekte
  # Flexionsformen.

  set $Ausgabe := $Ausgabe &
                  [Wortform: $Links.Wortform & $Rechts.Allomorph,
                   Grundform: $Links.Grundform];

  result $Ausgabe, rules Endregel;

end combi-rule;

#----------------------------------------------------------------------#
#                                                                      #
#                               Komposition                            #
#                                                                      #
#----------------------------------------------------------------------#

# Diese Regel erledigt die Komposition von Lexemen. Die Entscheidung, ob
# eine Komposition zulässig ist, wird (in Ermangelung semantischer
# Information) nur aufgrund morphologisch-syntaktischer Kriterien gefällt.
# Daher generiert und analysiert diese Regel bei lexikalischen Ambiguitäten
# stark über; der Gewichtungsmechanismus erlaubt es allerdings, mit hoher
# Wahrscheinlichkeit der/den korrekten Analyse(n) ein im Vergleich hohes
# Gewicht zu geben.

combi-rule Komposition;

  start $Links, next $Rechts;

  ? Klasse in $Links
    and $Links.Klasse in <Substantiv, Adjektiv, Verb,
                          Komp_Substantiv, Komp_Adjektiv>;
  # Nur Lexeme als Erstglied zulassen;

  ? {$Rechts.Klasse in <Substantiv, Adjektiv, Verb>}
    or
    { Px in $Rechts and $Rechts.Px.Ofl = ja };

  # Die zweite Komponente muß ein Substantiv, Adjektiv oder Verb sein.
  # Die Regel erlaubt auch das Anhängen eines Oberflächenpräfixes, das
  # dann allerdings in der nächsten Regel mit einem Verb oder einem
  # Substantiv kombiniert werden muß. In diesem Fall findet die
  # eigentliche Komposition erst in der Regel Präfigierung statt.

  $Ausgabe := [];
  # Ausgabe-Verbund vordefinieren

  case
  # Fallunterscheidung: Lexem-Präfix oder Lexem-Lexem

  of Px in $Rechts:
  # Lexem-Präfix

    ? $Rechts.Px.OFl = ja and {not Suche in $Rechts.Px};
    # Das Attribut Px.OFl hat den Wert ja, wenn die Präfigierung
    # lexikalisiert ist (also in der Präfixliste eines Verb-
    # oder Nomeneintrags. Andere Präfigierungen ("ge" des
    # Partizip II oder "zu" des Infinitivs mit zu) dürfen
    # nicht zur Komposition zugelassen werden; diese erkennt
    # man am Vorhandensein des Attributs Px.Suche

    $WortStruktur := $Links.Wortstruktur &
                     <[Typ: nil,
                       Basis: $Rechts.Basis,
                       Allomorph: $Rechts.Allomorph,
                       Klasse: $Links.Klasse]>;
    # Im Attribut Wortstruktur werden Kompositions- und
    # Derivationsprozesse dokumentiert.

    set $Ausgabe :=   $Links/WortStruktur & $Rechts &
                      [Wortform: $Links.Wortform & $Rechts.Allomorph,
                       Grundform: $Links.Wortform & $Rechts.Allomorph,
                       WortStruktur: $WortStruktur];

    result $Ausgabe, rules Präfigierung;

  else
  # Lexem-Lexem

    # Fallunterscheidung nach Wortklassen der zweiten Komponente;

    include "i_dmm.kompo";
    # Tabelle der Kompositionsmuster laden; dadurch wird die
    # Variable $Komposition belegt.

    choose $Grundwort in $Komposition;
    # $Komposition ist eine Liste; jedes Listenelement steht für
    # eine Grundwort-Klasse

    ? $Rechts.Klasse = $Grundwort.Klasse;
    # Richtige Klasse des Bestimmungsworts?

    choose $Bestimmungswort in $Grundwort.BW;

    ? $Links.Klasse in $Bestimmungswort.Klasse;
```

```
    foreach $Suchliste in $Bestimmungswort.Suche:

      $Suche          := $Suchliste.1L;
      $Was            := $Suchliste.2L;

      ? $Suche in $Links;
      ? $Links.$Suche ~ $Was;

    end foreach;

    if $Bestimmungswort.Blockieren /= <> then

      $Blockiert := ja;

      foreach $Blockierliste in $Bestimmungswort.Blockieren:

        $Suche          := $Blockierliste.1L;
        $Was            := $Blockierliste.2L;

        if not {$Suche in $Links and $Links.$Suche ~ $Was} then

          set $Blockiert := nein;

        end if;

      end foreach;

      ? not $Blockiert = ja;

    end if;

    set $Ausgabe := $Rechts &
                    [Wortform: $Links.Wortform & $Rechts.Allomorph,
                     Grundform: $Links.Wortform & $Rechts.Basis,
                     Gewicht: $Links.Gewicht * $Bestimmungswort.Gewicht,
                     WortStruktur: $Links.WortStruktur &
                     <[Typ: $Bestimmungswort.Typ,
                       Basis: $Rechts.Basis,
                       Allomorph: $Rechts.Allomorph,
                       Klasse: $Rechts.Klasse]>];

    result $Ausgabe, rules Komposition, Flexion, Suffigierung, Endregel;

  end case;

end combi-rule;

#----------------------------------------------------------------------#
#                                                                      #
#                              Präfixregel                             #
#                                                                      #
#----------------------------------------------------------------------#

# Diese Regel erledigt zum einen lexikalisierte Präfigierungen (d.h. solche,
# die in Verb- und Substantiveinträgen durch die Präfixliste Präfixe kodiert
# sind), zum anderen das Einfügen des Partizip-II-Markers "ge" und das
# Einfügen des Infinitiv-Markers "zu", zum dritten die Präfigierung mit
# Präfixoiden. Lexikalisierte Präfigierungen werden
# über die Präfixoberfläche gesteuert (im Präfixeintrag erkennbar am Attribut
# Px.PxOFl mit dem Wert ja), die anderen über das Attribut Px.Suche im
# Präfixeintrag. Bei Kombinationen von beiden (wie. z.B. "ein/ge/schlaf/en")
# werden beim ersten Durchlauf der Regel das Oberflächenpräfix und der Marker
# kombiniert; im Resultat hat dann Px.PxOFl den Wert ja und auch Px.Suche ist
# vorhanden. Danach wird die Regel ein weiteres Mal durchlaufen. Die
# Kombination über die Oberfläche und die Kombination über Px.Suche erfolgen
# hintereinander und unabhängig. Dadurch kann die Regel ohne komplexe
# Fallunterscheidungen alle zulässigen Kombinationen (ge-Stamm, Px-Stamm,
# Px-ge-Stamm, Px-zu-Stamm, Präfixoid-Stamm) erkennen.

combi-rule Präfigierung;

start $Links, next $Rechts;

  ? Px in $Links;
  # Das Attribut Px ist nur in Präfixeinträgen vorhanden. Andere Wortanfänge
  # dürfen nicht in die Präfixregel laufen.

  $Ausgabe := [];

  case

  of Px in $Rechts and {Suche in $Rechts.Px}:

  # Präfix und Marker

    ? $Links.Px.OFl = ja;
    # Wortanfang muß ein Oberflächenpräfix sein

    ? Suche in $Rechts.Px;
    # nächstes Segment muß einer der beiden Marker sein

    ? not $Rechts.Klasse = Präfixoid;

    set $Ausgabe := $Links/Struktur & $Rechts/Basis/Allomorph &
                    [Klasse: Präfix,
                     Wortform: $Links.Allomorph,
                     Grundform:$Links.Basis,
                     Px: [OFl: ja,
```

```
                         Ergebnis: $Rechts.Px.Ergebnis,
                         Suche: $Rechts.Px.Suche]];

    result $Ausgabe, rules Präfigierung;

  of $Rechts.Klasse in < Verb, Substantiv, OblPxSubstantiv, Adjektiv>:
  # Präfix und Stamm

    if $Links.Px.OFl = ja
    # Oberflächenpräfix: es muß geprüft werden, ob der anzuhängende
    # Stamm eine Präfixliste hat und das Präfix in dieser vorkommt

      then

      ? Präfixe in $Rechts;
      # Präfixliste im Stamm vorhanden?

      choose $Präfix in $Rechts.Präfixe;
      # erzeuge einen Pfad für jedes Präfix in der Präfixliste

      ? $Präfix.PxOFl = $Links.PxOFl;
      # richtiges Präfix?
      # wenn ja:

      $Klasse := $Rechts.Klasse;

      if $Rechts.Klasse = OblPxSubstantiv then set $Klasse := Substantiv; end if;
      # Substantivstämme, die obligatorisch präfigiert werden müssen (z.B.
      # Ab/nahme, An/nahme, Auf/trag) haben die Wortklasse OblPxSubstantiv;
      # nach der Präfigierung können sie wie normale Substantive behandelt
      # werden. $Klasse wird nur in diesem Fall verändert.

      # Falls die Kompositionsregel bei der vorherigen Regelanwendung ein
      # Präfix angehängt hat, muß jetzt noch der Typ der Komposition in
      # das Attribut WortStruktur eingetragen werden. Dazu hat die Kompositions-
      # regel in das Attribut WortStruktur.1R.Klasse die Wortklasse des letzten
      # Lexems eingetragen. Anhand dessen und der Wortklasse des neuen
      # Segments wird nun im folgenden Abschnitt der Kompositionstyp
      # ermittelt und die neue WortStruktur erzeugt.

#       include "i_dmm.kompo";

#       choose $Grundwort in $Komposition;

#       ? $Grundwort.Klasse = $Klasse;

#       choose $Bestimmungswort in $Grundwort.BW;

#       ? $Links.WortStruktur.1R.Klasse in $Bestimmungswort.Klasse;

#       $Typ     := $Bestimmungswort.Typ;
#       $Gewicht := $Bestimmungswort.Gewicht;

      $Typ := WortStart;
      $Gewicht := 0;

      if {$Links.WortStruktur.1R.Klasse in <Substantiv, Komp_Substantiv>
          and $Rechts.Klasse in <Substantiv, Komp_Substantiv>}
        then set $Typ := Subst-Subst-Komp;
             set $Gewicht := 0.9;
      elseif {$Links.WortStruktur.1R.Klasse in <Substantiv, Komp_Substantiv>
              and $Rechts.Klasse in <Verb>}
        then set $Typ := Subst-Verb-Komp;
             set $Gewicht := 0.2;
      elseif {$Links.WortStruktur.1R.Klasse in <Verb>}
        then
        if $Rechts.Klasse in <Substantiv, Komp_Substantiv>
          then set $Typ := Verb-Subst-Komp;
               set $Gewicht := 0.3;
        else fail;
        end if;
      end if;

      $WortStruktur := $Links.WortStruktur/1R &
                       <[Typ: $Typ,
                         Basis: $Links.WortStruktur.1R.Basis & $Rechts.Basis,
                         Allomorph: $Links.WortStruktur.1R.Basis &$Rechts.Allomorph,
                         Klasse: $Klasse]>;

      set $Ausgabe := $Rechts/Präfixe/Klasse & $Präfix/PxOFl &
                      [Klasse: $Klasse,
                       Gewicht: $Links.Gewicht * $Gewicht,
                       Wortform: $Links.Wortform & $Rechts.Allomorph,
                       Grundform: $Links.Wortform & $Rechts.Basis,
                       WortStruktur: $WortStruktur];

    elseif $Links.Klasse = Adverb then fail;
    # "zu", das die Wortklasse Adverb hat, darf nicht vor einen Stamm
    # gehängt werden, wenn nicht vorher ein Oberflächenpräfix vor das
    # "zu" gestellt wurde.
    # "ge" hat eine andere Wortklasse und wird nicht herausgefiltert.

    end if;
    # dieses end if schließt die oberfächengesteuerte Präfigierung ab.

    # Im nächsten Abschnitt wird überprüft, ob auch "ge"/"zu"
    # kombiniert wird (und zwar unabhängig von obiger Überprüfung).

    if Suche in $Links.Px
```

```
    # Am Attribut Px.Suche werden die Marker erkannt

      then

      # Px.Suche ist eine Liste. Deren erstes Element ist der Name des
      # Attributs, das im Stamm gesucht werden soll. Das zweite
      # Element ist ein Zugriffsmarker, der angibt, welche Art von
      # Überprüfung an diesem Attribut erfolgen soll. Das dritte Element
      # gibt den Wert an, mit dem das Attribut überprüft werden soll.
      # Das vierte Element schließlich ist das Ergebnis, falls die
      # Überprüfung positiv ausfällt.

      # (Die gleiche Strategie wird in der Suffixregel verfolgt. Sie erlaubt
      # es, die Regeln strukturell sehr einfach zu halten, da keine Fall-
      # unterscheidung für verschiedene Suffixe gemacht werden muß. Ermöglicht
      # wird dies durch die Verwendung von Konstrukten wie $Rechts.$Suche)
      #                                                           ^

      $Suche          := $Links.Px.Suche.1L;
      $Zugriffsmarker := $Links.Px.Suche.2L;
      $Was            := $Links.Px.Suche.3L;
      $Ergebnis := [];

      if Ergebnis in $Links.Px then

        set $Ergebnis       := $Links.Px.Ergebnis;

      end if;

      if     $Zugriffsmarker = kongr then
      # Wenn der Zugiffsmarker kongr ist, wird einfach geprüft, ob der Wert
      # des Atrributs in $Suche mit $Was kongruiert

        ? $Rechts.$Suche ~ $Was;

      elseif $Zugriffsmarker = ist_in then

        ? $Rechts.$Suche in $Was;

      else fail;

      end if;

      $WortStruktur := <>;

      if $Links.Klasse = Präfixoid then

        set $WortStruktur := <[Typ: WortStart,
                               Basis: $Links.Basis,
                               Allomorph: $Links.Allomorph,
                               Klasse: Präfixoid],
                              [Typ: Präfixoid,
                               Basis: $Rechts.Basis,
                               Allomorph: $Rechts.Allomorph,
                               Klasse: $Rechts.Klasse]>;
      else

        set $WortStruktur := <[Typ: WortStart,
                               Basis: $Links.Basis & $Rechts.Basis,
                               Allomorph: $Links.Allomorph & $Rechts.Allomorph,
                               Klasse: Verb]>;

      end if;

      set $Ausgabe := $Rechts & $Ergebnis &
                      [Gewicht: $Links.Gewicht,
                       Wortform: $Links.Wortform & $Rechts.Allomorph,
                       Grundform:$Links.Wortform & $Rechts.Basis,
                       WortStruktur: $WortStruktur];

    end if;

    result $Ausgabe, rules Komposition, Suffigierung, Flexion, Endregel;

  else fail;

  end case;

end combi-rule;

#----------------------------------------------------------------------#
#                                                                      #
#                  Suffixregel: hauptsächlich Derivation                #
#                                                                      #
#----------------------------------------------------------------------#

# Diese Regel hängt verschiedenste Suffixe and einen Stamm an: Fugen,
# Derivationssuffixe, die Partizip-II-Endugen "en"/"t"/"et" und anderes.
# Die "Intelligenz", die entscheidet, ob ein Suffix an einen Stamm
# angehängt werden darf, steckt nicht in den Regeln, sondern in den
# Suffixeinträgen. Die Regel überprüft nur ob bestimmte Attribute, die
# im Suffixeintrag spezifiziert werden, im Stamm vorhanden sind und ob
# diese bestimmte Werte haben (ähnlich funktioniert ein Teil der Präfix-
# regel). Das einzige Segment, auf das die Regel explizit zugreift, ist
# der Zugriffsmarker.

combi-rule Suffigierung;

  start $Links, next $Rechts;
```

```
? Sx in $Rechts;
# neues Segment mußein Suffix sein

? $Links.Klasse in <Verb, Substantiv, Adjektiv, Name>;
# entweder ein Lexem oder ein Name

#? SgDekl in $Links or FlexTyp in $Links or
#  AdjFlex in $Links or NamKl in $Links;
# entweder ein Lexem oder ein Name

$Ausgabe    := $Links;

choose $Suffix in $Rechts.Sx;
# Das Attribut Sx ist eine Liste von möglichen Interpretationen
# des Suffixes. Für jede Interpretation wird ein Pfad erzeugt,
# das Listenelement ist in $Suffix gespeichert.

# $Suffix ist ein Verbund. $Suffix.Such_L ist eine Liste von
# im Stamm zu suchenden Merkmalen. $Suffix.Lösch_L ist eine
# Liste von Attributen, die aus dem Stamm gelöscht werden
# sollen, wenn die Prüfung positiv ausfällt. $Suffix.Erg ist
# das Ergebnis der erfolgreichen Konkatenation.

$Suchliste  := $Suffix.Suche;
$Löschliste := $Suffix.Lösche;
$Schreibe   := $Suffix.Ergebnis;
$Gewicht    := 1;

if Gewicht in $Suffix then set $Gewicht := $Suffix.Gewicht; end if;

if Blockieren in $Suffix then
  foreach $BlockBeschr in $Suffix.Blockieren:

    $Suche := $BlockBeschr.1L;
    $Was   := $BlockBeschr.2L;

    if $Suche in $Links and $Links.$Suche = $Was
      then fail;
    end if;
  end foreach;
end if;


foreach $Suchtripel in $Suchliste:
# Hier werden sequentiell alle Merkmale in der Suchliste geprüft.
# Alle Prüfungen müssen positiv ausfallen, damit die Regel erfolg-
# reich ist.
# Die Listenelemente in $Suchliste sind wiederum Listen, die aus
# drei Elementen bestehen; das erste ist der Name des zu unter-
# suchenden Attributs im Stamm, das zweite ein Zugriffsmarker
# (siehe Präfixregel) und das dritte der Wert, der mit dem Attribut
# in $Suche verglichen werden soll. Zum Verständnis des folgenden
# Abschnitts siehe Präfixregel.

  $Suche          := $Suchtripel.1L;
  $Zugriffsmarker := $Suchtripel.2L;
  $Wert           := $Suchtripel.3L;

  ? $Suche in $Links;
  # Gesuchtes Attribut vorhanden?

  # ist der Wert des Attributs $Suche in der Liste $Wert enthalten?
  if $Zugriffsmarker = ist_in then ? $Links.$Suche in $Wert;

  # ist $Wert im Attribut $Suche (das eine Liste ist) enthalten?
  elseif $Zugriffsmarker = in_Liste then ? $Wert in $Links.$Suche;

  # Suche nach Feature-Value-Paar
  elseif $Zugriffsmarker = kongr then ? $Links.$Suche ~ $Wert;

  # Attribut enthalten?
  elseif $Zugriffsmarker = nein then  ? $Suche in $Links;

  else fail;

  end if;

end foreach;
# Überprüfung der Suchliste abgeschlossen

set $Ausgabe := $Ausgabe/$Löschliste;
# Die in $Löschliste spezifizierten Attribute aus der Ausgabe löschen

if $Suffix.SxTyp in <nil, Präteritalinfix> then
# Der Suffixtyp ist bei den Fugen nil. In diesem Fall wird nichts an
# der WortStruktur geändert

  set $Ausgabe :=  $Ausgabe & $Schreibe;

# Im anderen Fall wird der Typ der Suffigierung in der WortStruktur
# vermerkt. Dieser steht im Attribut SxTyp.
else set $Ausgabe := $Ausgabe/WortStruktur & $Schreibe &
                     [WortStruktur: $Ausgabe.WortStruktur &
                                    <[Typ: $Suffix.SxTyp,
                                      Basis: $Rechts.Basis,
                                      Allomorph: $Rechts.Allomorph,
                                      Klasse: $Suffix.Ergebnis.Klasse]>];
```

```
  end if;

  if {Blockieren in $Ausgabe and Freigeben in $Schreibe}
  # Falls der Stamm irgendwelche Blockierungen aufweist (vgl. Anfang der
  # Flexionsregel), die am Attribut Blockieren erkennbar sind, und das
  # Suffix Blockierungen wieder freigibt (Attribut Freigeben), geschieht
  # das an dieser Stelle.

    then

    if $Ausgabe.Blockieren = $Schreibe.Freigeben

      then set $Ausgabe := $Ausgabe/Blockieren;

    end if;

  end if;

  $Grundform := $Links.Wortform & $Rechts.Basis;

  if $Suffix.SxTyp = Präteritalinfix then

    set $Grundform := $Links.Grundform;

  end if;

  set $Ausgabe := $Ausgabe/Gewicht &
                  [Wortform: $Links.Wortform & $Rechts.Allomorph,
                   Grundform: $Grundform,
                   Gewicht: $Ausgabe.Gewicht * $Gewicht];

  result $Ausgabe, rules Flexion, Suffigierung, Komposition, Endregel;

end combi-rule;

#----------------------------------------------------------------------#
#                                                                      #
#                              Zahlenregeln                            #
#                                                                      #
#----------------------------------------------------------------------#

# Diese Regeln erkennen Zahlen (in arabischen Ziffern). Wer sich bis hier
# durchgekämpft hat, sollte mit diesen Regeln keinerlei Problem habem.

combi-rule Zahl;

  start $Links, next $Rechts;

  ? $Links.Klasse = Num;
  # Wortanfanf eine Zahl?

  case

  of $Rechts.Klasse = Num:
  # neues Allo eine Zahl?

    result [Klasse: Num ], rules Zahl, Endregel;

  of $Rechts.Klasse = Interpunkt:
  # neues Allo ein Trennzeichen?

    case

    of $Rechts.Ergebnis = Punkt:
    # optisches Trennzeichen oder angelsächsisches "Komma"

      result [Klasse: Num ], rules Nachkommazahl, Endregel;

    of $Rechts.Ergebnis = Komma:
    # Dezimalkomma

      result [Klasse: Num ], rules Nachkommazahl;

    end case;

  end case;

end combi-rule;


combi-rule Nachkommazahl;

  start $Links, next $Rechts;

  ? $Links.Klasse = Num;
  ? $Rechts.Klasse = Num;

  result [Klasse: Num ], rules Zahl, Endregel;

end combi-rule;

#----------------------------------------------------------------------#
#                                                                      #
#                              Endregel                                #
#                                                                      #
#----------------------------------------------------------------------#

# In der Endregel laufen alle bisher erfolgreichen Pfade wieder zusammen.
# In ihr werden kosmetische Operationen vollzogen, also hauptsächlich
```

```
# Kategoriesegmente gelöscht. Es wird aber auch die kategoriale Information
# der ohne Suffix flektierten Lexeme angepaßt.

end-rule Endregel;

  start $Links;

  $Ausgabe := [];

  $Treffer := nein;

  foreach $Kat in < SgDekl, PlDekl, SF, AdjFlex >:

    if $Kat in $Links then set $Treffer := ja; end if;

  end foreach;
  # Ist in der Wortform katgoriale Information vorhanden, die
  # schließen läßt, daß die Wortform unflektiert ist?

  if $Treffer = ja then

    if $Links.Klasse = Adjektiv then

      if PII-Auxil in $Links and not PII-Ge in $Links then
      # Attribut PII-Auxil nur bei PartizipII vorhanden,

        set $Ausgabe := $Links/AdjFlex/Präfixe/PII-Ge/PhonEnd/HeitSx &
                        [Klasse: PII,
                         WortStruktur: $Links.WortStruktur/1R &
                                       <$Links.WortStruktur.1R & [Klasse: PII]>];

      elseif InfZu in $Links then

        set $Ausgabe := $Links/AdjFlex/Präfixe/PII-Ge/PII-Auxil/InfZu &
                          [Klasse: InfZu];

      else ? $Links.AdjFlex ~ AdjFlex_a;
      # ansonsten Adverb, wenn adverbialer Gebrauch möglich

        set $Ausgabe := $Links/AdjFlex & [Klasse: Adverb];

      end if;

    elseif $Links.Klasse = Verb then

      ? Nullflex in $Links;

      set $Ausgabe := $Links/SF/PhonEnd/PII-Ge/PII-Auxil &
                      [Konjugation: <>];

      if Präsens in $Ausgabe.Nullflex then

        set $Ausgabe := $Ausgabe &
                        [Konjugation: $Ausgabe.Konjugation &
                                      <[Subjekt: Sg3,
                                        Tempus: Präsens,
                                        Modus: Indikativ]>];
      end if;

      if Präteritum in $Ausgabe.Nullflex then

        set $Ausgabe := $Ausgabe &
                        [Konjugation: $Ausgabe.Konjugation &
                                      <[Subjekt: Sg13,
                                        Tempus: Präteritum,
                                        Modus: Indikativ]>];
      end if;

      if Imperativ in $Ausgabe.Nullflex then

        set $Ausgabe := $Ausgabe &
                        [Konjugation: $Ausgabe.Konjugation &
                                       <[Subjekt: Sg2,
                                         Tempus: Präsens,
                                         Modus: Imperativ]>];
      end if;

      set $Ausgabe := $Ausgabe/Nullflex;

    elseif $Links.Klasse = Substantiv then

      ?  {$Links.SgDekl in <Sg_0, Sg_e_s, Sg_es_e, Sg_en, Sg_en_s,
                            Sg_n, Sg_n_s, Sg_s, Sg_se_s >}
      or {$Links.PlDekl in <Pl_0, Pl_0_n >};

      # Bei diesen Singular/Plural-Bildungstypen ist Flexion
      # ohne Flexionssuffix möglich

      set $Ausgabe := $Links/SgDekl/PlDekl/Diminutiv;

    elseif $Links.Klasse in <Komp_Substantiv, Komp_Adjektiv> then

      fail;

    else set $Ausgabe := $Links;

    end if;
```

```
  else set $Ausgabe := $Links/Diminutiv/PII-Auxil;

  end if;

  if $Ausgabe.Klasse = Verb then

    $Konjugation := <>;

    if Konjugation in $Ausgabe then

      foreach $SubjektModus in $Ausgabe.Konjugation:

        set $Konjugation := $Konjugation &
                            <[Subjekt: $SubjektModus.Subjekt,
                              Tempus:  $Ausgabe.Tempus,
                              Modus:   $SubjektModus.Modus]>;

        if $SubjektModus.Subjekt = Pl13 and $Ausgabe.Tempus = Präsens and
          $SubjektModus.Modus ~ Indikativ and {not Infinitiv in $Ausgabe} then

          set $Ausgabe := $Ausgabe & [Infinitiv: ja];

        end if;

        if Infinitiv in $Ausgabe and $Ausgabe.Infinitiv = nein then

          set $Ausgabe := $Ausgabe/Infinitiv;

        end if;

      end foreach;

      set $Ausgabe := $Ausgabe/Tempus & [Konjugation: $Konjugation];

    else

      ? Infinitiv in $Ausgabe and $Ausgabe.Infinitiv = ja;

      set $Ausgabe := $Ausgabe/Konjugation;

    end if;

  end if;

  result $Ausgabe/Basis/PII-Ge/Blockieren/HeitSx
                /Px/Fuge/Allomorph/Wortform/Freigeben
                /NullFlex/FlexTyp,
  accept;

end end-rule;
```

C.2 Die Datei `i_dmm.kompo`

```
#----------------------------------------------------------------------#
#                                                                      #
# i_dmm.kompo                                                          #
# ===========                                                          #
#                                                                      #
# Deutsche Malaga-Morphologie, Version 0.1                             #
#                                                                      #
# (c) 1996 Abteilung für Computerlinguistik der Universität            #
# Erlangen-Nürnberg                                                    #
#                                                                      #
# Tabelle der Kompositionsmuster                                       #
#                                                                      #
# Oliver Lorenz, 29.02.1996                                            #
#                                                                      #
#----------------------------------------------------------------------#

$Komposition :=
<[Klasse: Substantiv,
  BW: <[Klasse: <Substantiv, Komp_Substantiv>,
        Gewicht: 0.9,
        Suche: <>,
        Blockieren: <<Diminutiv, ja>,
                     <SgDekl, nein>,
                     <PlDekl, nein>,
                     <Genus, nein>>,
        Typ: Subst-Subst-Komp],
       [Klasse: <Adjektiv, Komp_Adjektiv>,
        Gewicht: 0.7,
        Suche: <<AdjFlex, AdjFlex_d>>,
        Blockieren: <>,
        Typ: Adj-Subst-Komp],
       [Klasse: <Verb>,
        Gewicht: 0.3,
        Suche: <<SF, SF_1_el_er>>,
        Blockieren: <>,
        Typ: Verb-Subst-Komp]>],
 [Klasse: Adjektiv,
  BW: <[Klasse: <Adjektiv, Komp_Adjektiv>,
        Gewicht: 0.5,
        Suche: <<AdjFlex, AdjFlex_d>>,
        Blockieren: <>,
        Typ: Adj-Adj-Komp],
```

```
      [Klasse: <Substantiv, Komp_Substantiv>,
       Gewicht: 0.2,
       Suche: <>,
       Blockieren: <<Diminutiv, ja>,
                    <SgDekl, nein>,
                    <PlDekl, nein>,
                    <Genus, nein>>,
       Typ: Subst-Adj-Komp]>],
[Klasse: Verb,
 BW: <[Klasse: <Adjektiv, Komp_Adjektiv>,
       Gewicht: 0.1,
       Suche: <<AdjFlex, AdjFlex_d>>,
       Blockieren: <>,
       Typ: Adj-Verb-Komp],
      [Klasse: <Substantiv, Komp_Substantiv>,
       Gewicht: 0.2,
       Suche: <>,
       Blockieren: <<Diminutiv, ja>,
                    <SgDekl, nein>,
                    <PlDekl, nein>,
                    <Genus, nein>>,
       Typ: Subst-Verb-Komp]>]>;
```

Anhang D

Die Projektdatei

```
#----------------------------------------------------------------------#
#                                                                      #
# dmm.pro                                                              #
# =======                                                              #
#                                                                      #
# Deutsche Malaga-Morphologie, Version 1.0                             #
#                                                                      #
# (c) 1996 Abteilung für Computerlinguistik der Universität            #
# Erlangen-Nürnberg                                                    #
#                                                                      #
# Projektdatei                                                         #
#                                                                      #
# Oliver Lorenz, 29.02.1996                                            #
#                                                                      #
# orlorenz@linguistik.uni-erlangen.de                                  #
#                                                                      #
#----------------------------------------------------------------------#

# Pfad zur Segmentdatei
seg: rules/dmm.seg

# Pfad zur Allomorphregel-Datei (mit include-Dateien)
all: rules/dmm.all rules/i_dmm.ablaut rules/i_dmm.kasus

# Pfad zu den Lexika
lex: rules/dmm.lex lex/adjektive.lex lex/interpunktion.lex
lex: lex/partikel.lex lex/suffixe.lex lex/flexend.lex lex/verben.lex
lex: lex/substantive.lex lex/abk.lex lex/num.lex lex/paradigma.lex

# Pfad zur Kombinationsregel-Datei
mor: rules/dmm.mor rules/i_dmm.kompo

# Datei für die Disambiguierung
dis: rules/dmm.dis

# Kommando, das bei interaktiven Aufruf von mallex ausgeführt
# werden soll
mallex: hide Präfixe

# Kommando, das bei interaktiven Aufruf von malaga ausgeführt
# werden soll
malaga: hide Präfixe
```

Anhang E

Die Lexika

Vorbemerkung: Aufgrund des großen Umfangs der Lexika für Eigennamen, Verben, Substantive und Adjektive wird hier auf eine vollständige Auflistung dieser Lexika verzichtet. Statt dessen wird bei diesen Wortklassen nur eine repräsentative Auswahl an Lexikoneinträgen dargestellt.

E.1 Verben

```
[LemOFl: "lächeln",
 Klasse: Verb,
 Valenzen: <<PräpErg>, <>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge,
 Präfixe: <[PxOFl: "zu",
            Valenzen: <<Dat>, <>>,
            PII-Auxil:<PII-haben>,
            PII-Ge: Ge],
           [PxOFl: "an",
            Valenzen: <<Akk>, <>>,
            PII-Auxil: <PII-haben>,
            PII-Ge: Ge]>];

[LemOFl: "fr{e}ssen",
 Klasse: Verb,
 AK: Abl_Norm_e7,
 Valenzen: <<Akk>, <PräpErg>, <>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge,
 Präfixe: <[PxOFl: "zusammen",
            Valenzen: <<Akk>>,
            PII-Auxil: <PII-haben>,
            PII-Ge: Ge],
           [PxOFl: "weg",
            Valenzen: <<Akk>, <Akk, DatBen>>,
            PII-Auxil: <PII-haben>,
            PII-Ge: Ge],
           [PxOFl: "kahl",
            Valenzen: <<Akk>>,
            PII-Auxil: <PII-haben>,
            PII-Ge: Ge],
           [PxOFl: "hinein",
            Valenzen: <<Akk>, <Akk, PräpErg>, <PräpErg>, <>>,
            PII-Auxil: <PII-haben>,
            PII-Ge: Ge],
           [PxOFl: "hindurch",
            Valenzen: <<AkkRefl>, <AkkRefl, PräpErg>>,
            PII-Auxil: <PII-haben>,
            PII-Ge: Ge],
           [PxOFl: "heraus",
            Valenzen: <<Akk>, <Akk, DatBen>, <PräpErg>, <>>,
            PII-Auxil: <PII-haben>,
            PII-Ge: Ge],
           [PxOFl: "ein",
            Valenzen: <<AkkRefl>, <AkkRefl, PräpErg>>,
            PII-Auxil: <PII-haben>,
            PII-Ge: Ge],
           [PxOFl: "durch",
            Valenzen: <<Akk>, <AkkRefl, PräpErg>>,
            PII-Auxil: <PII-haben>,
            PII-Ge: Ge, Mater: <h>],
           [PxOFl: "aus",
            Valenzen: <<Akk>>,
            PII-Auxil: <PII-haben>,
            PII-Ge: Ge],
           [PxOFl: "auf",
            Valenzen: <<Akk>>,
```

```
                                                            PII-Auxil: <PII-haben>,
                                            PII-Ge: Ge],
          [PxOFl: "an",
           Valenzen: <<Akk>>,
                                                            PII-Auxil: <PII-haben>,
                                            PII-Ge: Ge],
          [PxOFl: "ab",
           Valenzen: <<Akk>, <Akk, DatBen>>,
                                                            PII-Auxil: <PII-haben>,
                                            PII-Ge: Ge]>];
[LemOFl: "bl{a}sen", Klasse: Verb, AK: Abl_Norm_a4,
 Valenzen: <<Akk>, <>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge, Präfixe: <[PxOFl: "zusammen",
                        Valenzen: <<Akk>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "zurück",
                        Valenzen: <<Akk>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "zu",
                        Valenzen: <<Akk, Dat>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "weg",
                        Valenzen: <<Akk>, <Akk, PräpErg>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "vor",
                        Valenzen: <<Akk>, <Akk, DatBen>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "um",
                        Valenzen: <<Akk>, <Akk, DatBen>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge, Mater: <h>],
                       [PxOFl: "rein",
                        Valenzen: <<Akk>, <Akk, PräpErg>, <Akk, PräpErg, DatBen>, <Akk, DatBen>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "nieder",
                        Valenzen: <<Akk>, <Akk, PräpErg>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "hinweg",
                        Valenzen: <<Akk>, <Akk, PräpErg>, <Akk, PräpErg, DatBen>, <Akk, DatBen>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "hinein",
                        Valenzen: <<Akk>, <Akk, PräpErg>, <PräpErg>, <>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "hindurch",
                        Valenzen: <<PräpErg>, <>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "herein",
                        Valenzen: <<Akk>, <Akk, PräpErg>, <PräpErg>, <>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "fort",
                        Valenzen: <<Akk>, <Akk, DatBen>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "ein",
                        Valenzen: <<Akk, Dat>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "durch",
                        Valenzen: <<Akk>, <Akk, DatBen>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge, Mater: <h>],
                       [PxOFl: "dazwischen",
                        Valenzen: <<PräpErg>, <>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "auseinander",
                        Valenzen: <<Akk>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "aus",
                        Valenzen: <<Akk>, <Akk, DatBen>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "auf",
                        Valenzen: <<Akk>, <Akk, DatBen>, <PräpErg>, <>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "an",
                        Valenzen: <<Akk>, <Akk, DatBen>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "ab",
                        Valenzen: <<Akk>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge]>];
[LemOFl: "zittern",
 Klasse: Verb,
```

```
 Valenzen: <<PräpErg>, <>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge, Präfixe: <[PxOFl: "nach",
                        Valenzen: <<PräpErg>, <>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "an",
                        Valenzen: <<PräpErg>, <>>,
                        PII-Auxil: <PII-sein>,
                        PII-Ge: Ge],
                       [PxOFl: "ab",
                        Valenzen: <<PräpErg>, <>>,
                        PII-Auxil: <PII-sein>,
                        PII-Ge: Ge]>];
[LemOFl: "glauben",
 Klasse: Verb,
 Valenzen: <<Dat>, <Dat, Akk>, <AkkRefl, AdjGS>, <InfZu>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge, Präfixe: <[PxOFl: "tot",
                        Valenzen: <<Akk>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge]>, Mater: <h>];
[LemOFl: "g{e}ben",
 Klasse: Verb,
 AK: Abl_Norm_ei1,
 Valenzen: <<Akk>, <Akk, DatBen>, <Dat, Akk>, <NomGS>, <Dat, InfZu>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge, Präfixe: <[PxOFl: "zusammen",
                        Valenzen: <<Akk>, <Akk, PräpErg>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "zurück",
                        Valenzen: <<Akk>, <Akk, DatBen>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "zu",
                        Valenzen: <<Akk>, <Akk, DatBen>, <InfZu>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge, Mater: <h>],
                       [PxOFl: "zufrieden",
                        Valenzen: <<AkkRefl>, <AkkRefl, PräpErg>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "wieder",
                        Valenzen: <<Akk>, <Akk, DatBen>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "weiter",
                        Valenzen: <<Akk>, <Akk, DatBen>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "weg",
                        Valenzen: <<Akk>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "vor",
                        Valenzen: <<Akk>, <Akk, DatBen>, <InfZu>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge, Mater: <h>],
                       [PxOFl: "statt",
                        Valenzen: <<Dat>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "rum",
                        Valenzen: <<Akk>, <Akk, PräpErg>, <Akk, PräpErg, DatBen>, <Akk, DatBen>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "raus",
                        Valenzen: <<Akk>, <Akk, DatBen>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "preis",
                        Valenzen: <<Akk>, <Akk, DatBen>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "nach",
                        Valenzen: <<Akk, Dat>, <Dat>, <>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "mit",
                        Valenzen: <<Akk, Dat>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "los",
                        Valenzen: <<Akk>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "kund",
                        Valenzen: <<Akk>, <Akk, DatBen>, <PräpErg>, <>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "hinzu",
                        Valenzen: <<Akk>, <Akk, PräpErg>, <Akk, PräpErg, DatBen>, <Akk, DatBen>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
                       [PxOFl: "hin",
                        Valenzen: <<Akk>, <Akk, DatBen>, <Dat>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge],
```

```
[PxOF1: "hinaus",
 Valenzen: <<Akk>, <Akk, PräpErg>, <Akk, PräpErg, DatBen>, <Akk, DatBen>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge],
[PxOF1: "hinab",
 Valenzen: <<Akk>, <Akk, PräpErg>, <Akk, PräpErg, DatBen>, <Akk, DatBen>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge],
[PxOF1: "herum",
 Valenzen: <<Akk>, <Akk, PräpErg>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge],
[PxOF1: "her",
 Valenzen: <<Akk>, <Akk, DatBen>, <PräpErg>, <>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge],
[PxOF1: "heraus",
 Valenzen: <<Akk, Dat>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge],
[PxOF1: "gefangen",
 Valenzen: <<AkkRefl>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge],
[PxOF1: "frei",
 Valenzen: <<Akk>, <Akk, DatBen>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge],
[PxOF1: "ein",
 Valenzen: <<Akk, Dat>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge],
[PxOF1: "durch",
 Valenzen: <<Akk>, <Akk, DatBen>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge],
[PxOF1: "drein",
 Valenzen: <<AkkRefl>, <AkkRefl, PräpErg>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge],
[PxOF1: "drauf",
 Valenzen: <<Akk>, <Akk, DatBen>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge],
[PxOF1: "dran",
 Valenzen: <<Akk>, <Akk, PräpErg>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge],
[PxOF1: "dazu",
 Valenzen: <<Akk>, <Akk, DatBen>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge],
[PxOF1: "darunter",
 Valenzen: <<Akk>, <Akk, PräpErg>, <Akk, PräpErg, DatBen>, <Akk, DatBen>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge],
[PxOF1: "darein",
 Valenzen: <<Akk>, <Akk, PräpErg>, <Akk, PräpErg, DatBen>, <Akk, DatBen>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge],
[PxOF1: "darauf",
 Valenzen: <<Akk>, <Akk, DatBen>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge],
[PxOF1: "daran",
 Valenzen: <<Akk>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge],
[PxOF1: "bloß",
 Valenzen: <<Akk, Dat>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge],
[PxOF1: "bekannt",
 Valenzen: <<Akk>, <Akk, DatBen>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge],
[PxOF1: "bei",
 Valenzen: <<Akk, Dat>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge],
[PxOF1: "aus",
 Valenzen: <<Akk>, <PräpErg>, <>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge],
[PxOF1: "auf",
 Valenzen: <<Akk>, <Akk, DatBen>, <PräpErg>, <>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge],
[PxOF1: "anheim",
 Valenzen: <<Akk, Dat>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge],
[PxOF1: "an",
 Valenzen: <<Akk>, <PräpErg>, <>, <InfZu>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge, Mater: <h>],
[PxOF1: "acht",
 Valenzen: <<PräpErg>, <>>,
 PII-Auxil: <PII-haben>,
```

```
                         PII-Ge: Ge],
                        [PxOFl: "ab",
                         Valenzen: <<Akk>, <Akk, DatBen>, <PräpErg>, <>>,
                         PII-Auxil: <PII-haben>,
                         PII-Ge: Ge]>, Mater: <wieder, h>];
[LemOFl: "sagen",
 Klasse: Verb,
 Valenzen: <<Akk>, <Akk, DatBen>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge, Präfixe: <[PxOFl: "zu",
                         Valenzen: <<Akk, Dat>, <InfZu>>,
                         PII-Auxil: <PII-haben>,
                         PII-Ge: Ge, Mater: <h>],
                        [PxOFl: "wieder",
                         Valenzen: <<Akk, Dat>>,
                         PII-Auxil: <PII-haben>,
                         PII-Ge: Ge],
                        [PxOFl: "weiter",
                         Valenzen: <<Akk>, <Akk, DatBen>>,
                         PII-Auxil: <PII-haben>,
                         PII-Ge: Ge],
                        [PxOFl: "vor",
                         Valenzen: <<Akk, Dat>>,
                         PII-Auxil: <PII-haben>,
                         PII-Ge: Ge],
                        [PxOFl: "vorher",
                         Valenzen: <<Akk>, <Akk, DatBen>>,
                         PII-Auxil: <PII-haben>,
                         PII-Ge: Ge],
                        [PxOFl: "voraus",
                         Valenzen: <<Akk>, <Akk, DatBen>>,
                         PII-Auxil: <PII-haben>,
                         PII-Ge: Ge],
                        [PxOFl: "tot",
                         Valenzen: <<Akk>>,
                         PII-Auxil: <PII-haben>,
                         PII-Ge: Ge],
                        [PxOFl: "nein",
                         PII-Auxil: <PII-haben>,
                         PII-Ge: Ge],
                        [PxOFl: "nach",
                         Valenzen: <<Akk, Dat>>,
                         PII-Auxil: <PII-haben>,
                         PII-Ge: Ge],
                        [PxOFl: "los",
                         Valenzen: <<AkkRefl>, <AkkRefl, PräpErg>>,
                         PII-Auxil: <PII-haben>,
                         PII-Ge: Ge],
                        [PxOFl: "hin",
                         Valenzen: <<Akk>, <Akk, PräpErg>, <PräpErg>, <>>,
                         PII-Auxil: <PII-haben>,
                         PII-Ge: Ge],
                        [PxOFl: "hindurch",
                         Valenzen: <<Akk>, <Akk, PräpErg>, <Akk, PräpErg, DatBen>, <Akk, DatBen>>,
                         PII-Auxil: <PII-haben>,
                         PII-Ge: Ge],
                        [PxOFl: "her",
                         Valenzen: <<Akk>, <Akk, DatBen>>,
                         PII-Auxil: <PII-haben>,
                         PII-Ge: Ge],
                        [PxOFl: "heraus",
                         Valenzen: <<Akk>, <Akk, PräpErg>>,
                         PII-Auxil: <PII-haben>,
                         PII-Ge: Ge],
                        [PxOFl: "gut",
                         Valenzen: <<PräpErg>, <>>,
                         PII-Auxil: <PII-haben>,
                         PII-Ge: Ge],
                        [PxOFl: "durch",
                         Valenzen: <<Akk>, <Akk, DatBen>>,
                         PII-Auxil: <PII-haben>,
                         PII-Ge: Ge],
                        [PxOFl: "dank",
                         Valenzen: <<Dat>>,
                         PII-Auxil: <PII-haben>,
                         PII-Ge: Ge],
                        [PxOFl: "daher",
                         Valenzen: <<Akk>>,
                         PII-Auxil: <PII-haben>,
                         PII-Ge: Ge],
                        [PxOFl: "aus",
                         Valenzen: <<Akk>>,
                         PII-Auxil: <PII-haben>,
                         PII-Ge: Ge],
                        [PxOFl: "auf",
                         Valenzen: <<Akk>, <Akk, DatBen>>,
                         PII-Auxil: <PII-haben>,
                         PII-Ge: Ge],
                        [PxOFl: "an",
                         Valenzen: <<Akk>, <Akk, DatBen>, <PräpErg>, <>>,
                         PII-Auxil: <PII-haben>,
                         PII-Ge: Ge],
                        [PxOFl: "ab",
                         Valenzen: <<Akk>, <Dat>, <InfZu>>,
                         PII-Auxil: <PII-haben>,
                         PII-Ge: Ge, Mater: <h>]>, Mater: <wieder>];
[LemOFl: "schwächen",
 Klasse: Verb,
 Valenzen: <<Akk>, <Akk, DatBen>, <PräpErg>, <>>,
```

```
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge, Präfixe: <[PxOFl: "ab",
                        Valenzen: <<Akk>, <PräpErg>, <>>,
                        PII-Auxil: <PII-haben>,
                        PII-Ge: Ge]>];
[LemOFl: "reisen", Klasse: Verb,
 Valenzen: <<PräpErg>, <>>,
 PII-Auxil: <PII-sein>,
 PII-Ge: Ge, Präfixe: <[PxOFl: "zurück",
                        Valenzen: <<PräpErg>, <>>,
                        PII-Auxil: <PII-sein>,
                        PII-Ge: Ge],
                       [PxOFl: "zu",
                        Valenzen: <<PräpErg>, <>>,
                        PII-Auxil: <PII-sein>,
                        PII-Ge: Ge],
                       [PxOFl: "weiter",
                        Valenzen: <<PräpErg>, <>>,
                        PII-Auxil: <PII-sein>,
                        PII-Ge: Ge],
                       [PxOFl: "weg",
                        Valenzen: <<PräpErg>, <>>,
                        PII-Auxil: <PII-sein>,
                        PII-Ge: Ge],
                       [PxOFl: "voraus",
                        Valenzen: <<PräpErg>, <>>,
                        PII-Auxil: <PII-sein>,
                        PII-Ge: Ge],
                       [PxOFl: "umher",
                        Valenzen: <<PräpErg>, <>>,
                        PII-Auxil: <PII-sein>,
                        PII-Ge: Ge],
                       [PxOFl: "rum",
                        Valenzen: <<PräpErg>, <>>,
                        PII-Auxil: <PII-sein>,
                        PII-Ge: Ge],
                       [PxOFl: "nach",
                        Valenzen: <<Dat>>,
                        PII-Auxil: <PII-sein>,
                        PII-Ge: Ge],
                       [PxOFl: "mit",
                        Valenzen: <<PräpErg>, <>>,
                        PII-Auxil: <PII-sein>,
                        PII-Ge: Ge],
                       [PxOFl: "los",
                        Valenzen: <<PräpErg>, <>>,
                        PII-Auxil: <PII-sein>,
                        PII-Ge: Ge],
                       [PxOFl: "hin",
                        Valenzen: <<PräpErg>, <>>,
                        PII-Auxil: <PII-sein>,
                        PII-Ge: Ge],
                       [PxOFl: "hindurch",
                        Valenzen: <<PräpErg>, <>>,
                        PII-Auxil: <PII-sein>,
                        PII-Ge: Ge],
                       [PxOFl: "herum",
                        Valenzen: <<PräpErg>, <>>,
                        PII-Auxil: <PII-sein>,
                        PII-Ge: Ge],
                       [PxOFl: "her",
                        Valenzen: <<PräpErg>, <>>,
                        PII-Auxil: <PII-sein>,
                        PII-Ge: Ge],
                       [PxOFl: "heim",
                        Valenzen: <<PräpErg>, <>>,
                        PII-Auxil: <PII-sein>,
                        PII-Ge: Ge],
                       [PxOFl: "fort",
                        Valenzen: <<PräpErg>, <>>,
                        PII-Auxil: <PII-sein>,
                        PII-Ge: Ge],
                       [PxOFl: "entgegen",
                        Valenzen: <<Dat>>,
                        PII-Auxil: <PII-sein>,
                        PII-Ge: Ge],
                       [PxOFl: "ein",
                        Valenzen: <<PräpErg>, <>>,
                        PII-Auxil: <PII-sein>,
                        PII-Ge: Ge],
                       [PxOFl: "durch",
                        Valenzen: <<PräpErg>, <>>,
                        PII-Auxil: <PII-haben, PII-sein>,
                        PII-Ge: Ge, Mater: <h>],
                       [PxOFl: "aus",
                        Valenzen: <<PräpErg>, <>>,
                        PII-Auxil: <PII-sein>,
                        PII-Ge: Ge],
                       [PxOFl: "an",
                        Valenzen: <<PräpErg>, <>>,
                        PII-Auxil: <PII-sein>,
                        PII-Ge: Ge],
                       [PxOFl: "ab",
                        Valenzen: <<PräpErg>, <>>,
                        PII-Auxil: <PII-sein>,
                        PII-Ge: Ge]>];
[LemOFl: "beh{e}ben",
 Klasse: Verb,
 AK: Abl_Norm_e9,
```

```
 Valenzen: <<Akk>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge-, Mater: <h>];
[LemOFl: "überstellen",
 Klasse: Verb,
 Valenzen: <<Akk, Dat>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge-];
[LemOFl: "zerhauen",
 Klasse: Verb,
 Valenzen: <<Akk>, <Akk, DatBen>>,
 PII-Auxil: <PII-haben>,
 PII-Ge: Ge-];
```

E.2 Substantive

```
[LemOFl: "absolutism{us}",
 Klasse: Substantiv,
 SgDekl: Sg_0,
 PlDekl: Pl_0,
 Genus: Maskulinum];
[LemOFl: "achse",
 Klasse: Substantiv,
 SgDekl: Sg_0,
 PlDekl: Pl_n,
 Genus: Femininum];
[LemOFl: "alb{um}",
 Klasse: Substantiv,
 SgDekl: Sg_s,
 PlDekl: Pl_0,
 Genus: Neutrum];
[LemOFl: "algebr{a}",
 Klasse: Substantiv,
 SgDekl: Sg_0,
 PlDekl: Pl_0,
 Genus: Femininum];
[LemOFl: "herz",
 Klasse: Substantiv,
 SgDekl: Sg_en_s,
 PlDekl: Pl_en,
 Genus: Neutrum];
[LemOFl: "h{a}us",
 Klasse: Substantiv,
 SgDekl: Sg_es_e,
 PlDekl: Pl_er_n,
 Genus: Neutrum];
[LemOFl: "imbi'ß",
 Klasse: Substantiv,
 SgDekl: Sg_es_e,
 PlDekl: Pl_e_n,
 Genus: Maskulinum];
[LemOFl: "mo{dus}",
 Klasse: Substantiv,
 SgDekl: Sg_0,
 PlDekl: Pl_0,
 Genus: Maskulinum];
[LemOFl: "m{u}nd",
 Klasse: Substantiv,
 SgDekl: Sg_e_s,
 PlDekl: Pl_er_n,
 Genus: Maskulinum];
[LemOFl: "n{o}t",
 Klasse: Substantiv,
 SgDekl: Sg_0,
 PlDekl: Pl_e_n,
 Genus: Femininum];
[LemOFl: "n{u}'ß",
 Klasse: Substantiv,
 SgDekl: Sg_0,
 PlDekl: Pl_e_n,
 Genus: Femininum];
[LemOFl: "tod",
 Klasse: Substantiv,
 SgDekl: Sg_e_s,
 PlDekl: Pl_e_n,
 Genus: Maskulinum];
[LemOFl: "torf",
 Klasse: Substantiv,
 SgDekl: Sg_e_s,
 PlDekl: KeinPl,
 Genus: Maskulinum];
[LemOFl: "uhr",
 Klasse: Substantiv,
 SgDekl: Sg_0,
 PlDekl: Pl_en,
 Genus: Femininum];
[LemOFl: "zeichen",
 Klasse: Substantiv,
 SgDekl: Sg_s,
 PlDekl: Pl_0,
 Genus: Neutrum];
[LemOFl: "{a}bt",
 Klasse: Substantiv,
 SgDekl: Sg_e_s,
 PlDekl: Pl_e_n,
 Genus: Maskulinum];
```

```
[LemOFl: "äquator",
 Klasse: Substantiv,
 SgDekl: Sg_s,
 PlDekl: Pl_en,
 Genus: Maskulinum];
[LemOFl: "ödnis",
 Klasse: Substantiv,
 SgDekl: Sg_0,
 PlDekl: Pl_se_n,
 Genus: Femininum];
```

E.3 Adjektive

```
[LemOFl: "abrupt",
 Klasse: Adjektiv,
 AdjFlex: AdjFlex_adks];
[LemOFl: "alleinig",
 Klasse: Adjektiv,
 AdjFlex: AdjFlex_dk];
[LemOFl: "amerikanisch",
 Klasse: Adjektiv,
 AdjFlex: AdjFlex_adks];
[LemOFl: "ander",
 Klasse: Adjektiv,
 AdjFlex: AdjFlex_d];
[LemOFl: "blutig",
 Klasse: Adjektiv,
 AdjFlex: AdjFlex_adks];
[LemOFl: "duftig",
 Klasse: Adjektiv,
 AdjFlex: AdjFlex_adks];
[LemOFl: "d{u}mm",
 Klasse: Adjektiv,
 AdjFlex: AdjFlex_adks];
[LemOFl: "falsch",
 Klasse: Adjektiv,
 AdjFlex: AdjFlex_adks];
[LemOFl: "gestrig",
 Klasse: Adjektiv,
 AdjFlex: AdjFlex_dk];
[LemOFl: "gr{o}ß",
 Klasse: Adjektiv,
 AdjFlex: AdjFlex_adks,
 PhonEnd: s-Tilgung];
[LemOFl: "heikel",
 Klasse: Adjektiv,
 AdjFlex: AdjFlex_adks];
[LemOFl: "heiter",
 Klasse: Adjektiv,
 AdjFlex: AdjFlex_adks];
[LemOFl: "h{a}rt",
 Klasse: Adjektiv,
 AdjFlex: AdjFlex_adks];
[LemOFl: "j{u}ng",
 Klasse: Adjektiv,
 AdjFlex: AdjFlex_adks];
[LemOFl: "kahl",
 Klasse: Adjektiv,
 AdjFlex: AdjFlex_adks];
[LemOFl: "mäßig",
 Klasse: Adjektiv,
 AdjFlex: AdjFlex_adks];
[LemOFl: "na'ß",
 Klasse: Adjektiv,
 AdjFlex: AdjFlex_adks];
[LemOFl: "neu",
 Klasse: Adjektiv,
 AdjFlex: AdjFlex_adks,
 PhonEnd: e-Einfügung,
 HeitSx: <heit, igkeit>];
[LemOFl: "rauh",
 Klasse: Adjektiv,
 AdjFlex: AdjFlex_adks,
 HeitSx: <heit, igkeit>];
[LemOFl: "schön",
 Klasse: Adjektiv,
 AdjFlex: AdjFlex_adks];
[LemOFl: "übel",
 Klasse: Adjektiv,
 AdjFlex: AdjFlex_adks,
 HeitSx: <keit>];
```

E.4 Eigennamen

```
[LemOFL: "angela",
 Klasse: Name,
 NamKl: VorName,
 Genus: Femininum];
[LemOFL: "armin",
 Klasse: Name,
 NamKl: VorName,
 Genus: Maskulinum];
[LemOFL: "australien",
```

```
 Klasse: Name,
 NamKl: StaatsName];
[LemOFL: "aweritschow",
 Klasse: Name,
 NamKl: NachName];
[LemOFL: "baden-württemberg",
 Klasse: Name,
 NamKl: TopolName];
[LemOFL: "barzel",
 Klasse: Name,
 NamKl: NachName];
[LemOFL: "beer",
 Klasse: Name,
 NamKl: NachName];
[LemOFL: "braunschweig",
 Klasse: Name,
 NamKl: OrtsName];
[LemOFL: "eggebrecht",
 Klasse: Name,
 NamKl: NachName];
[LemOFL: "erfurt",
 Klasse: Name,
 NamKl: OrtsName];
[LemOFL: "erwin",
 Klasse: Name,
 NamKl: VorName,
 Genus: Maskulinum];
[LemOFL: "gabriele",
 Klasse: Name,
 NamKl: VorName,
 Genus: Femininum];
[LemOFL: "geisler",
 Klasse: Name,
 NamKl: NachName];
[LemOFL: "hessen",
 Klasse: Name,
 NamKl: TopolName];
[LemOFL: "jefferson",
 Klasse: Name,
 NamKl: NachName];
[LemOFL: "johansen",
 Klasse: Name,
 NamKl: NachName];
[LemOFL: "jura",
 Klasse: Name,
 NamKl: TopolName];
[LemOFL: "karlsruhe",
 Klasse: Name,
 NamKl: OrtsName];
[LemOFL: "kowalski",
 Klasse: Name,
 NamKl: NachName];
[LemOFL: "paris",
 Klasse: Name,
 NamKl: OrtsName];
[LemOFL: "petra" ,
 Klasse: Name,
 NamKl: VorName,
 Genus: Femininum] ;
[LemOFL: "renate",
 Klasse: Name,
 NamKl: VorName,
 Genus: Femininum];
[LemOFL: "taunus",
 Klasse: Name,
 NamKl: TopolName];
[LemOFL: "waltraud",
 Klasse: Name,
 NamKl: VorName,
 Genus: Femininum];
```

E.5 Flexionsendungen

```
#                                                                   #
# (c) 1995 Abteilung für Computerlinguistik der Universität         #
# Erlangen-Nürnberg                                                 #
#                                                                   #

# Flexionsendungen

[LemOFl: "e",
 FlexSx: <[StammKlasse: Verb,
           Flexion: <<FlexTyp, schwach+stark>,
                     <SF,  SF_1_l_r_er>,
                     <PhonEnd, allePhonEnd>>,
           Blockieren: <<Orthographie, Ortho-ß>>,
           Konjugation: <[Subjekt: Sg1, Modus: Indikativ],
                         [Subjekt: Sg13, Modus: Konjunktiv],
                         [Subjekt: Sg2, Modus: Imperativ]>],
          [StammKlasse: Verb,
           Flexion: <<FlexTyp, schwach>,
                     <SF,  SF_34, exakt>,
                     <PhonEnd, allePhonEnd>>,
           Blockieren: <<Orthographie, Ortho-ß>>,
           Konjugation: <[Subjekt: Sg13, Modus: Indikativ+Konjunktiv]>],
```

```
          [StammKlasse: Verb,
           Flexion: <<FlexTyp, stark>,
                      <SF,  SF_34, exakt>,
                      <PhonEnd, allePhonEnd>>,
           Blockieren: <<Orthographie, Ortho-ß>>,
           Konjugation: <[Subjekt: Sg13, Modus: Konjunktiv]>],
          [StammKlasse: Verb,
           Flexion: <<FlexTyp, stark>,
                      <SF,  SF_4, exakt>,
                      <PhonEnd, allePhonEnd>>,
           Blockieren: <<Orthographie, Ortho-ß>>,
           Konjugation: <[Subjekt: Sg13, Modus: Konjunktiv]>],
          [StammKlasse: Adjektiv,
           Flexion: <<AdjFlex, AdjFlex_d>>,
           Endung: e],
          [StammKlasse: Substantiv,
           Flexion: <<SgDekl, Sg_e_s>,
                      <PlDekl, ANYP-E_N>>,
           KasusNumerus: DatSg],
          [StammKlasse: Substantiv,
           Flexion: <<SgDekl, Sg_es_e>,
                      <PlDekl, ANYP-E_N>>,
           KasusNumerus: DatSg],
          [StammKlasse: Substantiv,
           Flexion: <<SgDekl, SG_D_E>,
                      <PlDekl, Pl_e_n>>,
           KasusNumerus: DatSg+NomPl+GenPl+AkkPl],
          [StammKlasse: Substantiv,
           Flexion: <<SgDekl, SG_D_0>,
                      <PlDekl, Pl_e_n>>,
           KasusNumerus: NomPl+GenPl+AkkPl]>];

[LemOFl: "em",
 FlexSx: <[StammKlasse: Adjektiv,
           Flexion: <<AdjFlex, AdjFlex_d>>,
           Endung: em]>];

[LemOFl: "en",
 FlexSx: <[StammKlasse: Verb,
           Flexion: <<FlexTyp, schwach>,
                      <SF,  SF_13>,
                      <PhonEnd, allePhonEnd>>,
           Blockieren: <<Orthographie, Ortho-ß>>,
           Konjugation: <[Subjekt: Pl13, Modus: Indikativ+Konjunktiv],
                          [Subjekt: Pl1, Modus: Imperativ]>],
          [StammKlasse: Verb,
           Flexion: <<FlexTyp, stark>,
                      <SF,  SF_1>,
                      <PhonEnd, allePhonEnd>>,
           Blockieren: <<Orthographie, Ortho-ß>>,
           Konjugation: <[Subjekt: Pl13, Modus: Indikativ+Konjunktiv],
                          [Subjekt: Sg2+Pl1, Modus: Imperativ]>],
          [StammKlasse: Verb,
           Flexion: <<FlexTyp, stark>,
                      <SF,  SF_3, exakt>,
                      <PhonEnd, allePhonEnd>>,
           Blockieren: <<Orthographie, Ortho-ß>>,
           Konjugation: <[Subjekt: Pl13, Modus: Indikativ]>],
          [StammKlasse: Verb,
           Flexion: <<FlexTyp, stark>,
                      <SF,  SF_4, exakt>,
                      <PhonEnd, allePhonEnd>>,
           Blockieren: <<Orthographie, Ortho-ß>>,
           Konjugation: <[Subjekt: Pl13, Modus: Konjunktiv]>],
          [StammKlasse: Verb,
           Flexion: <<FlexTyp, stark>,
                      <SF,  SF_34, exakt>,
                      <PhonEnd, allePhonEnd>>,
           Blockieren: <<Orthographie, Ortho-ß>>,
           Konjugation: <[Subjekt: Pl13, Modus: Indikativ+Konjunktiv]>],
          [StammKlasse: Adjektiv,
           Flexion: <<AdjFlex, AdjFlex_d>>,
           Endung: en],
          [StammKlasse: Substantiv,
           Flexion: <<SgDekl, Sg_en>,
                      <PlDekl, ANYP-EN>>,
           KasusNumerus: GenSg+DatSg+AkkSg],
          [StammKlasse: Substantiv,
           Flexion: <<SgDekl, ANYSUB>,
                      <PlDekl, Pl_e_n>>,
           KasusNumerus: DatPl],
          [StammKlasse: Substantiv,
           Flexion: <<SgDekl, ANYS-EN>,
                      <PlDekl, Pl_en>>,
           KasusNumerus: Plural],
          [StammKlasse: Substantiv,
           Flexion: <<SgDekl, Sg_en>,
                      <PlDekl, Pl_en>>,
           KasusNumerus: GenSg+DatSg+AkkSg+Plural],
          [StammKlasse: Substantiv,
           Flexion: <<SgDekl, Sg_en_s>,
                      <PlDekl, Pl_en>>,
           KasusNumerus: DatSg+Plural]>];


[LemOFl: "ens",
 FlexSx: <[StammKlasse: Substantiv,
           Flexion: <<SgDekl, Sg_en_s>,
                      <PlDekl, Pl_en>>,
```

```
        KasusNumerus: GenSg]>];

[LemOFl: "er",
 FlexSx: <[StammKlasse: Adjektiv,
          Flexion: <<AdjFlex, AdjFlex_d>>,
          Endung: er],
         [StammKlasse: Substantiv,
          Flexion: <<SgDekl, ANYSUB>,
                   <PlDekl, Pl_er_n>>,
          KasusNumerus: NomPl+GenPl+AkkPl]>];
[LemOFl: "ern",
 FlexSx: <[StammKlasse: Substantiv,
          Flexion: <<SgDekl, ANYSUB>,
                   <PlDekl, Pl_er_n>>,
          KasusNumerus: DatPl]>];
[LemOFl: "es",
 FlexSx: <[StammKlasse: Adjektiv,
          Flexion: <<AdjFlex, AdjFlex_d>>,
          Endung: es],
         [StammKlasse: Substantiv,
          Flexion: <<SgDekl, Sg_e_s>,
                   <PlDekl, ANYSUB>>,
          KasusNumerus: GenSg],
         [StammKlasse: Substantiv,
          Flexion: <<SgDekl, Sg_es_e>,
                   <PlDekl, ANYSUB>>,
          KasusNumerus: GenSg]>];

[LemOFl: "est",
 FlexSx: <[StammKlasse: Verb,
          Flexion: <<FlexTyp, schwach+stark>,
                   <SF,  SF_12, exakt>,
                   <PhonEnd, Dental>>,
          Blockieren: <<Orthographie, Ortho-ß>>,
          Konjugation: <[Subjekt: Sg2, Modus: Indikativ+Konjunktiv]>],
         [StammKlasse: Verb,
          Flexion: <<FlexTyp, stark>,
                   <SF,  SF_1, exakt>,
                   <PhonEnd, allePhonEnd>>,
          Blockieren: <<Orthographie, Ortho-ß>>,
          Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>],
         [StammKlasse: Verb,
          Flexion: <<FlexTyp, stark>,
                   <SF,  SF_12, exakt>,
                   <PhonEnd, nichtDental>>,
          Blockieren: <<Orthographie, Ortho-ß>>,
          Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>],
         [StammKlasse: Verb,
          Flexion: <<FlexTyp, schwach>,
                   <SF,  SF_12_l_er>,
                   <PhonEnd, nichtDental>>,
          Blockieren: <<Orthographie, Ortho-ß>>,
          Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>],
         [StammKlasse: Verb,
          Flexion: <<FlexTyp, stark>,
                   <SF,  SF_4>,
                   <PhonEnd, allePhonEnd>>,
          Blockieren: <<Orthographie, Ortho-ß>>,
          Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>],
         [StammKlasse: Verb,
          Flexion: <<FlexTyp, stark>,
                   <SF,  SF_3, exakt>,
                   <PhonEnd, Dental>>,
          Blockieren: <<Orthographie, Ortho-ß>>,
          Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>],
         [StammKlasse: Verb,
          Flexion: <<FlexTyp, schwach>,
                   <SF,  SF_34>,
                   <PhonEnd, Dental>>,
          Blockieren: <<Orthographie, Ortho-ß>>,
          Konjugation: <[Subjekt: Sg2, Modus: Indikativ+Konjunktiv]>]>];

[LemOFl: "et",
 FlexSx: <[StammKlasse: Verb,
          Flexion: <<FlexTyp, schwach>,
                   <SF,  SF_34>,
                   <PhonEnd, allePhonEnd>>,
          Blockieren: <<Orthographie, Ortho-ß>>,
          Konjugation: <[Subjekt: Pl2, Modus: Indikativ+Konjunktiv]>],
         [StammKlasse: Verb,
          Flexion: <<FlexTyp, stark>,
                   <SF,  SF_34, exakt>,
                   <PhonEnd, allePhonEnd>>,
          Blockieren: <<Orthographie, Ortho-ß>>,
          Konjugation: <[Subjekt: Pl2, Modus: Konjunktiv]>],
         [StammKlasse: Verb,
          Flexion: <<FlexTyp, schwach+stark>,
                   <SF,  SF_1_l_er>,
                   <PhonEnd, nichtDental>>,
          Blockieren: <<Orthographie, Ortho-ß>>,
          Konjugation: <[Subjekt: Pl2, Modus: Konjunktiv]>],
         [StammKlasse: Verb,
          Flexion: <<FlexTyp, stark>,
                   <SF,  SF_3, exakt>,
                   <PhonEnd, Dental>>,
          Blockieren: <<Orthographie, Ortho-ß>>,
          Konjugation: <[Subjekt: Pl2, Modus: Indikativ]>],
         [StammKlasse: Verb,
          Flexion: <<FlexTyp, stark>,
```

```
                     <SF,  SF_4, exakt>,
                     <PhonEnd, allePhonEnd>>,
           Blockieren: <<Orthographie, Ortho-ß>>,
           Konjugation: <[Subjekt: Pl2, Modus: Konjunktiv]>],
          [StammKlasse: Verb,
           Flexion: <<FlexTyp, schwach+stark>,
                     <SF,  SF_12, exakt>,
                     <PhonEnd, Dental>>,
           Blockieren: <<Orthographie, Ortho-ß>>,
           Konjugation: <[Subjekt: Pl2, Modus: Indikativ+Konjunktiv],
                         [Subjekt: Sg3, Modus: Indikativ]>],
          [StammKlasse: Verb,
           Flexion: <<FlexTyp, schwach+stark>,
                     <SF,  SF_1, exakt>,
                     <PhonEnd, Dental>>,
           Blockieren: <<Orthographie, Ortho-ß>>,
           Konjugation: <[Subjekt: Pl2, Modus: Indikativ+Konjunktiv+Imperativ]>]>];

[LemOFl: "ien",
 FlexSx: <[StammKlasse: Substantiv,
           Flexion: <<SgDekl, ANYSUB>,
                     <PlDekl, Pl_ien>>,
           KasusNumerus: Plural]>];

[LemOFl: "n",
 FlexSx: <[StammKlasse: Verb,
           Flexion: <<FlexTyp, schwach>,
                     <SF,  SF_el_er>,
                     <PhonEnd, allePhonEnd>>,
           Konjugation: <[Subjekt: Pl13, Modus: Indikativ+Konjunktiv]>],
          [StammKlasse: Substantiv,
           Flexion: <<SgDekl, Sg_0>,
                     <PlDekl, Pl_n>>,
           KasusNumerus: Plural],
          [StammKlasse: Substantiv,
           Flexion: <<SgDekl, Sg_n_s>,
                     <PlDekl, ANYSUB>>,
           KasusNumerus: DatSg+AkkSg],
          [StammKlasse: Substantiv,
           Flexion: <<SgDekl, ANYSUB>,
                     <PlDekl, Pl_0_n>>,
           KasusNumerus: DatPl],
          [StammKlasse: Substantiv,
           Flexion: <<SgDekl, Sg_n>,
                     <PlDekl, Pl_n>>,
           KasusNumerus: GenSg+DatSg+AkkSg+Plural],
          [StammKlasse: Substantiv,
           Flexion: <<SgDekl, Sg_s>,
                     <PlDekl, Pl_n>>,
           KasusNumerus: Plural]>];

[LemOFl: "nen",
 FlexSx: <[StammKlasse: Substantiv,
           Flexion: <<SgDekl, ANYSUB>,
                     <PlDekl, Pl_nen>>,
           KasusNumerus: Plural]>];

[LemOFl: "ns",
 FlexSx: <[StammKlasse: Substantiv,
           Flexion: <<SgDekl, Sg_n_s>,
                     <PlDekl, ANYSUB>>,
           KasusNumerus: GenSg]>];

[LemOFl: "s",
 FlexSx: <[StammKlasse: Substantiv,
           Flexion: <<SgDekl, Sg_e_s>,
                     <PlDekl, ANYSUB>>,
           KasusNumerus: GenSg],
          [StammKlasse: Substantiv,
           Flexion: <<SgDekl, Sg_s>,
                     <PlDekl, ANYSUB>>,
           KasusNumerus: GenSg],
          [StammKlasse: Substantiv,
           Flexion: <<SgDekl, ANYSUB>,
                     <PlDekl, Pl_s>>,
           KasusNumerus: Plural],
          [StammKlasse: Name,
           Flexion: <<PhonEnd, unmarkiert>>,
           KasusNumerus: GenSg]>];

[LemOFl: "st",
 FlexSx: <[StammKlasse: Verb,
           Flexion: <<FlexTyp, schwach+stark>,
                     <SF,  SF_2_el_er>,
                     <PhonEnd, unmarkiert>>,
           Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>],
          [StammKlasse: Verb,
           Flexion: <<FlexTyp, stark>,
                     <SF,  SF_2, exakt>,
                     <PhonEnd, Dental>,
                     <NullFlex, Präsens, in_Liste>>,
           Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>],
          [StammKlasse: Verb,
           Flexion: <<FlexTyp, stark>,
                     <SF,  SF_3>,
                     <PhonEnd, unmarkiert>>,
           Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>],
          [StammKlasse: Verb,
           Flexion: <<FlexTyp, stark>,
```

```
                     <SF,  SF_4, exakt>,
                     <PhonEnd, unmarkiert>>,
           Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>]>];

[LemOFl: "t",
 FlexSx: <[StammKlasse: Verb,
           Flexion: <<FlexTyp, schwach>,
                     <SF,  SF_12_el_er>,
                     <PhonEnd, unmarkiert>>,
           Blockieren: <<Orthographie, Ortho-ss>>,
           Konjugation: <[Subjekt: Sg3+Pl2, Modus: Indikativ]>],
          [StammKlasse: Verb,
           Flexion: <<FlexTyp, schwach>,
                     <SF,  SF_12>,
                     <PhonEnd, Sibilant>>,
           Blockieren: <<Orthographie, Ortho-ss>>,
           Konjugation: <[Subjekt: Sg23+Pl2, Modus: Indikativ]>],
          [StammKlasse: Verb,
           Flexion: <<FlexTyp, stark>,
                     <SF, SF_1, exakt>,
                     <PhonEnd, nichtDental>>,
           Blockieren: <<Orthographie, Ortho-ss>>,
           Konjugation: <[Subjekt: Pl2, Modus: Indikativ]>],
          [StammKlasse: Verb,
           Flexion: <<FlexTyp, stark>,
                     <SF, SF_2, exakt>,
                     <PhonEnd, nichtSibilant>>,
           Blockieren: <<Orthographie, Ortho-ss>,
                        <NullFlex, Präsens, in_Liste>>,
           Konjugation: <[Subjekt: Sg3, Modus: Indikativ]>],
          [StammKlasse: Verb,
           Flexion: <<FlexTyp, stark>,
                     <SF, SF_2, exakt>,
                     <PhonEnd, Sibilant>>,
           Blockieren: <<Orthographie, Ortho-ss>>,
           Konjugation: <[Subjekt: Sg23, Modus: Indikativ]>],
          [StammKlasse: Verb,
           Flexion: <<FlexTyp, stark>,
                     <SF, SF_12, exakt>,
                     <PhonEnd, unmarkiert>>,
           Blockieren: <<Orthographie, Ortho-ss>>,
           Konjugation: <[Subjekt: Sg3+Pl2, Modus: Indikativ]>],
          [StammKlasse: Verb,
           Flexion: <<FlexTyp, stark>,
                     <SF, SF_3, exakt>,
                     <PhonEnd, unmarkiert>>,
           Blockieren: <<Orthographie, Ortho-ss>>,
           Konjugation: <[Subjekt: Pl2, Modus: Indikativ]>],
          [StammKlasse: Verb,
           Flexion: <<FlexTyp, stark>,
                     <SF, SF_4, exakt>,
                     <PhonEnd, unmarkiert>>,
           Blockieren: <<Orthographie, Ortho-ss>>,
           Konjugation: <[Subjekt: Pl2, Modus: Konjunktiv]>],
          [StammKlasse: Verb,
           Flexion: <<FlexTyp, stark>,
                     <SF, SF_34, exakt>,
                     <PhonEnd, nichtSibilant>>,
           Blockieren: <<Orthographie, Ortho-ss>>,
           Konjugation: <[Subjekt: Pl2, Modus: Indikativ]>],
          [StammKlasse: Verb,
           Flexion: <<FlexTyp, stark>,
                     <SF, SF_34, exakt>,
                     <PhonEnd, Sibilant>>,
           Blockieren: <<Orthographie, Ortho-ss>>,
           Konjugation: <[Subjekt: Sg2+Pl2, Modus: Indikativ]>]>];

[LemOFl: "’",
 FlexSx:  <[StammKlasse: Name,
            Flexion: <<PhonEnd, Sibilant>>,
            KasusNumerus: GenSg]>];

[LemOFl: "se",
 FlexSx: <[StammKlasse: Substantiv,
           Flexion: <<SgDekl, ANYSUB>,
                     <PlDekl, Pl_se_n>>,
           KasusNumerus: NomPl+GenPl+AkkPl]>];

[LemOFl: "sen",
 FlexSx: <[StammKlasse: Substantiv,
           Flexion: <<SgDekl, ANYSUB>,
                     <PlDekl, Pl_se_n>>,
           KasusNumerus: DatPl]>];

[LemOFl: "ses",
 FlexSx: <[StammKlasse: Substantiv,
           Flexion: <<SgDekl, Sg_se_s>,
                     <PlDekl, ANYSUB>>,
           KasusNumerus: GenSg]>];
```

E.6 Suffixe

```
#                                                                     #
# (c) 1995 Abteilung für Computerlinguistik der Universität           #
```

```
# Erlangen-Nürnberg                                                  #
#                                                                    #

# Superlativ- und Komparativ-Infix

[LemOFl: "st",   Sx: <[Suche: <<AdjFlex, kongr, AdjFlex_s>,
                               <PhonEnd, kongr, unmarkiert>>,
                       Lösche: <>,
                       Ergebnis: [AdjFlex: AdjFlex_d,
                                  PhonEnd: nein,
                                  Komparation: Superlativ]]>];

[LemOFl: "t",    Sx: <[Suche: <<AdjFlex, kongr, AdjFlex_s>,
                               <PhonEnd, kongr, s-Tilgung>>,
                       Lösche: <>,
                       Ergebnis: [AdjFlex: AdjFlex_d,
                                  PhonEnd: nein,
                                  Komparation: Superlativ]]>];
[LemOFl: "est",  Sx: <[Suche: <<AdjFlex, kongr, AdjFlex_s>,
                               <PhonEnd, kongr, e-Einfügung>>,
                       Lösche: <>,
                       Ergebnis: [AdjFlex: AdjFlex_d,
                                  PhonEnd: nein,
                                  Komparation: Superlativ]]>];
[LemOFl: "er",   Sx: <[Suche: <<AdjFlex, kongr, AdjFlex_k>>,
                       Lösche: <>,
                       Ergebnis: [AdjFlex: AdjFlex_ad,
                                  PhonEnd: nein,
#                                 Klasse: Adverb,
                                  Komparation: Komparativ]]>];

# Diminutivsuffixe

[LemOFl: "chen", Sx: <[Suche: <<Diminutiv, kongr, ja>>,
                       Lösche: <>,
                       SxTyp: Diminutiv,
                       Ergebnis: [Klasse: Substantiv,
                                  SgDekl: Sg_s,
                                  PlDekl: Pl_0,
                                  Genus: Neutrum,
                                  KasusNumerus: NomSg+DatSg+AkkSg,
                                  Diminutiv: nein]]>];
[LemOFl: "lein", Sx: <[Suche: <<Diminutiv, kongr, ja>>,
                       Lösche: <>,
                       SxTyp: Diminutiv,
                       Ergebnis: [Klasse: Substantiv,
                                  SgDekl: Sg_s,
                                  PlDekl: Pl_0,
                                  Genus: Neutrum,
                                  KasusNumerus: NomSg+DatSg+AkkSg,
                                  Diminutiv: nein]]>];

# Derivationssuffixe

[LemOFl: "ung", Sx: <[Suche: <<SF, kongr, SF_1_l_er>>,
                      Lösche: <PII-Auxil, Valenzen, Präfixe, SF, FlexTyp, PhonEnd, Tempus>,
                      Gewicht: 0.7,
                      SxTyp: Verb-Subst-Deriv,
                      Ergebnis: [Klasse: Substantiv,
                                 SgDekl: Sg_0,
                                 PlDekl: Pl_en,
                                 Genus: Femininum,
                                 KasusNumerus: Singular,
                                 Fuge: <Fuge_s>]]>];

[LemOFl: "bar", Sx: <[Suche: <<SF, kongr, SF_1_el_er>>,
                      Lösche: <PII-Auxil, Valenzen, Präfixe, SF, FlexTyp, PhonEnd, Tempus>,
                      Gewicht: 0.8,
                      SxTyp: Verb-Adj-Deriv,
                      Ergebnis: [Klasse: Adjektiv,
                                 AdjFlex: AdjFlex_adks,
                                 PhonEnd: nein,
                                 HeitSx: <keit>]]>];

[LemOFl: "end", Sx: <[Suche: <<SF, kongr, SF_1>, <PII-Ge, nein, Ge->>,
                      Lösche: <InfZu, Blockieren, SF, FlexTyp, Tempus>,
                      SxTyp: Verb-Adj-Deriv,
                      Ergebnis: [Klasse: Adjektiv,
                                 AdjFlex: AdjFlex_adks,
                                 PhonEnd: nein]]>];

[LemOFl: "end", Sx: <[Suche: <<SF, kongr, SF_el_er>, <PII-Ge, nein, Ge->>,
                      Lösche: <InfZu, Blockieren, SF, FlexTyp, Tempus>,
                      SxTyp: Verb-Adj-Deriv,
                      Ergebnis: [Klasse: Adjektiv,
                                 AdjFlex: AdjFlex_adks,
                                 PhonEnd: nein]]>];

[LemOFl: "er", Sx:  <[Suche: <<SF, kongr, SF_12_l_er>>,
                      Lösche: <PII-Auxil, Valenzen, Präfixe, SF, FlexTyp, PhonEnd, Tempus>,
                      Gewicht: 0.7,
                      SxTyp: Verb-Subst-Deriv,
                      Ergebnis: [Klasse: Substantiv,
                                 SgDekl: Sg_s,
                                 Pldekl: Pl_0_n,
                                 Genus: Maskulinum,
                                 KasusNumerus: NomSg+DatSg+AkkSg+Plural]],
                     [Suche: <<NamKl, kongr, OrtsName>>,
                      Lösche: <>,
```

```
                   SxTyp: Name-Adj-Deriv,
                   Ergebnis: [Klasse: Adjektiv, Endung: ersnm]]>];

[LemOFl: "erin", Sx:<[Suche: <<SF, kongr, SF_12_l_er>>,
                   Lösche: <PII-Auxil, Valenzen, Präfixe, SF, FlexTyp, PhonEnd, Tempus>,
                   Gewicht: 0.7,
                   SxTyp: Verb-Subst-Deriv,
                   Ergebnis: [Klasse: Substantiv,
                             SgDekl: Sg_0,
                             PlDekl: Pl_nen,
                             Genus: Femininum,
                             KasusNumerus: Singular]],
                  [Suche: <<NamKl, kongr, OrtsName>>,
                   Lösche: <>,
                   SxTyp: Name-Adj-Deriv,
                   Ergebnis: [Klasse: Substantiv,
                             SgDekl: Sg_0,
                             PlDekl: Pl_nen,
                             Genus: Femininum,
                             KasusNumerus: Singular]]>];

[LemOFl: "heit", Sx: <[Suche: <<HeitSx, in_Liste, heit>>,
                    Lösche: <HeitSx, PII-Auxil, Valenzen, SF, FlexTyp, PhonEnd, AdjFlex>,
                    SxTyp: Adj-Subst-Deriv,
                    Ergebnis: [Klasse: Substantiv,
                              SgDekl: Sg_0,
                              PlDekl: Pl_en,
                              Genus: Femininum,
                              KasusNumerus: Singular,
                              Fuge: <Fuge_s>]]>];

[LemOFl: "keit", Sx: <[Suche: <<HeitSx, in_Liste, keit>>,
                    Lösche: <HeitSx, PII-Auxil, Valenzen, SF, FlexTyp, PhonEnd>,
                    SxTyp: Adj-Subst-Deriv,
                    Ergebnis: [Klasse: Substantiv,
                              SgDekl: Sg_0,
                              PlDekl: Pl_en,
                              Genus: Femininum,
                              KasusNumerus: Singular,
                              Fuge: <Fuge_s>]]>];
[LemOFl: "igkeit", Sx: <[Suche: <<HeitSx, in_Liste, igkeit>>,
                      Lösche: <HeitSx, PII-Auxil, Valenzen, SF, FlexTyp, PhonEnd>,
                      SxTyp: Adj-Subst-Deriv,
                      Ergebnis: [Klasse: Substantiv,
                                SgDekl: Sg_0,
                                PlDekl: Pl_en,
                                Genus: Femininum,
                                KasusNumerus: Singular,
                                Fuge: <Fuge_s>]]>];

[LemOFl: "mäßig", Sx: <[Suche: <<Klasse, kongr, Substantiv>>,
                      Lösche: <SgDekl, PlDekl, Diminutiv>,
                      SxTyp: Subst-Adj-Deriv,
                      Ergebnis: [Klasse: Adjektiv,
                                AdjFlex: AdjFlex_adks,
                                PhonEnd: unmarkiert,
                                HeitSx: <keit>]]>];

# Fugen

[LemOFl: "-", Sx: <[Suche: <<Klasse, kongr, Komp_Substantiv>>,
                 Gewicht: 0.9,
                 Lösche: <>,
                 SxTyp: nil,
                 Ergebnis: [Klasse: Komp_Substantiv]],
                [Suche: <<Klasse, kongr, Substantiv>>,
                 Gewicht: 0.9,
                 Lösche: <>,
                 SxTyp: nil,
                 Ergebnis: [Klasse: Komp_Substantiv]],
                [Suche: <<Klasse, kongr, Komp_Adjektiv>>,
                 Gewicht: 0.9,
                 Lösche: <>,
                 SxTyp: nil,
                 Ergebnis: [Klasse: Komp_Adjektiv]]>];


[LemOFl: "es", Sx:<[Suche: <<SgDekl, ist_in, <Sg_e_s, Sg_es_e>>>,
                 Gewicht: 0.9,
                 Lösche: <>,
                 SxTyp: nil,
                 Ergebnis: [Klasse: Komp_Substantiv]]>];

[LemOFl: "n", Sx: <[Suche: <<SgDekl, ist_in, <Sg_n_s>>, <PlDekl, ist_in, <Pl_n>>>,
                 Gewicht: 0.9,
                 Lösche: <>,
                 SxTyp: nil,
                 Ergebnis: [Klasse: Komp_Substantiv]]>];

[LemOFl: "s", Sx: <[Suche: <<SgDekl, ist_in, <Sg_s, Sg_e_s>>>,
                 Gewicht: 0.9,
                 Lösche: <>,
                 SxTyp: nil,
                 Ergebnis: [Klasse: Komp_Substantiv]],
                [Suche: <<Fuge, kongr, <Fuge_s>>>,
                 Gewicht: 0.9,
                 Lösche: <>,
                 SxTyp: nil,
                 Ergebnis: [Klasse: Komp_Substantiv]]>];
```

```
[LemOFl: "en", Sx:<[Suche: <<PlDekl, ist_in, <Pl_en>>>,
                    Gewicht: 0.9,
                    Lösche: <>,
                    SxTyp: nil,
                    Ergebnis: [Klasse: Komp_Substantiv]],
                   [Suche: <<Fuge, kongr, <Fuge_en>>>,
                    Gewicht: 0.9,
                    Lösche: <>,
                    SxTyp: nil,
                    Ergebnis: [Klasse: Komp_Substantiv]]>];

[LemOFl: "ens", Sx:<[Suche: <<SgDekl, ist_in, <Sg_en_s>>>,
                     Gewicht: 0.9,
                     Lösche: <>,
                     SxTyp: nil,
                     Ergebnis: [Klasse: Komp_Substantiv]]>];
[LemOFl: "n", Sx:<[Suche: <<PlDekl, ist_in, <Pl_n>>>,
                    Gewicht: 0.9,
                    Lösche: <>,
                    SxTyp: nil,
                    Ergebnis: [Klasse: Komp_Substantiv]]>];
[LemOFl: "er", Sx:<[Suche: <<PlDekl, ist_in, <Pl_er_n>>>,
                    Gewicht: 0.9,
                    Lösche: <>,
                    SxTyp: nil,
                    Ergebnis: [Klasse: Komp_Substantiv]]>];
[LemOFl: "e", Sx: <[Suche: <<PlDekl, ist_in, <Pl_e_n>>>,
                    Gewicht: 0.9,
                    Lösche: <>,
                    SxTyp: nil,
                    Ergebnis: [Klasse: Komp_Substantiv]]>];

# Präteritalinfix

[LemOFl: "t",
 Basis: "t",
 Sx: <[Suche: <<FlexTyp, kongr, schwach>, # schwache Verben
               <SF, kongr, SF_12_el_er>,
               <PhonEnd, kongr, nichtDental>>,
       Blockieren: <<Orthographie, Ortho-ss>>,
       Lösche: <Orthographie>,
       SxTyp: Präteritalinfix,
       Ergebnis: [PhonEnd: dental,
                  SF: SF_34,
                  Tempus: Präteritum]],
      [Suche: <<FlexTyp, kongr, schwach>, # schwach und ablautend
               <SF, kongr, SF_34>,
               <PhonEnd, kongr, nichtDental>>,
       Blockieren: <<Orthographie, Ortho-ss>>,
       Lösche: <>,
       SxTyp: nil,
       Ergebnis: [PhonEnd: dental,
                  SF: SF_34,
                  Tempus: Präteritum]]>];

[LemOFl: "et",
 Basis: "et",
 Sx: <[Suche: <<FlexTyp, kongr, schwach>,
               <SF, kongr, SF_12>,
               <PhonEnd, kongr, Dental>>,
       Lösche: <Orthographie>,
       SxTyp: Präteritalinfix,
     Ergebnis: [PhonEnd: dental,
                SF: SF_34,
                Tempus: Präteritum]]>];

# Partizip-II-Endungen

[LemOFl: "en", Basis: "en",
               Sx: <[Suche: <<FlexTyp, kongr, stark>,
                             <PII-Ge, kongr, Ge->>,
                     Lösche: <PII-Ge, SF, FlexTyp, Präfixe>,
                     SxTyp: PartizipII,
                     Ergebnis: [Klasse: Adjektiv,
                                AdjFlex: AdjFlex_d,
                                PhonEnd: nein,
                                HeitSx: <heit>,
                                Freigeben: Flexion]],
                    [Suche: <<InfZu, kongr, InfZu>>,
                     Lösche: <SF, FlexTyp, PII-Ge, PII-Auxil,Präfixe>,
                     SxTyp: InfinitivZu,
                     Ergebnis: [Klasse: Adjektiv,
                                AdjFlex: AdjFlex_d,
                                PhonEnd: nein]]>];

[LemOFl: "t", Basis: "t",
              Sx: <[Suche: <<FlexTyp, kongr, schwach>,
                            <PhonEnd, kongr, nichtDental>,
                            <PII-Ge, kongr, Ge->>,
                    Lösche: <PII-Ge, SF, FlexTyp, Präfixe>,
                    SxTyp: PartizipII,
                    Ergebnis: [Klasse: Adjektiv,
                               AdjFlex: AdjFlex_d,
                               PhonEnd: nein,
                               HeitSx: <heit>,
                               Freigeben: Flexion]]>];

[LemOFl: "et", Basis: "en",
```

```
            Sx: <[Suche: <<FlexTyp, kongr, schwach>,
                          <PhonEnd, kongr, Dental>,
                          <PII-Ge, kongr, Ge->>,
                  Lösche: <PII-Ge, SF, FlexTyp, Präfixe>,
                  SxTyp: PartizipII,
                  Ergebnis: [Klasse: Adjektiv,
                             AdjFlex: AdjFlex_d,
                             PhonEnd: nein,
                             HeitSx: <heit>,
                             Freigeben: Flexion]]>];
```

E.7 Partikel

```
# Vorgangspassiv

[LemOFL: "worden", Klasse:  VorgPass, Ergebnis: VorgPass];

# Adverbien

# einfache Adverbien
[LemOFL: "abermalig",       Klasse: Adverb, Ergebnis: Adv];
[LemOFL: "allein",          Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "allerdings",      Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "allerfrühestens", Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "allerorts",       Klasse: Adverb, Ergebnis: AdvL];
[LemOFL: "allerspätestens", Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "alsbald",         Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "also",            Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "anders",          Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "auch",            Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "aufeinander",     Klasse: Adverb, Ergebnis: AdvL];
[LemOFL: "auseinander",     Klasse: Adverb, Ergebnis: AdvD];
[LemOFL: "bald",            Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "baldmöglichst",   Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "barfuß",          Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "barsch",          Klasse: Adverb, Ergebnis: Adv];
[LemOFL: "beieinander",     Klasse: Adverb, Ergebnis: AdvD];
[LemOFL: "beinahe",         Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "beisammen",       Klasse: Adverb, Ergebnis: AdvL];
[LemOFL: "bereit",          Klasse: Adverb, Ergebnis: Adv];
[LemOFL: "bereits",         Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "bis",             Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "bisher",          Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "bislang",         Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "da",              Klasse: Adverb, Ergebnis: AdvL];
[LemOFL: "dabei",           Klasse: Adverb, Ergebnis: AdvL];
[LemOFL: "dadurch",         Klasse: Adverb, Ergebnis: AdvK];
[LemOFL: "dafür",           Klasse: Adverb, Ergebnis: AdvK];
[LemOFL: "dagegen",         Klasse: Adverb, Ergebnis: AdvK];
[LemOFL: "daheim",          Klasse: Adverb, Ergebnis: AdvL];
[LemOFL: "daher",           Klasse: Adverb, Ergebnis: AdvK];
[LemOFL: "dahin",           Klasse: Adverb, Ergebnis: AdvD];
[LemOFL: "damals",          Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "damit",           Klasse: Adverb, Ergebnis: AdvK];
[LemOFL: "danach",          Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "dann",            Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "daran",           Klasse: Adverb, Ergebnis: AdvL];
[LemOFL: "darauf",          Klasse: Adverb, Ergebnis: AdvL];
[LemOFL: "daraus",          Klasse: Adverb, Ergebnis: AdvD];
[LemOFL: "darin",           Klasse: Adverb, Ergebnis: AdvL];
[LemOFL: "darum",           Klasse: Adverb, Ergebnis: AdvK];
[LemOFL: "darunter",        Klasse: Adverb, Ergebnis: AdvL];
[LemOFL: "darüber",         Klasse: Adverb, Ergebnis: AdvL];
[LemOFL: "dazu",            Klasse: Adverb, Ergebnis: AdvK];
[LemOFL: "dazwischen",      Klasse: Adverb, Ergebnis: AdvD];
[LemOFL: "demgemäß",        Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "demnach",         Klasse: Adverb, Ergebnis: AdvK];
[LemOFL: "derzeit",         Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "deshalb",         Klasse: Adverb, Ergebnis: AdvK];
[LemOFL: "diesmal",         Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "doch",            Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "dort",            Klasse: Adverb, Ergebnis: AdvL];
[LemOFL: "dorthin",         Klasse: Adverb, Ergebnis: AdvD];
[LemOFL: "draußen",         Klasse: Adverb, Ergebnis: AdvL];
[LemOFL: "drüben",          Klasse: Adverb, Ergebnis: AdvL];
[LemOFL: "durchaus",        Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "durchweg",        Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "ebenfalls",       Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "ebenso",          Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "eher",            Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "einerseits",      Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "eingehend",       Klasse: Adverb, Ergebnis: Adv];
[LemOFL: "einigermaßen",    Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "einmal",          Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "einstellig",      Klasse: Adverb, Ergebnis: Adv];
[LemOFL: "erstens",         Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "erstmals",        Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "extra",           Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "fast",            Klasse: Adverb, Ergebnis: AdvM];
#[LemOFL: "früher",         Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "frühmorgens",     Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "fünfmal",         Klasse: Adverb, Ergebnis: AdvM];
```

```
[LemOFL: "gar",             Klasse: Adverb, Ergebnis: Adv];
[LemOFL: "garnicht",        Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "genauso",         Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "genug",           Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "geradezu",        Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "gern",            Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "gerne",           Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "gestern",         Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "gleichermaßen",   Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "her",             Klasse: Adverb, Ergebnis: AdvD];
[LemOFL: "heraus",          Klasse: Adverb, Ergebnis: AdvD];
[LemOFL: "hervor",          Klasse: Adverb, Ergebnis: AdvD];
[LemOFL: "heute",           Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "heutzutage",      Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "hier",            Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "hierauf",         Klasse: Adverb, Ergebnis: AdvL];
[LemOFL: "hieraus",         Klasse: Adverb, Ergebnis: AdvD];
[LemOFL: "hierdurch",       Klasse: Adverb, Ergebnis: AdvD];
[LemOFL: "hierher",         Klasse: Adverb, Ergebnis: AdvD];
[LemOFL: "hierher",         Klasse: Adverb, Ergebnis: AdvD];
[LemOFL: "hierzu",          Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "hierzulande",     Klasse: Adverb, Ergebnis: AdvL];
[LemOFL: "hierüber",        Klasse: Adverb, Ergebnis: AdvD];
[LemOFL: "hin",             Klasse: Adverb, Ergebnis: AdvD];
[LemOFL: "hinaus",          Klasse: Adverb, Ergebnis: AdvD];
[LemOFL: "hinweg",          Klasse: Adverb, Ergebnis: AdvD];
[LemOFL: "hoch",            Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "höchstens",       Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "ihretwegen",      Klasse: Adverb, Ergebnis: AdvK];
[LemOFL: "immer",           Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "immerhin",        Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "indes",           Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "insbesondere",    Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "insgesamt",       Klasse: Adverb, Ergebnis: Adv];
[LemOFL: "insoweit",        Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "inzwischen",      Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "irgendwann",      Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "irgendwie",       Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "irgendwo",        Klasse: Adverb, Ergebnis: AdvL];
[LemOFL: "jedenfalls",      Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "jederzeit",       Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "jedesmal",        Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "jetzt",           Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "jüngst",          Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "kaum",            Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "leider",          Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "lieber",          Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "links",           Klasse: Adverb, Ergebnis: AdvL];
[LemOFL: "manchmal",        Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "mehr",            Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "meist",           Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "meistens",        Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "mindestens",      Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "möglicherweise",  Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "möglichst",       Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "nachher",         Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "nachts",          Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "nebenan",         Klasse: Adverb, Ergebnis: AdvL];
[LemOFL: "nicht",           Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "nie",             Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "niemals",         Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "nochmals",        Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "nun",             Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "nunmehr",         Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "nämlich",         Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "oben",            Klasse: Adverb, Ergebnis: AdvL];
[LemOFL: "obenan",          Klasse: Adverb, Ergebnis: AdvL];
[LemOFL: "oft",             Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "ohnegleichen",    Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "ohnehin",         Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "pianissimo",      Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "schon",           Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "seitdem",         Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "seither",         Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "selber",          Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "selbst",          Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "selten",          Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "sofort",          Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "solange",         Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "solo",            Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "sonst",           Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "spät",            Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "spätabends",      Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "spätestens",      Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "stets",           Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "tagsüber",        Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "trotzdem",        Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "ungern",          Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "unterdessen",     Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "vergebens",       Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "vielleicht",      Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "vorbei",          Klasse: Adverb, Ergebnis: AdvD];
[LemOFL: "vorgestern",      Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "vorher",          Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "vormittags",      Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "vorn",            Klasse: Adverb, Ergebnis: AdvL];
[LemOFL: "vorvorgestern",   Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "während",         Klasse: Kon,    Ergebnis: SubKon];
[LemOFL: "weiland",         Klasse: Adverb, Ergebnis: AdvT];
```

```
[LemOFL: "weiterhin",       Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "weitgehend",      Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "wenigstens",      Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "wieder",          Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "wiederum",        Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "ziemlich",        Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "zuerst",          Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "zugleich",        Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "zuletzt",         Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "zumeist",         Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "zunächst",        Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "zurück",          Klasse: Adverb, Ergebnis: AdvD];
[LemOFL: "zuvor",           Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "zuweilen",        Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "zwar",            Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "zwischendurch",   Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "überhaupt",       Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "übermorgen",      Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "überübermorgen",  Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "übrigens",        Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "alleine", Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "allemal", Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "allenfalls", Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "anderswo", Klasse :Adverb, Ergebnis: AdvL];
[LemOFL: "ansonsten", Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "daraufhin", Klasse: Adverb, Ergebnis: AdvK];
[LemOFL: "egal", Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "fortan", Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "hingegen", Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "hinten", Klasse: Adverb, Ergebnis: AdvL];
[LemOFL: "innen", Klasse: Adverb, Ergebnis: AdvL];
[LemOFL: "keinesfalls", Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "keineswegs", Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "mittlerweile", Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "nachmittags", Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "seinerzeit", Klasse: Adverb, Ergebnis: AdvT];
[LemOFL: "somit", Klasse: Adverb, Ergebnis: AdvK];
[LemOFL: "sowieso", Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "womöglich", Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "zudem", Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "zumindest", Klasse: Adverb, Ergebnis: AdvM];
[LemOFL: "öfter", Klasse: Adverb, Ergebnis: AdvM];

#Adverbien modifizierende Adverbien (Abtönungspartikel)
[LemOFL: "quasi", Klasse: Adverb,
 Struktur: <[Val: Adverb, Ergebnis: AdvM]>];
[LemOFL: "allzu",     Klasse: Adverb,
 Struktur: <[Val: Adverb, Ergebnis: AdvM]>];
[LemOFL: "besonders", Klasse: Adverb,
 Struktur: <[Val: Adverb, Ergebnis: AdvM]>];
[LemOFL: "bißchen",   Klasse: Adverb,
 Struktur: <[Val: Adverb, Ergebnis: AdvM]>];
[LemOFL: "durchwegs", Klasse: Adverb,
 Struktur: <[Val: Adverb, Ergebnis: AdvM]>];
[LemOFL: "etwa",      Klasse: Adverb,
 Struktur: <[Val: Adverb, Ergebnis: AdvM]>];
[LemOFL: "nahezu",    Klasse: Adverb,
 Struktur: <[Val: Adverb, Ergebnis: AdvM]>];
[LemOFL: "noch",      Klasse: Adverb,
 Struktur: <[Val: Adverb, Ergebnis: AdvM]>];
[LemOFL: "nur",       Klasse: Adverb,
 Struktur: <[Val: Adverb, Ergebnis: AdvM]>, Ergebnis: AdvM];
[LemOFL: "sehr",      Klasse: Adverb,
 Struktur: <[Val: Adverb, Ergebnis: AdvM]>, Ergebnis: AdvM];
[LemOFL: "so",        Klasse: Adverb,
 Struktur: <[Val: Adverb, Ergebnis: AdvM]>, Ergebnis: AdvM];
[LemOFL: "ungefähr",  Klasse: Adverb,
 Struktur: <[Val: Adverb, Ergebnis: AdvM]>, Ergebnis: AdvM];
[LemOFL: "weitaus",   Klasse: Adverb,
 Struktur: <[Val: Adverb, Ergebnis: AdvM]>];
[LemOFL: "ziemlich",  Klasse: Adverb,
 Struktur: <[Val: Adverb, Ergebnis: AdvM]>];
[LemOFL: "zirka",     Klasse: Adverb,
 Struktur: <[Val: Adverb, Ergebnis: AdvM]>];
[LemOFL: "zu",        Klasse: Adverb,
                      Px: [OFl: nein,
                           Suche: <SF, kongr, SF_1_el_er>,
                           Ergebnis: [InfZu: InfZu,
                                      Blockieren: Flexion,
                                      Struktur: <[Val: Adverb, Ergebnis: AdvM],
                                                 [Val: Dat, Ergebnis: AdvD]>]]];
[LemOFL: "überaus",   Klasse: Adverb, Struktur: <[Val: Adverb, Ergebnis: AdvM]>];


#Determinativa

#Determinativa
#hier noch ein Stück von Dir modifizierte, korrigierte Partikel

[LemOFL: "alle", Klasse: Artikel, Struktur: <
 [Adjektiv: e,  Genus: Kein_Genus, KasusNumerus: AkkPl, Ergebnis: AkkSg],
 [Adjektiv: en, Genus: Kein_Genus, KasusNumerus: NomPl+AkkPl, Ergebnis: Pl3+Akk]>,
 Ergebnis: Pl3+Akk];

[LemOFL: "allen", Klasse: Artikel, Struktur: <
 [Adjektiv: en, Genus: Kein_Genus, KasusNumerus: DatPl, Ergebnis: Dat]>,
 Ergebnis: Dat];

[LemOFL: "aller", Klasse: Artikel, Struktur: <
```

```
 [Adjektiv: EN, Genus: Kein_Genus, KasusNumerus: GenPl, Ergebnis: Gen]>,
 Ergebnis: Gen];

[LemOFL: "allerlei", Klasse: Artikel, Struktur: <
 [Adjektiv: E,  Genus: Kein_Genus, KasusNumerus: NomPl+AkkPl, Ergebnis: Pl3+Akk],
 [Adjektiv: EN, Genus: Kein_Genus, KasusNumerus: DatPl, Ergebnis: Dat],
 [Adjektiv: ER, Genus: Kein_Genus, KasusNumerus: GenPl, Ergebnis: Gen]>,
 Ergebnis: Pl3+Gen+Dat+Akk];

[LemOFL: "am",   Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: DatSg,
             Ergebnis: AdvL]>];
[LemOFL: "ans",  Klasse: Artikel,
 Struktur: <[Adjektiv: E,
             Genus: Neutrum,
             KasusNumerus: AkkSg,
             Ergebnis: AdvD]>];
[LemOFL: "aufs", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
             Genus: Neutrum,
             KasusNumerus: AkkSg,
             Ergebnis: AdvD]>];
[LemOFL: "beim", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: DatSg,
             Ergebnis: AdvL]>];

[LemOFL: "das", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
             Genus: Neutrum,
             KasusNumerus:
             NomSg+AkkSg,
             Ergebnis: Sg3+AkkSg]>,
 Ergebnis: Sg3+AkkSg];


[LemOFL: "dasjenige", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
             Genus: Neutrum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk]>,
 Ergebnis: Sg3+Akk];

[LemOFL: "dasselbe", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
             Genus: Neutrum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk]>,
 Ergebnis: Sg3+Akk];

[LemOFL: "dein", Klasse: Artikel,
 Struktur: <[Adjektiv: ES,
             Genus: Neutrum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk],
            [Adjektiv: ER,
             Genus: Maskulinum,
             KasusNumerus: NomSg,
             Ergebnis: Sg3]>];

[LemOFL: "deine", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
             Genus: Femininum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk],
            [Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: NomPl+AkkPl,
             Ergebnis: Pl3+Akk]>,
 Ergebnis: Sg3+Pl3+Akk];

[LemOFL: "deinem", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: DatSg,
             Ergebnis: Dat]>,
 Ergebnis: Dat];

[LemOFL: "deinen", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: DatPl,
             Ergebnis: Dat],
            [Adjektiv: EN,
             Genus: Maskulinum,
             KasusNumerus: AkkSg,
             Ergebnis: AkkSg]>,
 Ergebnis: Dat+Akk];

[LemOFL: "deiner", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: GenPl,
             Ergebnis: Gen],
            [Adjektiv: EN,
             Genus: Femininum,
```

```
             KasusNumerus: GenSg+DatSg,
             Ergebnis: GenSg+DatSg]>,
 Ergebnis: GenSg+DatSg];

[LemOFL: "deines", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: GenSg,
             Ergebnis: Gen]>,
 Ergebnis: Gen];
[LemOFL: "dem",    Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: DatSg,
             Ergebnis: Dat]>,
 Ergebnis: Dat];
[LemOFL: "demjenigen", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: DatSg,
             Ergebnis: Dat]>,
 Ergebnis: Dat];
[LemOFL: "demselben", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: DatSg,
             Ergebnis: Dat]>,
 Ergebnis: Dat];

[LemOFL: "den", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum,
             KasusNumerus: AkkSg,
             Ergebnis: AkkSg],
            [Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: DatPl,
             Ergebnis: DatPl]>,
 Ergebnis: AkkSg];

[LemOFL: "denjenigen", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum,
             KasusNumerus: AkkSg,
             Ergebnis: AkkSg],
            [Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: DatPl,
             Ergebnis: Dat]>,
 Ergebnis: Dat+Akk];

[LemOFL: "denselben", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: DatPl,
             Ergebnis: Dat],
            [Adjektiv: EN,
             Genus: Maskulinum,
             KasusNumerus: AkkSg,
             Ergebnis: AkkSg]>,
 Ergebnis: DatSg+AkkSg];

[LemOFL: "der", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
             Genus: Maskulinum,
             KasusNumerus: NomSg,
             Ergebnis: Sg3],
            [Adjektiv: EN,
             Genus: Femininum,
             KasusNumerus: GenSg+DatSg,
             Ergebnis: GenSg+DatSg],
            [Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: GenPl,
             Ergebnis: GenPl]>,
 Ergebnis: Sg3+Gen+DatSg];

[LemOFL: "deren", Klasse: Artikel,
 Struktur: <[Adjektiv: ER,
             Genus: Maskulinum,
             KasusNumerus: NomSg,
             Ergebnis: Sg3],
            [Adjektiv: ES,
             Genus: Neutrum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk],
            [Adjektiv: E,
             Genus: Femininum,
             KasusNumerus: Singular,
             Ergebnis: Sg3+Akk],
            [Adjektiv: E,
             Genus: Kein_Genus,
             KasusNumerus: NomPl+AkkPl,
             Ergebnis: Pl3+Akk],
            [Adjektiv: ER,
             Genus: Maskulinum,
             KasusNumerus: AkkSg,
             Ergebnis: AkkSg],
            [Adjektiv: ES,
```

```
             Genus: Neutrum,
             KasusNumerus: AkkSg,
             Ergebnis: AkkSg],
            [Adjektiv: E,
             Genus: Maskulinum,
             KasusNumerus: NomSg,
             Ergebnis: Sg3]>,
 Ergebnis: Sg3+Pl3+Akk];

################################################
#hier fängt das korrigierte Original los
[LemOFL: "deren", Klasse: Artikel,
 Struktur: <[Adjektiv: ER,
             Genus: Maskulinum,
             KasusNumerus: NomSg,
             Ergebnis: Sg3],
            [Adjektiv: ES,
             Genus: Neutrum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk],
            [Adjektiv: E,
             Genus: Femininum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk],
            [Adjektiv: E,
             Genus: Kein_Genus,
             KasusNumerus: NomPl+AkkPl,
             Ergebnis: Pl3+Akk],
            [Adjektiv: ER,
             Genus: Maskulinum,
             KasusNumerus: AkkSg,
             Ergebnis: AkkSg],
            [Adjektiv: ES,
             Genus: Neutrum,
             KasusNumerus: AkkSg,
             Ergebnis: AkkSg],
            [Adjektiv: E,
             Genus: Maskulinum,
             KasusNumerus: NomSg,
             Ergebnis: Sg3]>,
 Ergebnis: Sg3+Pl3+Akk];

[LemOFL: "derjenigen", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Femininum,
             KasusNumerus: GenSg+DatSg,
             Ergebnis: GenSg+DatSg],
            [Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: GenPl,
             Ergebnis: Gen]>,
 Ergebnis: GenSg+DatSg];

#derlei funktioniert nur mit Substantiven im Plural
#  oder Massennomina
[LemOFL: "derlei", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
             Genus: Kein_Genus,
             KasusNumerus: NomPl+AkkPl,
             Ergebnis: Pl3+Akk],
            [Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: DatPl,
             Ergebnis: Dat],
            [Adjektiv: E,
             Genus: Femininum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk],
            [Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: GenSg,
             Ergebnis: Gen],
            [Adjektiv: EN,
             Genus: Maskulinum,
             KasusNumerus: AkkSg,
             Ergebnis: AkkSg],
            [Adjektiv: E,
             Genus: Maskulinum,
             KasusNumerus: NomSg,
             Ergebnis: Sg3],
            [Adjektiv: EM,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: DatSg,
             Ergebnis: Dat],
            [Adjektiv: E,
             Genus: Neutrum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk]>,
 Ergebnis: AkkSg];

[LemOFL: "derselbe", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
             Genus: Maskulinum,
             KasusNumerus: NomSg,
             Ergebnis: Sg3]>,
 Ergebnis: Sg3];

[LemOFL: "derselben", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
```

```
             Genus: Femininum,
             KasusNumerus: GenSg+DatSg,
             Ergebnis: GenSg+DatSg],
            [Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: GenPl,
             Ergebnis: Gen]>,
 Ergebnis: GenSg+DatSg];

[LemOFL: "des", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: GenSg,
             Ergebnis: Gen]>,
 Ergebnis: Gen];

[LemOFL: "desjenigen", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: GenSg,
             Ergebnis: Gen]>,
 Ergebnis: Gen];

[LemOFL: "desselben", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: GenSg,
             Ergebnis: Gen]>,
 Ergebnis: Gen];

[LemOFL: "dessen", Klasse: Artikel,
 Struktur: <[Adjektiv: ER,
             Genus: Maskulinum,
             KasusNumerus: NomSg,
             Ergebnis: Sg3],
            [Adjektiv: ES,
             Genus: Neutrum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk],
            [Adjektiv: E,
             Genus: Femininum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk],
            [Adjektiv: E,
             Genus: Kein_Genus,
             KasusNumerus: NomPl+AkkPl,
             Ergebnis: Pl3+Akk],
            [Adjektiv: EN,
             Genus: Maskulinum,
             KasusNumerus: AkkSg,
             Ergebnis: AkkSg]>,
 Ergebnis: Gen];

[LemOFL: "dich", Klasse: Artikel,
 Struktur: <[Adjektiv: ES,
             Genus: Neutrum,
             KasusNumerus: AkkSg,
             Ergebnis: AkkSg],
            [Adjektiv: EN,
             Genus: Maskulinum,
             KasusNumerus: AkkSg,
             Ergebnis: AkkSg],
            [Adjektiv: E,
             Genus: Femininum,
             KasusNumerus: AkkSg,
             Ergebnis: AkkSg]>,
 Ergebnis: AkkSg];

[LemOFL: "die", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
             Genus: Femininum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+AkkSg],
  [Adjektiv: EN, Genus: Kein_Genus, KasusNumerus: NomPl+AkkPl, Ergebnis: Pl3+AkkPl]>,
 Ergebnis: Sg3+Pl3+Akk];

[LemOFL: "diejenige",  Klasse: Artikel,
 Struktur: <[Adjektiv: E,
             Genus: Femininum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk]>, Ergebnis: Sg3+Akk];
[LemOFL: "diejenigen", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: NomPl+AkkPl,
             Ergebnis: Pl3+Akk]>, Ergebnis: Pl3+Akk];

[LemOFL: "diese", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
             Genus: Femininum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk],
            [Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: NomPl+AkkPl,
             Ergebnis: Pl3+Akk]>,
 Ergebnis: Sg3+Pl3+Akk];

[LemOFL: "dieselbe",  Klasse: Artikel,
```

```
 Struktur: <[Adjektiv: E,
             Genus: Femininum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk]>, Ergebnis: Sg3+Akk];

[LemOFL: "dieselben", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: NomPl+AkkPl,
             Ergebnis: Pl3+Akk]>, Ergebnis: Pl3+Akk];

[LemOFL: "diesem", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: DatSg,
             Ergebnis: Dat]>, Ergebnis: Dat];

[LemOFL: "diesen", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum,
             KasusNumerus: AkkSg,
             Ergebnis: AkkSg],
            [Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: DatPl,
             Ergebnis: Dat]>,
 Ergebnis: Dat+Akk];

[LemOFL: "dieser", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
             Genus: Maskulinum,
             KasusNumerus: NomSg,
             Ergebnis: Sg3],
            [Adjektiv: EN,
             Genus: Femininum,
             KasusNumerus: GenSg+DatSg,
             Ergebnis: GenSg+DatSg],
            [Adjektiv: EN,
             Genus: Femininum,
             KasusNumerus: GenSg,
             Ergebnis: Gen],
            [Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: GenPl,
             Ergebnis: Gen]>,
 Ergebnis: Sg3+Gen+Dat];

[LemOFL: "dieses", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
             Genus: Neutrum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk],
            [Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: GenSg,
             Ergebnis: Gen]>,
 Ergebnis: Sg3+Akk];

[LemOFL: "dir", Klasse: Artikel,
 Struktur: <[Adjektiv: EM,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: DatSg,
             Ergebnis: Dat],
            [Adjektiv: EN,
             Genus: Femininum,
             KasusNumerus: DatSg,
             Ergebnis: Dat]>,
 Ergebnis: Dat];

[LemOFL: "ein", Klasse: Artikel,
 Struktur: <[Adjektiv: ER,
             Genus: Maskulinum,
             KasusNumerus: NomSg,
             Ergebnis: Sg3],
            [Adjektiv: ES,
             Genus: Neutrum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk]>];

[LemOFL: "eine", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
             Genus: Femininum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk]>, Ergebnis: Sg3+Akk];
[LemOFL: "einem", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: DatSg,
             Ergebnis: Dat]>, Ergebnis: Dat];
[LemOFL: "einen", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum,
             KasusNumerus: AkkSg,
             Ergebnis: AkkSg]>, Ergebnis: AkkSg];
[LemOFL: "einer", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Femininum,
             KasusNumerus: GenSg+DatSg,
             Ergebnis: GenSg+DatSg]>];
```

```
[LemOFL: "eines", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: GenSg,
             Ergebnis: Gen]>, Ergebnis: Sg3+Akk];
[LemOFL: "etliche", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
             Genus: Kein_Genus,
             KasusNumerus: NomPl+AkkPl,
             Ergebnis: Pl3+Akk]>,
 Ergebnis: Pl3+Akk];

[LemOFL: "etlichem", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: DatSg,
             Ergebnis: Dat]>, Ergebnis: Dat];

[LemOFL: "etlichen", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: DatPl,
             Ergebnis: Dat]>,
 Ergebnis: Dat];

[LemOFL: "etlicher", Klasse: Artikel,
 Struktur: <[Adjektiv: ER,
             Genus: Maskulinum,
             KasusNumerus: NomSg,
             Ergebnis: Sg3],
            [Adjektiv: ER,
             Genus: Femininum,
             KasusNumerus: GenSg+DatSg,
             Ergebnis: GenSg+DatSg],
            [Adjektiv: ER,
             Genus: Kein_Genus,
             KasusNumerus: GenPl,
             Ergebnis: Gen]>,
 Ergebnis: Sg3+Gen+Dat];

[LemOFL: "etliches", Klasse: Artikel,
 Struktur: <[Adjektiv: ES,
             Genus: Neutrum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk]>, Ergebnis: Sg3+Akk];

[LemOFL: "euer", Klasse: Artikel,
 Struktur: <[Adjektiv: ES,
             Genus: Neutrum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk],
            [Adjektiv: ER,
             Genus: Maskulinum,
             KasusNumerus: NomSg,
             Ergebnis: Sg3]>];

[LemOFL: "euerm", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: DatSg,
             Ergebnis: Dat]>, Ergebnis: Dat];
[LemOFL: "euern", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: DatPl,
             Ergebnis: Dat],
            [Adjektiv: EN,
             Genus: Maskulinum,
             KasusNumerus: AkkSg,
             Ergebnis: AkkSg]>, Ergebnis: Dat+Akk];

[LemOFL: "eure", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
             Genus: Femininum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk],
            [Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: NomPl+AkkPl,
             Ergebnis: Pl3+Akk]>,
 Ergebnis: Sg3+Pl3+Akk];

[LemOFL: "eurem", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: DatSg,
             Ergebnis: Dat]>, Ergebnis: Dat];
[LemOFL: "euren", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: DatPl,
             Ergebnis: Dat],
            [Adjektiv: EN,
             Genus: Maskulinum,
             KasusNumerus: AkkSg,
             Ergebnis: AkkSg]>,
 Ergebnis: Dat+Akk];

[LemOFL: "eurer", Klasse: Artikel,
```

```
 Struktur: <[Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: GenPl,
             Ergebnis: Gen],
            [Adjektiv: EN,
             Genus: Femininum,
             KasusNumerus: GenSg+DatSg,
             Ergebnis: GenSg+DatSg]>, Ergebnis: GenSg+DatSg];

[LemOFL: "eures", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: GenSg,
             Ergebnis: Gen]>];

[LemOFL: "ihre", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
             Genus: Femininum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk],
            [Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: NomPl+AkkPl,
             Ergebnis: Pl3+Akk]>,
 Ergebnis: Sg3+Pl3+Akk];

[LemOFL: "ihrem", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: DatSg,
             Ergebnis: Dat]>, Ergebnis: Dat];
[LemOFL: "ihren", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: DatPl,
             Ergebnis: Dat],
            [Adjektiv: EN,
             Genus: Maskulinum,
             KasusNumerus: AkkSg,
             Ergebnis: AkkSg]>,
 Ergebnis: Dat+Akk];

[LemOFL: "ihrer", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: GenPl,
             Ergebnis: Gen],
            [Adjektiv: EN,
             Genus: Femininum,
             KasusNumerus: GenSg+DatSg,
             Ergebnis: GenSg+DatSg]>,
 Ergebnis: GenSg+DatSg];

[LemOFL: "ihres", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: GenSg,
             Ergebnis: Gen]>];
[LemOFL: "im",    Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: DatSg,
             Ergebnis: AdvL]>];

[LemOFL: "irgendein", Klasse: Artikel,
 Struktur: <[Adjektiv: ER,
             Genus: Maskulinum,
             KasusNumerus: NomSg,
             Ergebnis: Sg3],
            [Adjektiv: ES,
             Genus: Neutrum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk]>];

[LemOFL: "irgendeine", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
             Genus: Femininum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk]>, Ergebnis: Sg3+Akk];
[LemOFL: "irgendeinem", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: DatSg,
             Ergebnis: Dat]>, Ergebnis: Dat];
[LemOFL: "irgendeinen", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum,
             KasusNumerus: AkkSg,
             Ergebnis: AkkSg]>, Ergebnis: AkkSg];
[LemOFL: "irgendeiner", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Femininum,
             KasusNumerus: GenSg+DatSg,
             Ergebnis: GenSg+DatSg]>, Ergebnis: GenSg+DatSg];
[LemOFL: "irgendeines", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: GenSg,
             Ergebnis: Gen]>, Ergebnis: Gen];
```

```
[LemOFL: "irgendwelche", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: NomPl+AkkPl,
             Ergebnis: Pl3+Akk]>, Ergebnis: Pl3+Akk];

[LemOFL: "irgendwelchen", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: DatPl,
             Ergebnis: Dat]>, Ergebnis: Dat];
[LemOFL: "irgendwelcher", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: GenPl,
             Ergebnis: Gen]>, Ergebnis: Gen];

[LemOFL: "jede", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
             Genus: Femininum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk]>, Ergebnis: Sg3+Akk];
[LemOFL: "jedem", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: DatSg,
             Ergebnis: Dat]>, Ergebnis: Dat];

[LemOFL: "jeden", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum,
             KasusNumerus: AkkSg,
             Ergebnis: AkkSg],
            [Adjektiv: EN,
             Genus: Neutrum,
             KasusNumerus: GenSg,
             Ergebnis: Gen]>,
 Ergebnis: AkkSg];

[LemOFL: "jeder", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
             Genus: Maskulinum,
             KasusNumerus: NomSg,
             Ergebnis: Sg3],
            [Adjektiv: EN,
             Genus: Femininum,
             KasusNumerus: GenSg+DatSg,
             Ergebnis: GenSg+DatSg]>,
 Ergebnis: Sg3+Gen+Dat];

[LemOFL: "jedes", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
             Genus: Neutrum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk],
            [Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: GenSg,
             Ergebnis: Gen]>,
 Ergebnis: Sg3+Akk];

[LemOFL: "jene", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
             Genus: Femininum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk],
            [Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Pl3+Akk]>,
 Ergebnis: Sg3+Pl3+Akk];

[LemOFL: "jenem", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: DatSg,
             Ergebnis: Dat]>, Ergebnis: Dat];

[LemOFL: "jenen", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum,
             KasusNumerus: AkkSg,
             Ergebnis: AkkSg],
            [Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: DatPl,
             Ergebnis: Dat]>,
 Ergebnis: Dat+Akk];

[LemOFL: "jener", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
             Genus: Maskulinum,
             KasusNumerus: NomSg,
             Ergebnis: Sg3],
            [Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: GenPl,
             Ergebnis: Gen],
            [Adjektiv: EN,
```

```
            Genus: Femininum,
            KasusNumerus: GenSg+DatSg,
            Ergebnis: GenSg+DatSg]>,
 Ergebnis: Sg3+Gen+Dat];

[LemOFL: "jenes", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
            Genus: Neutrum,
            KasusNumerus: NomSg+AkkSg,
            Ergebnis: Sg3+Akk],
           [Adjektiv: EN,
            Genus: Maskulinum+Neutrum,
            KasusNumerus: GenSg,
            Ergebnis: Gen]>,
 Ergebnis: Sg3+Akk];

[LemOFL: "kein", Klasse: Artikel,
 Struktur: <[Adjektiv: ER,
            Genus: Maskulinum,
            KasusNumerus: NomSg,
            Ergebnis: Sg3],
           [Adjektiv: ES,
            Genus: Neutrum,
            KasusNumerus: NomSg+AkkSg,
            Ergebnis: Sg3+Akk]>];

[LemOFL: "keine", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
            Genus: Femininum,
            KasusNumerus: NomSg+AkkSg,
            Ergebnis: Sg3+Akk]>, Ergebnis: Sg3+Akk];
[LemOFL: "keinem", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
            Genus: Maskulinum+Neutrum,
            KasusNumerus: DatSg,
            Ergebnis: Dat]>, Ergebnis: Dat];
[LemOFL: "keinen", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
            Genus: Maskulinum,
            KasusNumerus: AkkSg,
            Ergebnis: AkkSg]>, Ergebnis: AkkSg];
[LemOFL: "keiner", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
            Genus: Femininum,
            KasusNumerus: GenSg+DatSg,
            Ergebnis: GenSg+DatSg]>, Ergebnis: Sg3+Gen+Dat];
[LemOFL: "keines", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
            Genus: Maskulinum+Neutrum,
            KasusNumerus: GenSg,
            Ergebnis: Gen]>, Ergebnis: Sg3+Akk];

[LemOFL: "manch", Klasse: Artikel,
 Struktur: <[Adjektiv: ER,
            Genus: Maskulinum,
            KasusNumerus: NomSg,
            Ergebnis: Sg3],
           [Adjektiv: EN,
            Genus: Maskulinum,
            KasusNumerus: AkkSg,
            Ergebnis: AkkSg],
           [Adjektiv: E,
            Genus: Femininum,
            KasusNumerus: NomSg+AkkSg,
            Ergebnis: Sg3+Akk],
           [Adjektiv: ES,
            Genus: Neutrum,
            KasusNumerus: NomSg+AkkSg,
            Ergebnis: Sg3+Akk]>];

[LemOFL: "manche", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
            Genus: Femininum,
            KasusNumerus: NomSg+AkkSg,
            Ergebnis: Sg3+Akk],
           [Adjektiv: EN,
            Genus: Kein_Genus,
            KasusNumerus: NomPl+AkkPl,
            Ergebnis: Pl3+Akk]>,
 Ergebnis: Sg3+Pl3+Akk];

[LemOFL: "manchem", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
            Genus: Maskulinum+Neutrum,
            KasusNumerus: DatSg,
            Ergebnis: Dat]>, Ergebnis: Dat];

[LemOFL: "manchen", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
            Genus: Maskulinum,
            KasusNumerus: AkkSg,
            Ergebnis: AkkSg],
           [Adjektiv: EN,
            Genus: Kein_Genus,
            KasusNumerus: DatPl,
            Ergebnis: Dat]>,
 Ergebnis: Dat+Akk];

[LemOFL: "mancher", Klasse: Artikel,
```

```
 Struktur: <[Adjektiv: E,
             Genus: Maskulinum,
             KasusNumerus: NomSg,
             Ergebnis: Sg3],
            [Adjektiv: EN,
             Genus: Femininum,
             KasusNumerus: GenSg+DatSg,
             Ergebnis: GenSg+DatSg],
            [Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: GenPl,
             Ergebnis: Gen]>,
 Ergebnis: Sg3+Gen];

[LemOFL: "manches", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
             Genus: Neutrum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk],
            [Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: GenSg,
             Ergebnis: Gen]>,
 Ergebnis: Sg3+Akk];

[LemOFL: "mehrere", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
             Genus: Kein_Genus,
             KasusNumerus: NomPl+AkkPl,
             Ergebnis: Pl3+Akk]>, Ergebnis: Pl3+Akk];
[LemOFL: "mehreren", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: DatPl,
             Ergebnis: Dat]>, Ergebnis: Dat];
[LemOFL: "mehrerer", Klasse: Artikel,
 Struktur: <[Adjektiv: ER,
             Genus: Kein_Genus,
             KasusNumerus: GenPl,
             Ergebnis: Gen]>, Ergebnis: Gen];

[LemOFL: "mein", Klasse: Artikel,
 Struktur: <[Adjektiv: ER,
             Genus: Maskulinum,
             KasusNumerus: NomSg,
             Ergebnis: Sg3],
            [Adjektiv: ES,
             Genus: Neutrum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk]>];

[LemOFL: "meine", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
             Genus: Femininum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk],
            [Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: NomPl+AkkPl,
             Ergebnis: Pl3+Akk]>,
 Ergebnis: Sg3+Pl3+Akk];

[LemOFL: "meinem", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: DatSg,
             Ergebnis: Dat]>, Ergebnis: Dat];
[LemOFL: "meinen", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum,
             KasusNumerus: AkkSg,
             Ergebnis: AkkSg],
            [Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: DatPl,
             Ergebnis: Dat]>,
 Ergebnis: Dat+Akk];
[LemOFL: "meiner", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Femininum,
             KasusNumerus: GenSg+DatSg,
             Ergebnis: GenSg+DatSg],
            [Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: DatPl,
             Ergebnis: Gen]>,
 Ergebnis: GenSg+DatSg];
[LemOFL: "meines", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: GenSg,
             Ergebnis: Gen]>, Ergebnis: Sg3+Akk];

[LemOFL: "per", Klasse: Artikel,
 Struktur: <[Adjektiv: EM,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: DatSg,
             Ergebnis: AdvM],
            [Adjektiv: ER,
```

```
             Genus: Femininum,
             KasusNumerus: DatSg,
             Ergebnis: AdvM],
            [Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: DatPl,
             Ergebnis: AdvM]>];

[LemOFL: "sein", Klasse: Artikel,
 Struktur: <[Adjektiv: ER,
             Genus: Maskulinum,
             KasusNumerus: NomSg,
             Ergebnis: Sg3],
            [Adjektiv: ES,
             Genus: Neutrum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk]>];

[LemOFL: "seine", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
             Genus: Femininum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk],
            [Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: NomPl+AkkPl,
             Ergebnis: Pl3+Akk]>,
 Ergebnis: Sg3+Pl3+Akk];

[LemOFL: "seinem", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: DatSg,
             Ergebnis: Dat]>, Ergebnis: Dat];

[LemOFL: "seinen", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: DatPl,
             Ergebnis: Dat],
            [Adjektiv: EN,
             Genus: Maskulinum,
             KasusNumerus: AkkSg,
             Ergebnis: AkkSg]>,
 Ergebnis: Dat+Akk];

[LemOFL: "seiner", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: GenPl,
             Ergebnis: Gen],
            [Adjektiv: EN,
             Genus: Femininum,
             KasusNumerus: GenSg+DatSg,
             Ergebnis: GenSg+DatSg]>,
 Ergebnis: GenSg+DatSg];

[LemOFL: "seines", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
             Genus: Maskulinum+Neutrum,
             KasusNumerus: GenSg,
             Ergebnis: Gen]>,
 Ergebnis: Gen];

[LemOFL: "uns", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
             Genus: Kein_Genus,
             KasusNumerus: AkkPl,
             Ergebnis: AkkSg],
            [Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: DatPl,
             Ergebnis: Dat]>,
 Ergebnis: Dat+Akk];

[LemOFL: "unser", Klasse: Artikel,
 Struktur: <[Adjektiv: ES,
             Genus: Neutrum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk],
            [Adjektiv: ER,
             Genus: Maskulinum,
             KasusNumerus: NomSg,
             Ergebnis: Sg3]>,
 Ergebnis: Sg3+Akk];

[LemOFL: "unsere", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
             Genus: Femininum,
             KasusNumerus: NomSg+AkkSg,
             Ergebnis: Sg3+Akk],
            [Adjektiv: EN,
             Genus: Kein_Genus,
             KasusNumerus: NomPl+AkkPl,
             Ergebnis: Pl3+Akk]>,
 Ergebnis: Sg3+Pl3+Akk];

[LemOFL: "unserem", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
```

```
            Genus: Maskulinum+Neutrum,
            KasusNumerus: DatSg,
            Ergebnis: Dat]>,
 Ergebnis: Dat];

[LemOFL: "unseren", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
            Genus: Kein_Genus,
            KasusNumerus: DatPl,
            Ergebnis: Dat],
           [Adjektiv: EN,
            Genus: Maskulinum,
            KasusNumerus: AkkSg,
            Ergebnis: AkkSg]>,
 Ergebnis: Dat+Akk];

[LemOFL: "unserer", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
            Genus: Kein_Genus,
            KasusNumerus: GenPl,
            Ergebnis: Gen],
           [Adjektiv: EN,
            Genus: Femininum,
            KasusNumerus: GenSg+DatSg,
            Ergebnis: GenSg+DatSg]>,
 Ergebnis: GenSg+DatSg];

[LemOFL: "unseres", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
            Genus: Maskulinum+Neutrum,
            KasusNumerus: GenSg,
            Ergebnis: Gen]>,
 Ergebnis: Gen];

[LemOFL: "unserm", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
            Genus: Maskulinum+Neutrum,
            KasusNumerus: DatSg,
            Ergebnis: Dat]>, Ergebnis: Dat];

[LemOFL: "unsern", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
            Genus: Kein_Genus,
            KasusNumerus: DatPl,
            Ergebnis: Dat],
           [Adjektiv: EN,
            Genus: Maskulinum,
            KasusNumerus: AkkSg,
            Ergebnis: AkkSg]>,
 Ergebnis: Dat+Akk];

[LemOFL: "unsre", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
            Genus: Femininum,
            KasusNumerus: NomSg+AkkSg,
            Ergebnis: Sg3+Akk],
           [Adjektiv: EN,
            Genus: Kein_Genus,
            KasusNumerus: NomPl+AkkPl,
            Ergebnis: Pl3+Akk]>,
 Ergebnis: Sg3+Pl3+Akk];

[LemOFL: "unsrem", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
            Genus: Maskulinum+Neutrum,
            KasusNumerus: DatSg,
            Ergebnis: Dat]>, Ergebnis: Dat];

[LemOFL: "unsren", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
            Genus: Kein_Genus,
            KasusNumerus: DatPl,
            Ergebnis: Dat],
           [Adjektiv: EN,
            Genus: Maskulinum,
            KasusNumerus: AkkSg,
            Ergebnis: AkkSg]>,
 Ergebnis: Dat+Akk];

[LemOFL: "unsrer", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
            Genus: Kein_Genus,
            KasusNumerus: GenPl,
            Ergebnis: Gen],
           [Adjektiv: EN,
            Genus: Femininum,
            KasusNumerus: GenSg+DatSg,
            Ergebnis: GenSg+DatSg]>,
 Ergebnis: GenSg+DatSg];

[LemOFL: "unsres", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
            Genus: Maskulinum+Neutrum,
            KasusNumerus: GenSg,
            Ergebnis: Gen]>,
 Ergebnis: Gen];

[LemOFL: "viele", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
```

```
            Genus: Kein_Genus,
            KasusNumerus: NomPl+AkkPl,
            Ergebnis: Pl3+Akk]>];
[LemOFL: "vielen", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
            Genus: Kein_Genus,
            KasusNumerus: DatPl,
            Ergebnis: Dat]>, Ergebnis: Dat];
[LemOFL: "vieler", Klasse: Artikel,
 Struktur: <[Adjektiv: ER,
            Genus: Kein_Genus,
            KasusNumerus: GenPl,
            Ergebnis: Gen]>];

[LemOFL: "vielerlei", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
            Genus: Kein_Genus,
            KasusNumerus: NomPl+AkkPl,
            Ergebnis: Pl3+Akk],
           [Adjektiv: EN,
            Genus: Kein_Genus,
            KasusNumerus: DatPl,
            Ergebnis: Dat],
           [Adjektiv: ER,
            Genus: Kein_Genus,
            KasusNumerus: GenPl,
            Ergebnis: Gen]>,
 Ergebnis: AkkSg];

[LemOFL: "vom", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
            Genus: Maskulinum+Neutrum,
            KasusNumerus: DatSg,
            Ergebnis: AdvL]>];
[LemOFL: "vorm", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
            Genus: Maskulinum+Neutrum,
            KasusNumerus: DatSg,
            Ergebnis: AdvL]>];
[LemOFL: "weitere", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
            Genus: Kein_Genus,
            KasusNumerus: NomPl+AkkPl,
            Ergebnis: Pl3+Akk]>, Ergebnis: Pl3+Akk];
[LemOFL: "zum", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
            Genus: Maskulinum+Neutrum,
            KasusNumerus: DatSg,
            Ergebnis: AdvD]>];
[LemOFL: "zur", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
            Genus: Femininum,
            KasusNumerus: GenSg+DatSg,
            Ergebnis: AdvD]>];
[LemOFL: "durchs", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
            Genus: Neutrum,
            KasusNumerus: NomSg+AkkSg,
            Ergebnis: AdvD]>];
[LemOFL: "fürs", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
            Genus: Neutrum,
            KasusNumerus: NomSg+AkkSg,
            Ergebnis: AdvM]>];

[LemOFL: "keinerlei", Klasse: Artikel,
 Struktur: <[Adjektiv: EN,
            Genus: Kein_Genus,
            KasusNumerus: NomPl+AkkPl,
            Ergebnis: Pl3+Akk],
           [Adjektiv: ER,
            Genus: Kein_Genus,
            KasusNumerus: GenPl,
            Ergebnis: Gen],
           [Adjektiv: EN,
            Genus: Kein_Genus,
            KasusNumerus: DatPl,
            Ergebnis: Dat]>];

[LemOFL: "ums", Klasse: Artikel,
 Struktur: <[Adjektiv: E,
            Genus: Neutrum,
            KasusNumerus: NomSg+AkkSg,
            Ergebnis: AdvD]>];
[LemOFL: "weiterer", Klasse: Artikel,
 Struktur: <[Adjektiv: ER,
            Genus: Kein_Genus,
            KasusNumerus: GenPl,
            Ergebnis: Gen]>];


#Interrogativpronomina und -adverbien
#Frageadverbien
[LemOFL: "inwiefern", Klasse: Adverb,
 Struktur: <[Syntax: AdvK, Ergebnis: Interrog]>];
[LemOFL: "wann",      Klasse: Adverb,
 Struktur: <[Syntax: AdvT, Ergebnis: Interrog]>];
[LemOFL: "warum",     Klasse: Adverb,
 Struktur: <[Syntax: AdvK, Ergebnis: Interrog]>];
```

```
[LemOFL: "weshalb",   Klasse: Adverb,
 Struktur: <[Syntax: AdvK, Ergebnis: Interrog]>];
[LemOFL: "wie",       Klasse: Adverb,
 Struktur: <[Syntax: AdvM, Ergebnis: Interrog]>];
[LemOFL: "wo",        Klasse: Adverb,
 Struktur: <[Syntax: AdvL, Ergebnis: Interrog]>];
[LemOFL: "woher",     Klasse: Adverb,
 Struktur: <[Syntax: AdvD, Ergebnis: Interrog]>];
[LemOFL: "wohin",     Klasse: Adverb,
 Struktur: <[Syntax: AdvD, Ergebnis: Interrog]>];
[LemOFL: "wieviel", Klasse: Adverb,
 Struktur: <[Syntax: AdvM, Ergebnis: Interrog]>];
[LemOFL: "wodurch", Klasse: Adverb,
 Struktur: <[Syntax: AdvK, Ergebnis: Interrog]>];
[LemOFL: "wonach", Klasse: Adverb,
 Struktur: <[Syntax: AdvK, Ergebnis: Interrog]>];

#Fragepronomina
[LemOFL: "was",     Klasse: Pron,
 Struktur: <[Syntax: Sg3+Akk,  Ergebnis: Interrog]>];
[LemOFL: "wer",     Klasse: Pron,
 Struktur: <[Syntax: Sg3,    Ergebnis: Interrog]>];
[LemOFL: "wem",     Klasse: Pron,
 Struktur: <[Syntax: Dat,     Ergebnis: Interrog]>];
[LemOFL: "welche", Klasse: Pron,
 Struktur: <[Syntax: Sg3+Akk,  Ergebnis: Interrog]>];
[LemOFL: "welchem", Klasse: Pron,
 Struktur: <[Syntax: Dat,  Ergebnis: Interrog]>];
[LemOFL: "welchen", Klasse: Pron,
 Struktur: <[Syntax: DatSg+AkkSg,  Ergebnis: Interrog]>];
[LemOFL: "welcher", Klasse: Pron,
 Struktur: <[Syntax: Sg3+Dat,  Ergebnis: Interrog]>];
[LemOFL: "welches", Klasse: Pron,
 Struktur: <[Syntax: Sg3+Akk,  Ergebnis: Interrog]>];

#Konjunktionen
#Daß
[LemOFL: "daß",          Klasse: Kon, Ergebnis: DassNebs];

#Nebenordnende
[LemOFL: "aber",         Klasse: Kon, Ergebnis: KoKon];
[LemOFL: "also",         Klasse: Kon, Ergebnis: KoKon];
[LemOFL: "andererseits", Klasse: Kon, Ergebnis: KoKon];
[LemOFL: "außerdem",     Klasse: Kon, Ergebnis: KoKon];
[LemOFL: "denn",         Klasse: Kon, Ergebnis: KoKon];
[LemOFL: "dennoch",      Klasse: Kon, Ergebnis: KoKon];
[LemOFL: "derweil",      Klasse: Kon, Ergebnis: KoKon];
[LemOFL: "deswegen",     Klasse: Kon, Ergebnis: KoKon];
[LemOFL: "deswegen",     Klasse: Kon, Ergebnis: KoKon];
[LemOFL: "doch",         Klasse: Kon, Ergebnis: KoKon];
[LemOFL: "insofern",     Klasse: Kon, Ergebnis: KoKon];
[LemOFL: "jedoch",       Klasse: Kon, Ergebnis: KoKon];
[LemOFL: "mithin",       Klasse: Kon, Ergebnis: KoKon];
[LemOFL: "oder",         Klasse: Kon, Ergebnis: KoKon];
[LemOFL: "sowie",        Klasse: Kon, Ergebnis: KoKon];
[LemOFL: "und",          Klasse: Kon, Ergebnis: KoKon];
[LemOFL: "vielmehr",     Klasse: Kon, Ergebnis: KoKon];

#Unterordnende
[LemOFL: "als",          Klasse: Kon, Ergebnis: SubKon];
[LemOFL: "andernfalls",  Klasse: Kon, Ergebnis: SubKon];
[LemOFL: "bevor",        Klasse: Kon, Ergebnis: SubKon];
[LemOFL: "da",           Klasse: Kon, Ergebnis: SubKon];
[LemOFL: "damit",        Klasse: Kon, Ergebnis: SubKon];
[LemOFL: "falls",        Klasse: Kon, Ergebnis: SubKon];
[LemOFL: "indem",        Klasse: Kon, Ergebnis: SubKon];
[LemOFL: "nachdem",      Klasse: Kon, Ergebnis: SubKon];
[LemOFL: "obgleich",     Klasse: Kon, Ergebnis: SubKon];
[LemOFL: "obwohl",       Klasse: Kon, Ergebnis: SubKon];
[LemOFL: "sobald",       Klasse: Kon, Ergebnis: SubKon];
[LemOFL: "sofern",       Klasse: Kon, Ergebnis: SubKon];
[LemOFL: "solange",      Klasse: Kon, Ergebnis: SubKon];
[LemOFL: "soweit",       Klasse: Kon, Ergebnis: SubKon];
[LemOFL: "weil",         Klasse: Kon, Ergebnis: SubKon];
[LemOFL: "wenn",         Klasse: Kon, Ergebnis: SubKon];
[LemOFL: "weshalb",      Klasse: Kon, Ergebnis: SubKon];
[LemOFL: "wobei",        Klasse: Kon, Ergebnis: SubKon];
[LemOFL: "zumal", Klasse: Kon, Ergebnis: SubKon];

# Präfixoide

[LemOFl: "super",
 Klasse: Präfixoid,
 Gewicht: 0.6,
 Px: [OFl: nein,
      Suche: <Klasse, ist_in, <Substantiv, Adjektiv>>,
      Ergebnis: []]];

[LemOFl: "mini",
 Klasse: Präfixoid,
 Gewicht: 0.6,
 Px: [OFl: nein,
      Suche: <Klasse, ist_in, <Substantiv>>,
      Ergebnis: []]];

[LemOFl: "ex",
 Klasse: Präfixoid,
 Gewicht: 0.6,
 Px: [OFl: nein,
```

```
     Suche: <Klasse, ist_in, <Substantiv>>,
     Ergebnis: []]];

[LemOFl: "extra",
 Klasse: Präfixoid,
 Gewicht: 0.6,
 Px: [OFl: nein,
     Suche: <Klasse, ist_in, <Substantiv, Adjektiv>>,
     Ergebnis: []]];

[LemOFl: "un",
 Klasse: Präfixoid,
 Gewicht: 0.6,
 Px: [OFl: nein,
     Suche: <Klasse, ist_in, <Adjektiv>>,
     Ergebnis: []]];

# Präfixe (hier muß noch aufgeräumt werden !)
[LemOFl: "abwärts",     Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "acht",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "alt",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "aneinander",  Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "anheim",      Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "aufeinander", Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "aufrecht",    Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "auftrags",    Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "auseinander", Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "be",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "beiseite",    Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "bekannt",     Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "bereit",      Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "bestehen",    Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "bewußt",      Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "blank",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "blind",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "bloß",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "brach",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "breit",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "da",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "dabei",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "dafür",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "dagegen",     Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "daher",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "dahin",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "dahinter",    Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "daneben",     Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "danieder",    Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "dank",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "dar",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "daran",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "darauf",      Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "darauflos",   Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "darnieder",   Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "darunter",    Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "darüber",     Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "davon",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "davor",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "dazu",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "dazwischen",  Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "de",          Klasse: Präfix, Px: [OFl: ja]];
#[LemOFl: "des",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "dicht",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "dis",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "drauf",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "drauflos",    Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "draus",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "drei",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "drein",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "drin",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "durcheinander", Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "dünn",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "dünne",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "ein",           Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "einher",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "eis",           Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "empor",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "ent",           Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "entlang",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "entzwei",       Klasse: Präfix, Px: [OFl: ja]];
#[LemOFl: "er",           Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "falsch",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "fehl",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "feil",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "fein",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "fern",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "fertig",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "fest",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "film",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "flach",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "fliess",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "flott",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "for",           Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "fort",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "frei",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "froh",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "ge",            Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "ge",
 Klasse: Präfix,
 Px: [OFl: nein,
```

```
     Suche: <PII-Ge, kongr, Ge>,
     Ergebnis: [PII-Ge: Ge-, Blockieren: Flexion]]];
[LemOFl: "gefangen",      Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "gegeneinander", Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "geheim",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "general",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "gerade",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "gering",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "geschmeidig",   Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "gesund",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "glatt",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "gleich",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "groß",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "gut",           Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "haften",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "halt",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "haus",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "heilig",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "heim",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "hell",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "her",           Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "herab",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "heran",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "herauf",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "heraus",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "herbei",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "herein",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "herum",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "herumfuhr",     Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "herunter",      Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "hervor",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "herzu",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "herüber",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "hier",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "hierher",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "hin",           Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "hinab",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "hinauf",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "hinaus",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "hindurch",      Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "hinein",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "hintan",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "hinterdrein",   Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "hintereinander",Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "hinterher",     Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "hinunter",      Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "hinweg",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "hinzu",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "hinüber",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "hoch",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "hohl",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "hohn",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "hängen",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "ineinander",    Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "inne",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "instand",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "irre",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "kahl",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "kalt",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "kaputt",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "kehrt",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "kenn",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "kennen",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "klar",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "kleben",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "klebrig",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "klein",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "klug",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "knapp",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "krank",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "kund",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "kurz",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "lahm",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "lang",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "leer",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "leicht",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "licht",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "lieb",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "liegen",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "lob",           Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "locker",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "los",           Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "lust",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "maschine",      Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "maß",           Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "mies",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "mitan",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "mitent",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "miß",           Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "mobil",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "mündig",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "nahe",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "nebeneinander", Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "neu",           Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "nieder",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "not",           Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "näher",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "ob",            Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "offen",         Klasse: Präfix, Px: [OFl: ja]];
```

```
[LemOFl: "preis",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "preß",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "prä",           Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "quer",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "rad",           Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "raus",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "re",            Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "recht",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "rein",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "retro",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "richtig",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "ruhig",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "rund",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "rüber",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "rück",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "rücken",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "sauber",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "saug",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "scharf",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "schau",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "schief",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "schlecht",      Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "schlitten",     Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "schmelz",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "schutz",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "schwarz",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "schwer",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "schätzen",      Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "schön",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "seil",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "selbst",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "sicher",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "sitzen",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "spazieren",     Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "staffel",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "stand",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "staub",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "stecken",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "stehen",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "stich",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "still",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "straf",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "taucher",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "tot",           Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "traum",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "umbe",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "umher",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "ur",            Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "urauf",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "ver",           Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "verab",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "verleit",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "vertraut",      Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "voll",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "voran",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "voraus",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "vorbei",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "vorher",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "vorweg",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "vorwärts",      Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "wach",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "wahr",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "warm",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "weg",           Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "weis",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "weit",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "weiter",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "weiß",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "wert",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "wett",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "wider",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "wieder",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "wiederab",      Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "wiederan",      Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "wiederauf",     Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "wiederein",     Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "wiedergut",     Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "wiederher",     Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "wiederinstand", Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "wiederzu",      Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "will",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "wund",          Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "zer",           Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "zufrieden",     Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "zugrunde",      Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "zurecht",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "zurück",        Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "zusammen",      Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "zustande",      Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "zustatten",     Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "zuvor",         Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "zuwider",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "überein",       Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "übereinander",  Klasse: Präfix, Px: [OFl: ja]];
[LemOFl: "übrig",         Klasse: Präfix, Px: [OFl: ja]];

#Präpositionen
[LemOFL: "angesichts", Klasse: Präp,
 Struktur: <[Val: Gen, Ergebnis: AdvM]>];
```

```
[LemOFL: "anläßlich", Klasse: Präp,
 Struktur: <[Val: Gen, Ergebnis: AdvM]>];
[LemOFL: "außerhalb", Klasse: Präp,
 Struktur: <[Val: Gen, Ergebnis: AdvL]>];
[LemOFL: "binnen", Klasse: Präp,
 Struktur: <[Val: Gen, Ergebnis: AdvT]>];
[LemOFL: "zuviel", Klasse: Präp,
 Struktur: <[Val: Gen, Ergebnis: AdvM]>, Ergebnis: AdvM];

#Präpositionen (mit Verwendung als Präfix)

[LemOFL: "ab", Px: [OFl: ja], Klasse: Präp,
 Struktur: <[Val: Dat, Ergebnis: AdvT]>];
[LemOFL: "abseits", Klasse: Präp,
 Struktur: <[Val: Gen, Ergebnis: AdvL]>, Ergebnis: AdvL];
[LemOFL: "an", Klasse: Präp, Px: [OFl: ja],
 Struktur: <[Val: Dat, Ergebnis: AdvL], [Val: Akk, Ergebnis: AdvD]>];
[LemOFL: "anhand", Klasse: Präp,
 Struktur: <[Val: Gen, Ergebnis: AdvM]>];
[LemOFL: "auf", Klasse: Präp, Px: [OFl: ja],
 Struktur: <[Val: Dat, Ergebnis: AdvL], [Val: Akk, Ergebnis: AdvD]>];
[LemOFL: "aufgrund", Klasse: Präp,
 Struktur: <[Val: Gen, Ergebnis: AdvK]>];
[LemOFL: "aus", Klasse: Präp, Px: [OFl: ja],
 Struktur: <[Val: Dat, Ergebnis: AdvD]>];
[LemOFL: "außer", Klasse: Präp,
 Struktur: <[Val: Dat, Ergebnis: AdvM],
            [Val: Akk, Ergebnis: AdvM],
            [Val: Gen, Ergebnis: AdvM]>];
[LemOFL: "bei", Klasse: Präp, Px: [OFl: ja],
 Struktur: <[Val: Dat, Ergebnis: AdvL]>];
[LemOFL: "durch", Klasse: Präp, Px: [OFl: ja],
 Struktur: <[Val: Akk, Ergebnis: AdvD], [Val: Akk, Ergebnis: AdvK]>];
[LemOFL: "entgegen", Klasse: Präp, Px: [OFl: ja],
 Struktur: <[Val: Gen, Ergebnis: AdvM]>];
[LemOFL: "frühestens", Klasse: Präp,
 Struktur: <[Val: AdvT, Ergebnis: AdvT]>];
[LemOFL: "für", Klasse: Präp, Px: [OFl: ja],
 Struktur: <[Val: Akk, Ergebnis: AdvM]>];
[LemOFL: "gegen", Klasse: Präp, Px: [OFl: ja],
 Struktur: <[Val: Akk, Ergebnis: AdvD]>];
[LemOFL: "gegenüber", Klasse: Präp, Px: [OFl: ja],
 Struktur: <[Val: Dat, Ergebnis: AdvL]>, Ergebnis: AdvL];
[LemOFL: "gemäß", Klasse: Präp,
 Struktur: <[Val: Gen, Ergebnis: AdvM]>];
[LemOFL: "hinsichtlich", Klasse: Präp,
 Struktur: <[Val: Gen, Ergebnis: AdvM]>];
[LemOFL: "hinter", Klasse: Präp, Px: [OFl: ja],
 Struktur: <[Val: Dat, Ergebnis: AdvL], [Val: Akk, Ergebnis: AdvD]>];
[LemOFL: "in", Klasse: Präp, Px: [OFl: ja],
 Struktur: <[Val: Dat, Ergebnis: AdvL], [Val: Akk, Ergebnis: AdvD]>];
[LemOFL: "innerhalb", Klasse: Präp,
 Struktur: <[Val: Gen, Ergebnis: AdvL]>];
[LemOFL: "ins", Klasse: Präp,
 Struktur: <[Adjektiv: E,
             Genus: Neutrum,
             KasusNumerus: NomSg+DatSg+AkkSg,
             Ergebnis: AdvD]>];
[LemOFL: "mit", Klasse: Präp, Px: [OFl: ja],
 Struktur: <[Val: Dat, Ergebnis: AdvM]>];
[LemOFL: "mittels", Klasse: Präp,
 Struktur: <[Val: Gen, Ergebnis: AdvM]>];
[LemOFL: "nach", Klasse: Präp, Px: [OFl: ja],
 Struktur: <[Val: Dat, Ergebnis: AdvT], [Val: Dat, Ergebnis: Advm]>];
[LemOFL: "neben", Klasse: Präp, Px: [OFl: ja],
 Struktur: <[Val: Dat, Ergebnis: AdvL], [Val: Akk, Ergebnis: AdvL]>];
[LemOFL: "ohne", Klasse: Präp,
 Struktur: <[Val: Akk, Ergebnis: AdvM]>];
[LemOFL: "seit", Klasse: Präp,
 Struktur: <[Val: Dat, Ergebnis: AdvT]>];
[LemOFL: "statt", Klasse: Präp, Px: [OFl: ja],
 Struktur: <[Val: Gen, Ergebnis: AdvM]>];
[LemOFL: "trotz", Klasse: Präp,
 Struktur: <[Val: Gen, Ergebnis: AdvM]>];
[LemOFL: "um", Klasse: Präp, Px: [OFl: ja],
 Struktur: <[Val: Akk, Ergebnis: AdvD]>];
[LemOFL: "unter", Klasse: Präp, Px: [OFl: ja],
 Struktur: <[Val: Dat, Ergebnis: AdvL], [Val: Akk, Ergebnis: AdvD]>];
[LemOFL: "von", Klasse: Präp,
 Struktur: <[Val: Dat, Ergebnis: AdvD]>];
[LemOFL: "vor", Klasse: Präp, Px: [OFl: ja],
 Struktur: <[Val: Dat, Ergebnis: AdvL], [Val: Akk, Ergebnis: AdvD]>];
[LemOFL: "während", Klasse: Präp,
 Struktur: <[Val: Gen, Ergebnis: AdvT]>];
[LemOFL: "wegen", Klasse: Präp,
 Struktur: <[Val: Gen, Ergebnis: AdvK]>];
[LemOFL: "weitaus", Klasse: Präp,
 Struktur: <[Val: Adverb, Ergebnis: AdvM]>];
[LemOFL: "ziemlich", Klasse: Präp,
 Struktur: <[Val: Adverb, Ergebnis: AdvM]>];
[LemOFL: "zirka", Klasse: Präp,
 Struktur: <[Val: Adverb, Ergebnis: AdvM]>];
[LemOFL: "zu", Klasse: Präp, Px: [OFl: ja],
 Struktur: <[Val: Adverb, Ergebnis: AdvM], [Val: Dat, Ergebnis: AdvD]>];
[LemOFL: "zugunsten", Klasse: Präp,
 Struktur: <[Val: Gen, Ergebnis: AdvM]>];
[LemOFL: "zwischen", Klasse: Präp, Px: [OFl: ja],
 Struktur: <[Val: Dat, Ergebnis: AdvL], [Val: Akk, Ergebnis: AdvD]>];
[LemOFL: "über", Klasse: Präp, Px: [OFl: ja],
```

```
 Struktur: <[Val: Dat, Ergebnis: AdvL], [Val: Akk, Ergebnis: AdvD]>];
[LemOFL: "überaus", Klasse: Präp,
 Struktur: <[Val: Adverb, Ergebnis: AdvM]>];


#Pronomina

[LemOFL: "allem", Klasse: Pron, Ergebnis: Dat];
[LemOFL: "alles", Klasse: Pron, Ergebnis: Sg3+Akk];
[LemOFL: "denen", Klasse: Pron, Ergebnis: Dat];
[LemOFL: "derer", Klasse: Pron, Ergebnis: Gen];
[LemOFL: "dies", Klasse: Pron, Ergebnis: Sg3+Akk];
[LemOFL: "du", Klasse: Pron, Ergebnis: Sg2, Struktur: <
 [Adjektiv: er, Genus: Maskulinum, KasusNumerus: NomSg, Ergebnis: Sg2],
 [Adjektiv: es, Genus: Neutrum, KasusNumerus: NomSg, Ergebnis: Sg2],
 [Adjektiv: e, Genus: Femininum, KasusNumerus: NomSg, Ergebnis: Sg2]>];
[LemOFL: "er", Klasse: Pron, Ergebnis: Sg3];
[LemOFL: "es", Klasse: Pron, Ergebnis: Sg3+Akk];
[LemOFL: "etwas", Klasse: Pron, Ergebnis: Sg3+Akk];
[LemOFL: "ich", Klasse: Pron, Ergebnis: Sg1, Struktur: <
 [Adjektiv: er, Genus: Maskulinum, KasusNumerus: NomSg, Ergebnis: Sg1],
 [Adjektiv: es, Genus: Neutrum, KasusNumerus: NomSg, Ergebnis: Sg1],
 [Adjektiv: e, Genus: Femininum, KasusNumerus: NomSg, Ergebnis: Sg1]>];
[LemOFL: "ihm", Klasse: Pron, Ergebnis: DatSg];
[LemOFL: "ihn", Klasse: Pron, Ergebnis: AkkSg];

# Geändert auf Manfreds Wunsch 1.2.96
# Neues Kategoriesegment: DatSg+Pl2
[LemOFL: "ihr", Klasse: Pron, Struktur: <
 [Adjektiv: ER, Genus: Maskulinum, KasusNumerus: NomSg, Ergebnis: Sg3],
 [Adjektiv: ES, Genus: Neutrum, KasusNumerus: NomSg+AkkSg, Ergebnis: Sg3+Akk],
 [Adjektiv: en, Genus: Kein_Genus, KasusNumerus: NomPl, Ergebnis: Pl2]>,
 Ergebnis: DatSg+Pl2];


[LemOFL: "ihnen", Klasse: Pron, Ergebnis: Dat];
[LemOFL: "irgendetwas", Klasse: Pron, Ergebnis: Sg3+Akk];
[LemOFL: "jemand", Klasse: Pron, Ergebnis: Sg3];
[LemOFL: "jemandem", Klasse: Pron, Ergebnis: Dat];
[LemOFL: "jemandes", Klasse: Pron, Ergebnis: Gen];
[LemOFL: "man", Klasse: Pron, Ergebnis: Sg3];
[LemOFL: "mich", Klasse: Pron, Ergebnis: AkkSg];
[LemOFL: "mir", Klasse: Pron, Ergebnis: Dat];
[LemOFL: "nichts", Klasse: Pron, Ergebnis: Sg3+Akk];
[LemOFL: "niemand", Klasse: Pron, Ergebnis: Sg3+Akk];
[LemOFL: "niemandem", Klasse: Pron, Ergebnis: Dat];
[LemOFL: "niemandes", Klasse: Pron, Ergebnis: Gen];
[LemOFL: "sich", Klasse: Pron, Ergebnis: AkkRefl];
[LemOFL: "sie", Klasse: Pron, Ergebnis: Sg3+Akk, Struktur: <
 [Adjektiv: er, Genus: Maskulinum, KasusNumerus: NomSg, Ergebnis: Pl3],
 [Adjektiv: e,  Genus: Femininum, KasusNumerus: NomSg, Ergebnis: Pl3],
 [Adjektiv: en, Genus: Kein_Genus, KasusNumerus: NomSg, Ergebnis: Pl3]>];
[LemOFL: "wir", Klasse: Pron, Ergebnis: Pl1, Struktur: <
 [Adjektiv: en, Genus: Kein_Genus, KasusNumerus: NomPl, Ergebnis: Pl1]>];

[LemOFL: "irgendwelchem", Klasse: Pron, Ergebnis: Dat];
[LemOFL: "irgendwelches", Klasse: Pron, Ergebnis: Sg3+Akk];
```

E.8 Interpunktionszeichen

```
#                                                                      #
# (c) 1995 Abteilung für Computerlinguistik der Universität            #
# Erlangen-Nürnberg                                                    #
#                                                                      #

# Interpunktionszeichen

[LemOFL: ".", Klasse: Interpunkt, Ergebnis: Punkt];
[LemOFL: "?", Klasse: Interpunkt, Ergebnis: Fragezeichen];
[LemOFL: ",", Klasse: Interpunkt, Ergebnis: Komma];
[LemOFL: ";", Klasse: Interpunkt, Ergebnis: Semikolon];
[LemOFL: ":", Klasse: Interpunkt, Ergebnis: Doppelpunkt];
[LemOFL: "!", Klasse: Interpunkt, Ergebnis: Ausrufezeichen];
[LemOFL: "’", Klasse: Interpunkt, Ergebnis: Anführungszeichen];
[LemOFL: "/", Klasse: Interpunkt, Ergebnis: Schrägstrich];
[LemOFL: "(", Klasse: Interpunkt, Ergebnis: Klammer_auf];
[LemOFL: ")", Klasse: Interpunkt, Ergebnis: Klammer_zu];
[LemOFL: "-", Klasse: Interpunkt, Ergebnis: Bindestrich];
[LemOFL: "&", Klasse: Interpunkt, Ergebnis: Kaufmund];
[LemOFL: "*", Klasse: Interpunkt, Ergebnis: Multzeichen];
[LemOFL: "+", Klasse: Interpunkt, Ergebnis: Pluszeichen];
[LemOFL: "·", Klasse: Interpunkt, Ergebnis: Zeichen];
[LemOFL: "_", Klasse: Interpunkt, Ergebnis: Unterstrich];
```

E.9 Zahlwörter

```
#                                                                    #
# (c) 1995 Abteilung für Computerlinguistik der Universität          #
# Erlangen-Nürnberg                                                  #
#                                                                    #

[ LemOFl: "0", Klasse: Num ];
[ LemOFl: "1", Klasse: Num ];
[ LemOFl: "2", Klasse: Num ];
[ LemOFl: "3", Klasse: Num ];
[ LemOFl: "4", Klasse: Num ];
[ LemOFl: "5", Klasse: Num ];
[ LemOFl: "6", Klasse: Num ];
[ LemOFl: "7", Klasse: Num ];
[ LemOFl: "8", Klasse: Num ];
[ LemOFl: "9", Klasse: Num ];

[ LemOFL: "eins", Klasse: Num];
[ LemOFL: "zwei", Klasse: Num];
[ LemOFL: "drei", Klasse: Num];
[ LemOFL: "vier", Klasse: Num];
[ LemOFL: "fünf", Klasse: Num];
[ LemOFL: "sechs", Klasse: Num];
[ LemOFL: "sieben", Klasse: Num];
[ LemOFL: "acht", Klasse: Num];
[ LemOFL: "neun", Klasse: Num];
[ LemOFL: "zehn", Klasse: Num];
[ LemOFL: "elf", Klasse: Num];
[ LemOFL: "zwölf", Klasse: Num];
[ LemOFL: "dreizehn", Klasse: Num];
[ LemOFL: "vierzehn", Klasse: Num];
[ LemOFL: "fünfzehn", Klasse: Num];
[ LemOFL: "sechzehn", Klasse: Num];
[ LemOFL: "siebzehn", Klasse: Num];
[ LemOFL: "achtzehn", Klasse: Num];
[ LemOFL: "neunzehn", Klasse: Num];
[ LemOFL: "zwanzig", Klasse: Num];
[ LemOFL: "dreißig", Klasse: Num];
[ LemOFL: "vierzig", Klasse: Num];
[ LemOFL: "fünfzig", Klasse: Num];
[ LemOFL: "sechzig", Klasse: Num];
[ LemOFL: "siebzig", Klasse: Num];
[ LemOFL: "achtzig", Klasse: Num];
[ LemOFL: "neunzig", Klasse: Num];
[ LemOFL: "hundert", Klasse: Num];
[ LemOFL: "einhundert", Klasse: Num];
[ LemOFL: "zweihundert", Klasse: Num];
[ LemOFL: "dreihundert", Klasse: Num];
[ LemOFL: "vierhundert", Klasse: Num];
[ LemOFL: "fünfhundert", Klasse: Num];
[ LemOFL: "sechshundert", Klasse: Num];
[ LemOFL: "siebenhundert", Klasse: Num];
[ LemOFL: "achthundert", Klasse: Num];
[ LemOFL: "neunhundert", Klasse: Num];
[ LemOFL: "tausend", Klasse: Num];
[ LemOFL: "eintausend", Klasse: Num];

[ LemOFL: "eineinhalb", Klasse: Num ];
[ LemOFL: "zweieinhalb", Klasse: Num ];
```

E.10 Abkürzungen

```
#                                                                    #
# (c) 1995 Abteilung für Computerlinguistik der Universität          #
# Erlangen-Nürnberg                                                  #
#                                                                    #

[ LemOFL: "afp" , Klasse: Abk ] ;
[ LemOFL: "ap" , Klasse: Abk ] ;
[ LemOFl: "abs", Klasse: Abk ];
[ LemOFl: "adac", Klasse: Abk ];
[ LemOFl: "bmw", Klasse: Abk ];
[ LemOFl: "bzw", Klasse: Abk ];
[ LemOFl: "ca", Klasse: Abk ];
[ LemOFl: "cdu", Klasse: Abk ];
[ LemOFl: "cssr", Klasse: Abk ];
[ LemOFl: "csu", Klasse: Abk ];
[ LemOFl: "ddr", Klasse: Abk ];
[ LemOFl: "djk", Klasse: Abk ];
[ LemOFl: "dm" , Klasse: Abk ];
[ LemOFl: "dpa", Klasse: Abk ];
[ LemOFl: "etw", Klasse: Abk ];
[ LemOFl: "evtl", Klasse: Abk ];
[ LemOFl: "ewg", Klasse: Abk ];
[ LemOFl: "fam", Klasse: Abk ];
[ LemOFl: "fc", Klasse: Abk ];
[ LemOFl: "fdp", Klasse: Abk ];
[ LemOFl: "frl", Klasse: Abk ];
[ LemOFl: "ges", Klasse: Abk ];
[ LemOFl: "gg" , Klasse: Abk ];
[ LemOFl: "gmbh", Klasse: Abk ];
[ LemOFl: "grd", Klasse: Abk ];
[ LemOFl: "grüne", Klasse: Abk ];
[ LemOFl: "gus", Klasse: Abk ];
[ LemOFl: "hkl", Klasse: Abk ];
```

```
[ LemOFl: "inkl", Klasse: Abk ];
[ LemOFl: "kg", Klasse: Abk ];
[ LemOFl: "km", Klasse: Abk ];
[ LemOFl: "ksze", Klasse: Abk ];
[ LemOFl: "kve", Klasse: Abk ];
[ LemOFl: "l-fl", Klasse: Abk ];
[ LemOFl: "mio", Klasse: Abk ];
[ LemOFl: "mrd", Klasse: Abk ];
[ LemOFl: "nbg", Klasse: Abk ];
[ LemOFl: "pds", Klasse: Abk ];
[ LemOFl: "pkt", Klasse: Abk ];
[ LemOFl: "plo", Klasse: Abk ];
[ LemOFl: "ruh", Klasse: Abk ];
[ LemOFl: "sg", Klasse: Abk ];
[ LemOFl: "spd", Klasse: Abk ];
[ LemOFl: "ssd", Klasse: Abk ];
[ LemOFl: "str", Klasse: Abk ];
[ LemOFl: "sv", Klasse: Abk ];
[ LemOFl: "tel", Klasse: Abk ];
[ LemOFl: "tsv", Klasse: Abk ];
[ LemOFl: "tv", Klasse: Abk ];
[ LemOFl: "tüv", Klasse: Abk ];
[ LemOFl: "udssr", Klasse: Abk ];
[ LemOFl: "uhf", Klasse: Abk ];
[ LemOFl: "usa", Klasse: Abk ];
[ LemOFl: "usf", Klasse: Abk ];
[ LemOFl: "usw", Klasse: Abk ];
[ LemOFl: "usw", Klasse: Abk ];
[ LemOFl: "versch", Klasse: Abk ];
[ LemOFl: "vhf", Klasse: Abk ];
[ LemOFl: "wfl", Klasse: Abk ];
[ LemOFl: "whg", Klasse: Abk ];
[ LemOFl: "wrv", Klasse: Abk ];
[ LemOFl: "wv" , Klasse: Abk ];
[ LemOFl: "zdf", Klasse: Abk ];
```

E.11 Irreguläre Paradigmen

```
#                                                                      #
# (c) 1995 Abteilung für Computerlinguistik der Universität            #
# Erlangen-Nürnberg                                                    #
#                                                                      #

[LemOFl: "gut",
 Paradigma: Adjektiv,
 Wortform: <[LemOFl: "gut", AdjFlex: AdjFlex_ad],
            [LemOFl: "bess", AdjFlex: AdjFlex_k],
            [LemOFl: "be", AdjFlex: AdjFlex_s, PhonEnd: unmarkiert]>];

[LemOFl: "hoch",
 Paradigma: Adjektiv,
 Wortform: <[LemOFl: "hoch", AdjFlex: AdjFlex_a],
            [LemOFl: "hoh", AdjFlex: AdjFlex_d],
            [LemOFl: "höh", AdjFlex: AdjFlex_k],
            [LemOFl: "höch", AdjFlex: AdjFlex_s, PhonEnd: unmarkiert]>];

[LemOFl: "bedürfen",
 Paradigma: Verb,
 Valenzen: <<Gen>>,
 Wortform: <[LemOFl: "bedarf",
             Konjugation: <[Subjekt: Sg13, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "bedürfe",
             Konjugation: <[Subjekt: Sg13, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "bedarfst",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "bedürfest",
             Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "bedürfen",
             Konjugation: <[Subjekt: Pl13, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "bedürft",
             Konjugation: <[Subjekt: Pl2, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "bedürfet",
             Konjugation: <[Subjekt: Pl2, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "bedurfte",
             Konjugation: <[Subjekt: Sg13, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "bedurftest",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "bedurften",
             Konjugation: <[Subjekt: Pl13, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "bedurftet",
             Konjugation: <[Subjekt: Pl2, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "bedürfte",
             Konjugation: <[Subjekt: Sg13, Modus: Indikativ]>,
             Tempus: Präteritum],
```

```
            [LemOFl: "bedürftest",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "bedürften",
             Konjugation: <[Subjekt: Pl13, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "bedürftet",
             Konjugation: <[Subjekt: Pl2, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "bedurft",
             Klasse: Adjektiv,
             AdjFlex: AdjFlex_d,
             PII-Auxil: <PII-haben>]>];

[LemOFl: "dürfen",
 Paradigma: Verb,
 Valenzen: <<Inf>, <AdvD>>,
 Wortform: <[LemOFl: "darf",
             Konjugation: <[Subjekt: Sg13, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "dürfe",
             Konjugation: <[Subjekt: Sg13, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "darfst",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "dürfest",
             Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "dürfen",
             Konjugation: <[Subjekt: Pl13, Modus: Indikativ+Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "dürft",
             Konjugation: <[Subjekt: Pl2, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "dürfet",
             Konjugation: <[Subjekt: Pl2, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "durfte",
             Konjugation: <[Subjekt: Sg13, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "durftest",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "durften",
             Konjugation: <[Subjekt: Pl13, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "durftet",
             Konjugation: <[Subjekt: Pl2, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "dürfte",
             Konjugation: <[Subjekt: Sg13, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "dürftest",
             Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "dürften",
             Konjugation: <[Subjekt: Pl13, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "dürftet",
             Konjugation: <[Subjekt: Pl2, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "gedurft",
             Klasse: Adjektiv,
             AdjFlex: AdjFlex_d,
             PII-Auxil: <PII-haben>]>];

[LemOFl: "haben",
 Paradigma: Verb,
 Valenzen: <<PII-haben>, <Akk>>,
 Wortform: <[LemOFl: "habe",
             Konjugation: <[Subjekt: Sg1, Modus: Indikativ],
                           [Subjekt: Sg13, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "hast",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "habest",
             Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "hat",
             Konjugation: <[Subjekt: Sg3, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "haben",
             Konjugation: <[Subjekt: Pl13, Modus: Indikativ+Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "habt",
             Konjugation: <[Subjekt: Pl2, Modus: Indikativ+Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "hatte",
             Konjugation: <[Subjekt: Sg13, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "hattest",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "hatten",
             Konjugation: <[Subjekt: Pl13, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "hattet",
```

```
             Konjugation: <[Subjekt: Pl2, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "hätte",
             Konjugation: <[Subjekt: Sg13, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "hättest",
             Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "hätten",
             Konjugation: <[Subjekt: Pl13, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "hättet",
             Konjugation: <[Subjekt: Pl2, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "gehabt",
             Klasse: Adjektiv,
             AdjFlex: AdjFlex_d,
             PII-Auxil: <PII-haben>]>];

[LemOFl: "hervortun",
 Paradigma: Verb,
 Valenzen: <<Akk>>,
 Wortform: <[LemOFl: "hervortue",
             Konjugation: <[Subjekt: Sg13, Modus: Indikativ+Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "hervortust",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "hervortuest",
             Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "hervortut",
             Konjugation: <[Subjekt: Sg3+Pl2, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "hervortun",
             Konjugation: <[Subjekt: Pl13, Modus: Indikativ+Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "hervortuen",
             Konjugation: <[Subjekt: Pl13, Modus: Indikativ+Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "hervortuet",
             Konjugation: <[Subjekt: Pl2, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "hervortat",
             Konjugation: <[Subjekt: Sg13, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "hervoratest",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "hervortaten",
             Konjugation: <[Subjekt: Pl13, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "hervortatet",
             Konjugation: <[Subjekt: Pl2, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "hervortäte",
             Konjugation: <[Subjekt: Sg13, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "hervortätest",
             Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "hervortäte",
             Konjugation: <[Subjekt: Sg13, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "hervortäten",
             Konjugation: <[Subjekt: Pl13, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "hervortätet",
             Konjugation: <[Subjekt: Pl2, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "hervorgetan",
             Klasse: Adjektiv,
             AdjFlex: AdjFlex_d,
             PII-Auxil: <PII-haben>]>];

[LemOFl: "können",
 Paradigma: Verb,
 Valenzen: <<Inf>, <Akk>, <AdvD>>,
 Wortform: <[LemOFl: "kann",
             Konjugation: <[Subjekt: Sg13, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "könne",
             Konjugation: <[Subjekt: Sg13, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "kannst",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "könnest",
             Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "können",
             Konjugation: <[Subjekt: Pl13, Modus: Indikativ+Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "könnt",
             Konjugation: <[Subjekt: Pl2, Modus: Indikativ+Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "konnte",
             Konjugation: <[Subjekt: Sg13, Modus: Indikativ]>,
             Tempus: Präteritum],
```

```
            [LemOFl: "konntest",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "konnten",
             Konjugation: <[Subjekt: Pl13, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "konntet",
             Konjugation: <[Subjekt: Pl2, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "könnte",
             Konjugation: <[Subjekt: Sg13, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "könntest",
             Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "könnten",
             Konjugation: <[Subjekt: Pl13, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "könntet",
             Konjugation: <[Subjekt: Pl2, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "gekonnt",
             Klasse: Adjektiv,
             AdjFlex: AdjFlex_d,
             PII-Auxil: <PII-haben>]>];

[LemOFl: "mögen",
 Paradigma: Verb,
 Valenzen: <<Inf>, <Akk>>,
 Wortform: <[LemOFl: "mag",
             Konjugation: <[Subjekt: Sg13, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "möge",
             Konjugation: <[Subjekt: Sg13, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "magst",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "mögest",
             Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "mögen",
             Konjugation: <[Subjekt: Pl13, Modus: Indikativ+Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "mögt",
             Konjugation: <[Subjekt: Pl2, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "möget",
             Konjugation: <[Subjekt: Pl2, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "mochte",
             Konjugation: <[Subjekt: Sg13, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "mochtest",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "mochten",
             Konjugation: <[Subjekt: Pl13, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "mochtet",
             Konjugation: <[Subjekt: Pl2, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "möchte",
             Konjugation: <[Subjekt: Sg13, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "möchtest",
             Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "möchten",
             Konjugation: <[Subjekt: Pl13, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "möchtet",
             Konjugation: <[Subjekt: Pl2, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "gemocht",
             Klasse: Adjektiv,
             AdjFlex: AdjFlex_d,
             PII-Auxil: <PII-haben>]>];

[LemOFl: "müssen",
 Paradigma: Verb,
 Valenzen: <<Inf>, <AdvD>>,
 Wortform: <[LemOFl: "muß",
             Konjugation: <[Subjekt: Sg13, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "müsse",
             Konjugation: <[Subjekt: Sg13, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "mußt",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "müssest",
             Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "müssen",
             Konjugation: <[Subjekt: Pl13, Modus: Indikativ+Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "müßt",
             Konjugation: <[Subjekt: Pl2, Modus: Indikativ]>,
```

```
             Tempus: Präsens],
            [LemOFl: "müsset",
             Konjugation: <[Subjekt: Pl2, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "mußte",
             Konjugation: <[Subjekt: Sg13, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "mußtest",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "mußten",
             Konjugation: <[Subjekt: Pl13, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "mußtet",
             Konjugation: <[Subjekt: Pl2, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "müßte",
             Konjugation: <[Subjekt: Sg13, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "müßtest",
             Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "müßten",
             Konjugation: <[Subjekt: Pl13, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "müßtet",
             Konjugation: <[Subjekt: Pl2, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "gemußt",
             Klasse: Adjektiv,
             AdjFlex: AdjFlex_d,
             PII-Auxil: <PII-haben>]>];

[LemOFl: "sein",
 Paradigma: Verb,
 Valenzen: <<PII-sein>, <Nom>, <AdvL>>,
 Wortform: <[LemOFl: "sein",
             Infinitiv: ja],
            [LemOFl: "bin",
             Konjugation: <[Subjekt: Sg1, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "sei",
             Konjugation: <[Subjekt: Sg13, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "bist",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "seiest",
             Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "ist",
             Konjugation: <[Subjekt: Sg3, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "sind",
             Infinitiv: nein,
             Konjugation: <[Subjekt: Pl13, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "seien",
             Konjugation: <[Subjekt: Pl13, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "seid",
             Konjugation: <[Subjekt: Pl2, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "seiet",
             Konjugation: <[Subjekt: Pl2, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "war",
             Konjugation: <[Subjekt: Sg13, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "warst",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "waren",
             Konjugation: <[Subjekt: Pl13, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "wart",
             Konjugation: <[Subjekt: Pl2, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "wäre",
             Konjugation: <[Subjekt: Sg13, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "wärst",
             Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "wären",
             Konjugation: <[Subjekt: Pl13, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "wäret",
             Konjugation: <[Subjekt: Pl2, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "wärt",
             Konjugation: <[Subjekt: Pl2, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "gewesen",
             Klasse: Adjektiv,
             AdjFlex: AdjFlex_d,
             PII-Auxil: <PII-sein>]>];

[LemOFl: "tun",
```

```
 Paradigma: Verb,
 Valenzen: <<Inf>, <Akk>>,
 Wortform: <[LemOFl: "tue",
             Konjugation: <[Subjekt: Sg13, Modus: Indikativ+Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "tust",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "tuest",
             Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "tut",
             Konjugation: <[Subjekt: Sg3+Pl2, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "tun",
             Konjugation: <[Subjekt: Pl13, Modus: Indikativ+Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "tuen",
             Konjugation: <[Subjekt: Pl13, Modus: Indikativ+Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "tuet",
             Konjugation: <[Subjekt: Pl2, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "tat",
             Konjugation: <[Subjekt: Sg13, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "tatest",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "taten",
             Konjugation: <[Subjekt: Pl13, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "tatet",
             Konjugation: <[Subjekt: Pl2, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "täte",
             Konjugation: <[Subjekt: Sg13, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "tätest",
             Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "täten",
             Konjugation: <[Subjekt: Pl13, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "tätet",
             Konjugation: <[Subjekt: Pl2, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "getan",
             Klasse: Adjektiv,
             AdjFlex: AdjFlex_d,
             PII-Auxil: <PII-haben>]>];

[LemOFl: "vermögen",
 Paradigma: Verb,
 Valenzen: <<Inf>>,
 Wortform: <[LemOFl: "vermag",
             Konjugation: <[Subjekt: Sg13, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "vermöge",
             Konjugation: <[Subjekt: Sg13, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "vermagst",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "vermögest",
             Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "vermögen",
             Konjugation: <[Subjekt: Pl13, Modus: Indikativ+Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "vermögt",
             Konjugation: <[Subjekt: Pl2, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "vermöget",
             Konjugation: <[Subjekt: Pl2, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "vermochte",
             Konjugation: <[Subjekt: Sg13, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "vermochtest",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "vermochten",
             Konjugation: <[Subjekt: Sg13, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "vermochtet",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "vermöchte",
             Konjugation: <[Subjekt: Sg13, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "vermöchtest",
             Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "vermöchten",
             Konjugation: <[Subjekt: Sg13, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "vermöchtet",
             Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>,
```

```
             Tempus: Präteritum],
            [LemOFl: "vermocht",
             Klasse: Adjektiv,
             AdjFlex: AdjFlex_d,
             PII-Auxil: <PII-haben>]>];

[LemOFl: "werden",
 Paradigma: Verb,
 Valenzen: <<Inf>, <Nom>>,
 Wortform: <[LemOFl: "werde",
             Konjugation: <[Subjekt: Sg13, Modus: Indikativ+Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "wirst",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "wird",
             Konjugation: <[Subjekt: Sg3, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "werden",
             Konjugation: <[Subjekt: Pl13, Modus: Indikativ+Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "werdet",
             Konjugation: <[Subjekt: Pl2, Modus: Indikativ+Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "wurde",
             Konjugation: <[Subjekt: Sg13, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "wurdest",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "wurden",
             Konjugation: <[Subjekt: Pl13, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "wurdet",
             Konjugation: <[Subjekt: Pl2, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "würde",
             Konjugation: <[Subjekt: Sg13, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "würdest",
             Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "würden",
             Konjugation: <[Subjekt: Pl13, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "würdet",
             Konjugation: <[Subjekt: Pl2, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "geworden",
             Klasse: Adjektiv,
             AdjFlex: AdjFlex_d,
             PII-Auxil: <PII-sein>]>];

[LemOFl: "wissen",
 Paradigma: Verb,
 Valenzen: <<DassNebs>, <Akk>>,
 Wortform: <[LemOFl: "weiß",
             Konjugation: <[Subjekt: Sg13, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "wisse",
             Konjugation: <[Subjekt: Sg13, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "weißt",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "wissest",
             Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "wissen",
             Konjugation: <[Subjekt: Pl13, Modus: Indikativ+Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "wißt",
             Konjugation: <[Subjekt: Pl2, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "wisset",
             Konjugation: <[Subjekt: Pl2, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "wußte",
             Konjugation: <[Subjekt: Sg13, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "wußtest",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "wußten",
             Konjugation: <[Subjekt: Sg13, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "wußtet",
             Konjugation: <[Subjekt: Pl2, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "wüßte",
             Konjugation: <[Subjekt: Sg13, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "wüßtest",
             Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "wüßten",
             Konjugation: <[Subjekt: Sg13, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "wüßtet",
```

```
             Konjugation: <[Subjekt: Pl2, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "gewußt",
             Klasse: Adjektiv,
             AdjFlex: AdjFlex_d,
             PII-Auxil: <PII-haben>]>];

[LemOFl: "wollen",
 Paradigma: Verb,
 Valenzen: <<Inf>, <Akk>, <AdvD>>,
 Wortform: <[LemOFl: "will",
             Konjugation: <[Subjekt: Sg13, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "wolle",
             Konjugation: <[Subjekt: Sg13, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "willst",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "wollest",
             Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "wollen",
             Konjugation: <[Subjekt: Pl13, Modus: Indikativ+Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "wollt",
             Konjugation: <[Subjekt: Pl2, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "wollet",
             Konjugation: <[Subjekt: Pl2, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "wollte",
             Konjugation: <[Subjekt: Sg13, Modus: Indikativ+Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "wolltest",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ+Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "wollten",
             Konjugation: <[Subjekt: Pl13, Modus: Indikativ+Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "wolltet",
             Konjugation: <[Subjekt: Pl2, Modus: Indikativ+Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "gewollt",
             Klasse: Adjektiv,
             AdjFlex: AdjFlex_d,
             PII-Auxil: <PII-haben>]>];

[LemOFl: "sollen",
 Paradigma: Verb,
 Valenzen: <<Inf>>,
 Präfixe: <[PxOFl: "zurück",
            Valenzen: <<PräpErg>, <>>,
            PII-Auxil: <PII-haben>,
            PII-Ge: Ge],
           [PxOFl: "weiter",
            Valenzen: <<PräpErg>, <>>,
            PII-Auxil: <PII-haben>,
            PII-Ge: Ge],
           [PxOFl: "weg",
            Valenzen: <<PräpErg>, <>>,
            PII-Auxil: <PII-haben>,
            PII-Ge: Ge],
           [PxOFl: "mit",
            Valenzen: <<PräpErg>, <>>,
            PII-Auxil: <PII-haben>,
            PII-Ge: Ge],
           [PxOFl: "hin",
            Valenzen: <<PräpErg>, <>>,
            PII-Auxil: <PII-haben>,
            PII-Ge: Ge],
           [PxOFl: "her",
            Valenzen: <<PräpErg>, <>>,
            PII-Auxil: <PII-haben>,
            PII-Ge: Ge],
           [PxOFl: "durch",
            Valenzen: <<PräpErg>, <>>,
            PII-Auxil: <PII-haben>,
            PII-Ge: Ge]>,
 Wortform: <[LemOFl: "soll",
             Konjugation: <[Subjekt: Sg13, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "solle",
             Konjugation: <[Subjekt: Sg13, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "sollst",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "sollest",
             Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "sollen",
             Konjugation: <[Subjekt: Pl13, Modus: Indikativ+Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "sollt",
             Konjugation: <[Subjekt: Pl2, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "sollet",
             Konjugation: <[Subjekt: Pl2, Modus: Konjunktiv]>,
```

```
             Tempus: Präsens],
            [LemOFl: "sollte",
             Konjugation: <[Subjekt: Sg13, Modus: Indikativ+Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "solltest",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ+Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "sollten",
             Konjugation: <[Subjekt: Pl13, Modus: Indikativ+Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "solltet",
             Konjugation: <[Subjekt: Pl2, Modus: Indikativ+Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "gesollt",
             Klasse: Adjektiv,
             AdjFlex: AdjFlex_d,
             PII-Auxil: <PII-haben>]>];

[LemOFl: "laden",
 Paradigma: Verb,
 Valenzen: <<Akk>>,
 Präfixe: <[PxOFl: "zu",
            Valenzen: <<Akk>, <Akk, DatBen>>,
            PII-Auxil: <PII-haben>,
            PII-Ge: Ge],
           [PxOFl: "vor",
            Valenzen: <<Akk>>,
            PII-Auxil: <PII-haben>,
            PII-Ge: Ge],
           [PxOFl: "um",
            Valenzen: <<Akk>,<Akk, DatBen>>,
            PII-Auxil: <PII-haben>,
            PII-Ge: Ge],
           [PxOFl: "hinein",
            Valenzen: <<Akk>,<Akk, DatBen>>,
            PII-Auxil: <PII-haben>,
            PII-Ge: Ge],
           [PxOFl: "ein",
            Valenzen: <<Akk>, <PräpErg>, <>>,
            PII-Auxil: <PII-haben>,
            PII-Ge: Ge],
           [PxOFl: "drauf",
            Valenzen: <<Akk>, <Akk, DatBen>>,
            PII-Auxil: <PII-haben>,
            PII-Ge: Ge],
           [PxOFl: "darauf",
            Valenzen: <<Akk>, <Akk, DatBen>>,
            PII-Auxil: <PII-haben>,
            PII-Ge: Ge],
           [PxOFl: "aus",
            Valenzen: <<Akk>, <Akk, DatBen>>,
            PII-Auxil: <PII-haben>,
            PII-Ge: Ge],
           [PxOFl: "auf",
            Valenzen: <<Akk>, <Akk, DatBen>>,
            PII-Auxil: <PII-haben>,
            PII-Ge: Ge],
           [PxOFl: "ab",
            Valenzen: <<Akk>, <Akk, DatBen>>,
            PII-Auxil: <PII-haben>,
            PII-Ge: Ge]>,
 Wortform: <[LemOFl: "lade",
             Konjugation: <[Subjekt: Sg1, Modus: Indikativ],
                           [Subjekt: Sg13, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "lädst",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "ladest",
             Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "lädt",
             Konjugation: <[Subjekt: Sg1, Modus: Indikativ]>,
             Tempus: Präsens],
            [LemOFl: "laden",
             Konjugation: <[Subjekt: Pl13, Modus: Indikativ+Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "ladet",
             Konjugation: <[Subjekt: Pl2, Modus: Indikativ+Konjunktiv]>,
             Tempus: Präsens],
            [LemOFl: "lud",
             Konjugation: <[Subjekt: Sg13, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "ludest",
             Konjugation: <[Subjekt: Sg2, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "luden",
             Konjugation: <[Subjekt: Pl13, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "ludet",
             Konjugation: <[Subjekt: Pl2, Modus: Indikativ]>,
             Tempus: Präteritum],
            [LemOFl: "lüde",
             Konjugation: <[Subjekt: Sg13, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "lüdest",
             Konjugation: <[Subjekt: Sg2, Modus: Konjunktiv]>,
             Tempus: Präteritum],
            [LemOFl: "lüden",
```

```
 Konjugation: <[Subjekt: Pl13, Modus: Konjunktiv]>,
 Tempus: Präteritum],
[LemOFl: "lüdet",
 Konjugation: <[Subjekt: Pl2, Modus: Konjunktiv]>,
 Tempus: Präteritum],
[LemOFl: "geladen",
 Klasse: Adjektiv,
 AdjFlex: AdjFlex_d,
 PII-Auxil: <PII-haben>]>];
```