gefördert vom
Bundesministerium
für Forschung
und Technologie

Verbmobil
Verbundvorhaben

Substantivflexion in LEX4 — Ein Applikationsbericht

Gunter Gebhardi
Johannes Heinecke

Humboldt Universität zu Berlin

Vm Memo 62
Februar 1995

Februar 1995

Gunter Gebhardi
Johannes Heinecke

Computerlinguistik
Institut für deutsche Sprache und Linguistik
Philosophische Fakultät II
Humboldt-Universität zu Berlin
Unter den Linden 6
10099 Berlin

Tel.: (030) 20192 - 553
Fax: (030) 20192 - 560
e-mail: {gebhardi|heinecke}@compling.hu-berlin.de

Die vorliegende Arbeit wurde im Rahmen des Verbundvorhabens Verbmobil vom Bundesministerium für Forschung und Technologie (BMFT) unter dem Förderkennzeichen 01 IV 101 G gefördert. Die Verantwortung für den Inhalt dieser Arbeit liegt bei den Autoren.

Inhaltsverzeichnis

1 Überblick

Der folgende Report[1] dient

1. der Dokumentation des Kerns von Flexionsregeln für Substantive,
2. der Dokumentation der Implementation dieser Regeln im Lexikonformalismus *LeX4* und
3. als Beispiel für ein einfaches Programm in *LeX4*.

Nach einer kurzen Erläuterung des gewählten Ansatzes zur Flexionsbehandlung wird die gewählte Implementation vorgestellt und erläutert.

2 Hintergrund

„*Das langfristige Ziel des VERBMOBIL-Projektes ist die Entwicklung eines mobilen Systems zur Übersetzung von Verhandlungsdialogen in face–to–face Situationen.*“ [8]. Als ein System zur Behandlung gesprochener natürlicher Sprache wird dieses System linguistisch motivierte und in Teilen begründete Bestandteile umfassen. Einer dieser Bestandteile ist ein Lexikon. Inhalt, Art und Umfang dieses Lexikons werden durch Parameter aus dem Projekt bestimmt. Eine prinzipielle Entscheidung im Projekt ist, das Lexikon als **Vollformenlexikon** anzulegen.

3 Lexikonarchitektur

Die Architektur des Lexikons sieht mehrere unterschiedliche Beschreibungsebenen vor, zwischen denen mit Hilfe des Lexikonformalismus *LeX4* und des Transformationsformalismus *LeX4*TRAFO vermittelt wird: Die Relationen zwischen den *internen* Ebenen sind durch den Lexikonformalismus definiert, die Relation zu den externen, nutzerspezifischen Datenstrukturen[2] erfolgt mit Hilfe des Lexikontransformationsformalismus. Hinter dieser Architektur verbirgt sich die Idee, einerseits weitgehend unabhängig von den Anforderungen der Nutzer an die Struktur der Darstellung ein Lexikon zu entwickeln – mit Hilfe des Lexikonformalismus *LeX4* – und andererseits

[1] Wir danken Ute Ehrlich für ihre druckvolle Unterstützung

[2] Bei dieser Umsetzung geht es um die syntaktische Umformung der Lexikoneinträge. Ein Eintrag in *LeX4* `agr: (num: pl & case: dat)` wird dabei etwa in die Form `agr(pl, dat)` überführt.

das einmal entwickelte Lexikon möglichst frei entsprechend den Nutzeranforderungen anzupassen – Lexikontransformationsformalismus TRAFO. Damit findet die Forderung nach Flexibilität des Lexikons Berücksichtigung im Formalismus zur Modellierung, zum Aufbau, zur Wartung und Pflege des Lexikons und im Formalismus zur Lexikonadaption und macht den sonst üblichen „*Griff in die UNIX-Werkzeugkiste*" überflüssig.

Zwei Ebenen, zwischen denen eine Relation durch *LeV4* definiert ist, werden näher betrachtet: ein interne Ebene, diejenige, auf welcher der Lexikonaufbau vorangetrieben wird, und eine externe Ebene, die dem Vollformenlexikon entspricht. Auf der internen Ebene werden die Flexionsinformationen verankert. Auf diese greifen die Flexionsregeln zu, die die Relationen zum Vollformenlexikon definieren.

4 Flexionsbehandlung für Substantive

4.1 Ausgangsposition

Deutsch ist ein flektierende Sprache. In ein Lexikon für das Deutsche sind daher entsprechende Informationen zur Flexion zu integrieren. Bezüglich der Art der Informationen und der Weise ihrer Integration in das Lexikon gibt es unterschiedliche Auffassungen. Der hier gewählte Ansatz wurde relativ pragmatisch gewählt: Wie lassen sich die erforderlichen Informationen effizient in das zu erstellende Lexikon integrieren? Unter effizient ist insbesondere der Arbeitsaufwand zu verstehen, der notwendig ist, um die Informationen ins Lexikon zu bringen und dort zu bearbeiten; denn über einen längeren Zeitraum betrachtet kann die Pflege und Wartung der Informationen überaus aufwendig werden. Das Ziel bestand also nicht darin, eine systematische morphologische Analyse der Flexion des Deutschen vorzunehmen, sondern die Informationen zu integrieren, die zur Beschreibung aller Vollformen erforderlich sind.

Hinweise zur Flexion finden sich in den Grammatiken und systematischen Lexika des Deutschen. Man kann mit der Aufzählung von Referenzen bei [6] beginnen, sollte sich für die hier zu bearbeitende Aufgabe aber eher an aktuelleren Grammatiken orientieren, wie [5], [7] oder [2] und Lexika wie [1], [9], [10], [3], [11]. Dabei wird die Flexion in Form von Schemata in [5], [7], [9], [10], [11] angegeben. Wenig erstaunlich ist die Ähnlichkeit der Schemata[3]. So ähneln auch die hier definierten Flexionsschemata diesen Formen.

Bei der Angabe der Flexionsinformationen im Lexikoneintrag wird entweder eine

[3]Interessant ist, daß dies oft auch die Beispiele einschließt.

Referenz auf die Flexionsklasse angegeben [9], [11] oder aber werden die markanten Flexionsformen genannt, die eine Bestimmung der jeweiligen Flexionsklasse ermöglichen [1], [3]. Hier wurde ebenfalls die Darstellung durch Referenzen gewählt, da diese Darstellung frei von Redundanzen ist. Bezüglich der angestrebten Effizienz beim Aufbau scheint diese Darstellung jedoch am Anfang der Arbeit weniger geeignet: Die Angabe markanter Flexionsformen, wie etwa die Formen im Genitiv Singular und Nominativ Plural, ist für einen Muttersprachler (in den meisten Fällen) trivial, die Angabe einer Referenz erfordert hingegen zunächst die Auswahl des Flexionsschema. Erfolgt die Eingabe manuell, erweist sich dies aber nach einer Einarbeitungsphase als Trugschluß: Die Eingabe der entsprechenden Wortformen ist zeitaufwendiger und stärker fehleranfällig. Aus Sicht der maschinellen Verarbeitung der Informationen bietet die Angabe einer Referenz zudem einen weiteren gravierenden Vorteil: Die Referenz kann unmittelbar als Zugriffsschlüssel auf das Schema dienen.

4.2 Flexionsschemata

Nach der Entscheidung zur Verwendung von Flexionsschemata einerseits und Referenzen in den Lexikoneinträgen auf diese andererseits gilt es, die einzelnen Flexionsschemata zu bestimmen.

Die offensichtlichste Erscheinung der Flexion sind die als Suffix hinzutretenden Morpheme am Ende der Wortformen – die Endungen – und die Veränderungen des Stammvokals – Umlaut. Beide Erscheinungen dienen zur Unterscheidung von Flexionsklassen. Eine Ausnahme bildet dabei die Behandlung einiger weniger Flexionsendungen in speziellen Flexionsklassen in Abhängigkeit vom Stammauslaut. Diese wurden innerhalb der Flexionsklassen durch konditionale Ausdrücke beschrieben. Teilweise wurden Flexionsklassen unterschieden nach Optionen für die Endungen im Genitiv und Dativ.

Auf eine Unterscheidung der Flexionsklassen bezüglich des Genus wurde verzichtet. Dadurch können bestimmte Flexionsklassen für maskuline und neutrale Substantive gleichermaßen genutzt und damit die Anzahl der Flexionsklassen insgesamt eingeschränkt werden.

Für die Behandlung der Flexion substantivierter Adjektive sind ebenso erweiterte Flexionsklassen erforderlich wie für die Behandlung von Fremdwörtern. Die entsprechenden Flexionsklassen werden hier nicht weiter diskutiert.

Eine solche pragmatische Definition von Flexionsschemata genügt kaum den Ansprüchen an eine tief linguistisch motivierte Darstellung der Informationen. Technisch erfüllt sie die dagegen vollständig die Aufgabe der Darstellung der entsprechenden orthographischen Formen und genügt damit den gegenwärtigen Anforderungen.

5 Implementation

Für die Implementation der Flexionsschemata wurde der Lexikonformalismus *LeV4* in der Version V 1.27[4] ([4]) benutzt.

Eine Darstellung in *LeV4* besteht grundsätzlich aus mindestens zwei Teilen: zum einen der Definition von `class` Klassen und von `base` Grundinstanzen. Die Berechnung der Einträge gemäß der Anwendung geschieht mit Hilfe der eigentlichen Inferenzmaschine des Formalismus, indem die Instanzen als `realize` Realisierungen von Grundinstanzen in den applikationsspezifischen Klassen berechnet werden.

Um die Arbeit mit *LeV4* zu erläutern, sollen zwei Implementationsvarianten angegeben werden.

In beiden Implementationsvarianten (die vollständigen Implementationsvarianten mit Beispielen und eine Routine zu deren Ausgabe befindet sich im Anhang) werden die Flexionsschemata mit Hilfe der Macros `noun_flex_/2`, z.B.

```
# noun_flex_(4, Stamm) :: #n_c_flex(Stamm, ~gen, sg).
# noun_flex_(4, Stamm) :: #n_ce_flex(Stamm, s, gen, sg).
# noun_flex_(4, Stamm) :: #n_ce_flex(Stamm, es, gen, sg).
# noun_flex_(4, Stamm) :: #n_ce_flex(Stamm, e, dat, sg).

# noun_flex_(4, Stamm) :: #n_ce_flex(Stamm, er, ~dat, pl).
# noun_flex_(4, Stamm) :: #n_ce_flex(Stamm, ern, dat, pl).
```

dargestellt. Diese Macros greifen auf die Macros `n_`*xxx*`_flex/{3;4;5}`, beispielsweise `n_ce_flex/4` zu.

```
#n_ce_flex(Stamm,Endung,Kasus,Numerus) ::
        syn_ibm: phon: #atomcat(Stamm,Endung) &
        case: Kasus &
        num: Numerus.
```

zu, die die Werte der einzelnen Attribute setzen. Der Unterschied der beiden Implementierungen liegt im Aufruf der Macros.

5.1 Variante 1

Die Klasse `subst_eintrag` definiert die Klasse der Lexikoneinträge:

[4]Dies war die aktuelle Version zum Zeitpunkt der hier dokumentierten Implementation.

```
class subst_eintrag :< top >:
  orth: top &
  flex: (1 \ '1_1' \ '1_2' \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \
         11 \ 12 \ 13 \ 20 \ 21 \ 22 \ 23 \ 24 \ 25 \ 26 \ 27 \ 40).
```

Die Instanzen dieser Klasse werden mit Hilfe des Macros

```
#subst_macro(String, FleKlasse) ::
  orth: String &
  flex: FleKlasse.
```

belegt.

Eine `base`–Definition wie

```
base 'Feld' :<< subst_eintrag >>: #subst_macro('Feld', 4).
```

instantiiert das Attribut `orth` daher mit `'Feld'` und `flex` mit `4`. Es ist wichtig festzustellen, daß damit genau **ein** Lexikoneintrag je Grundform angelegt wird.

Die flektierten Formen werden erst in der von `subst_eintrag` abgeleiteten Klasse `test_flex`

```
class test_flex :< subst_eintrag >:
  orth: String - orth &
  flex: FleKlasse - flex &
  #subst_flex(String, FleKlasse).
```

gebildet, indem über das Macro `subst_flex`

```
#subst_flex(String, FleKlasse) ::
  (#noun_flex_(FleKlasse, String) & num:sg & gender:gender_v) \
  (#noun_flex_(FleKlasse, String) & num:pl).
```

auf die jeweiligen Flexionsschemata zugegriffen wird.

5.2 Variante 2

Die Klasse der Lexikoneinträge wird hier durch die Klasse `test_flex` definiert:

```
class test_flex :< top >: syn_ibm: phon: top &
                          case: top &
                          num: number_v &
                          gender:gender_v.
```

Ein Lexikoneintrag wie

```
base 'Feld' :<< test_flex >>: #subst_flex('Feld', 4).
```

belegt nun über das Macro `subst_flex/2`

```
#subst_flex(String, FleKlasse) ::
  (#noun_flex_(FleKlasse, String) & num:sg) \
  (#noun_flex_(FleKlasse, String) & num:pl).
```

die Lexikoneinträge. `subst_flex/2` greift dabei jeweils auf die Flexionsschemata durch das Macro `noun_flex_/2` zu. Infolgedessen ergeben sich mehrere Belegungsvarianten für die Klasse `test_flex` und damit **mehrere** Lexikoneinträge je Grundform.

5.3 Vergleich

Obwohl dieselben Flexionsschemata in beiden Versionen benutzt werden und beide Versionen das gleiche Ergebnis liefern, unterscheiden sie sich in einem entscheidenden Punkt: der Anzahl der Einträge je Grundform. Einer Implementation entsprechend Variante 1 sollte daher immer der Vorrang gegeben werden, da damit das Lexikon mit den Grundformen[5] – aus welchem letztlich alle anderen Lexika abgeleitet werden – wesentlich kompakter gehalten werden kann.

6 Zusammenfassung

Es wurde eine Implementation einer einfachen Flexionsbehandlung für Substantive des Deutschen gezeigt und diskutiert. Auf die engen Grenzen des linguistischen Ansatzes wurde hingewiesen. Die Nutzung des Lexikonformalismus *LeX4* in seinen Grundfunktionen wurde demonstriert. Die Implementation ist in angepaßter Form

[5] Für eine der Folgeversionen des Lexikonformalismus *LeX4* ist eine Ablage dieser Grundformen in einem Datenbanksystem vorgesehen.

Bestandteil eines größeren Lexikons, aus dem das Lexikon für den VERBMOBIL–Demonstrator generiert wurde, und erfüllt dort die gestellten Anforderungen zur Beschreibung der flektierten Formen.

Literatur

[1] *Der Große Duden.* VEB Bibliographisches Institut Leipzig. 1984.

[2] Drosdowski, G. (Hrsg.): *Duden — Die Grammatik.* Dudenverlag. Mannheim. 1984.

[3] *Duden — Rechtschreibung der deutschen Sprache.* Dudenverlag. Mannheim. 1991.

[4] Gebhardi, G. und Heinecke, H.: *Lexikonformlismus LeV4 — Sprachbeschreibung.* Verbmobil Technisches Dokument Nr. 19. Humboldt–Universität zu Berlin. 1995.

[5] Helbig, G. und Buscha, J.: *Deutsche Grammatik.* VEB Verlag Enzyklopädie Leipzig. 1972.

[6] Paul, H.: *Deutsche Grammatik.* 1896 (?). Reprint: VEB Max Niemeyer Verlag. Halle (Saale). 1958

[7] Schulz, D. und Griesbach, H.: Grammatik der deutschen Sprache. Max Huebner Verlag. Ismaning. 1978.

[8] Wahlster, W. und Engelkamp, J.: *Wissenschaftliche Ziele und Netzpläne für das VERBMOBIL-Projekt.* Deutsches Forschungszentrum für Künstliche Intelligenz. Saarbrücken. 1992.

[9] Wahrig, G. (Hrsg.): *Deutsches Wörterbuch.* Bertelsmann Lexikon-Verlag. Gütersloh. 1978.

[10] Wahrig-Burfeind, R. (Hrsg.): *Deutsches Wörterbuch.* Bertelsmann Lexikon–Verlag. Gütersloh. 1994.

[11] *Wörterbuch Deutsch – Englisch.* VEB Verlag Enzyklopädie Leipzig. 1980.

A Deklination (Variante 1)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Verbmobil Lexikon - Version 0.32
% Deklination, Variant1
% HUB - LeX4
% Autoren: G.Gebhardi, J.Heinecke
%          Computerlinguistik
%          Humboldt Universit"at zu Berlin
%          {gebhardi|heinecke}@compling.hu-berlin.de
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Macros zur Verknuepfung des Stamms, der Endung, des Kasus und des
%%% Numerus unter Beachtung etwaiger Umlautung
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Endung % Umlaut % ein Kasus % Praedikat
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   -    %   -    %      -    % n_flex/3
%   -    %   -    %      +    % n_c_flex/4
%   +    %   -    %      +    % n_ce_flex/5
%   +    %   -    %      -    % n_e_flex/4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   -    %   +    %      -    % n_u_flex/3
%   -    %   +    %      +    % n_uc_flex/4
%   +    %   +    %      +    % n_uce_flex/5
%   +    %   +    %      -    % n_ue_flex/4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% n_ce_flex/5
% neue Endung dran
#n_ce_flex(Stamm,Endung,Kasus,Numerus) ::
        syn_ibm: phon: #atomcat(Stamm,Endung) &
        case: Kasus &
        num: Numerus.

% n_c_flex/4
% nix dran
#n_c_flex(Stamm,Kasus,Numerus) ::
        syn_ibm: phon: Stamm &
        case: Kasus &
        num: Numerus.
```

```
% n_e_flex/4
% neue Endung dran und alles ein Kasus
#n_e_flex(Stamm,Endung,Numerus) ::
        syn_ibm: phon: #atomcat(Stamm,Endung) &
        case: case_v &
        num: Numerus.

% n_flex/3
% nix dran und alles ein Kasus
#n_flex(Stamm,Numerus) ::
        syn_ibm: phon: Stamm &
        case: case_v &
        num: Numerus.

% --- Umlaut ---

% n_uce_flex/5
% umlauten und neue Endung dran
#n_uce_flex(Stamm,Endung,Kasus,Numerus) ::
        top &&
        tmp: (#atomumlaut(Stamm) & NeuerStamm) &
        syn_ibm: phon: #atomcat(NeuerStamm,Endung) &
        case: Kasus &
        num: Numerus.

% n_uc_flex/4
% umlauten und nix dran
#n_uc_flex(Stamm,Kasus,Numerus) ::
        syn_ibm: phon: #atomumlaut(Stamm) &
        case: Kasus &
        num: Numerus.

% n_ue_flex/4
% umlauten und alles ein Kasus
#n_ue_flex(Stamm,Endung,Numerus) ::
        top &&
        tmp: (#atomumlaut(Stamm) & NeuerStamm) &
        syn_ibm: phon: #atomcat(NeuerStamm,Endung) &
        case: case_v &
        num: Numerus.

% n_u_flex/3
% umlauten und nix dran und alles ein Kasus
#n_u_flex(Stamm,Numerus) ::
        syn_ibm: phon: #atomumlaut(Stamm) &
        case: case_v &
        num: Numerus.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Zusatzmakros
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% subst_macro
#subst_macro(String, FleKlasse) ::
  orth: String &
  flex: FleKlasse.

% subst_flex/2
#subst_flex(String, FleKlasse) ::
  (#noun_flex_(FleKlasse, String) & num:sg & gender:gender_v) \
  (#noun_flex_(FleKlasse, String) & num:pl).




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Klassendefinition zu Anschauungszwecken
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

class subst_eintrag :< top >:
  orth: top &
  flex: (1\'1_1'\'1_2'\2\3\4\5\6\7\8\9\10\11\12\13\20\21\22\23\24\25\26\27\40).

class test_flex :< subst_eintrag >:
  orth: String - orth &
  flex: FleKlasse - flex &
  #subst_flex(String, FleKlasse).

class gender_v :< top >: (masc \ fem \ neut).
class number_v :< top >: (sg \ pl).
class case_v :< top >: (nom \ gen \ dat \ acc).

class test_c :< test_flex >: case: case_v & num: number_v.



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Klassendefinitionen
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%% die base-Definitionen dienen als Beispiel

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% es wird sowohl der Genitiv mit -s/-es als auch der Dativ
% mit -/-e gebildet

base 'Tag' :<< subst_eintrag >>: #subst_macro('Tag', 1).
base 'Brot' :<< subst_eintrag >>: #subst_macro('Brot', 1).

# noun_flex_(1, Stamm) :: #n_c_flex(Stamm, ~ gen, sg).      % - Endung + Kasus
# noun_flex_(1, Stamm) :: #n_ce_flex(Stamm, s, gen, sg).    % + Endung + Kasus
# noun_flex_(1, Stamm) :: #n_ce_flex(Stamm, es, gen, sg).   % + Endung + Kasus
# noun_flex_(1, Stamm) :: #n_ce_flex(Stamm, e, dat, sg).    % + Endung + Kasus

# noun_flex_(1, Stamm) :: #n_ce_flex(Stamm, e, ~dat, pl).   % + Endung + Kasus
# noun_flex_(1, Stamm) :: #n_ce_flex(Stamm, en, dat, pl).   % + Endung + Kasus



% es wird nur der Genitiv mit -es und der Dativ -e gebildet

base 'Index' :<< subst_eintrag >>: #subst_macro('Index', '1_1').
% der Dativ ist hier eher ungebraeuchlich

# noun_flex_('1_1', Stamm) :: #n_c_flex(Stamm, nom \ acc, sg). % - Endung + Kasus
# noun_flex_('1_1', Stamm) :: #n_ce_flex(Stamm, es, gen, sg).  % + Endung + Kasus
# noun_flex_('1_1', Stamm) :: #n_ce_flex(Stamm, e, dat, sg).   % + Endung + Kasus

# noun_flex_('1_1', Stamm) :: #n_ce_flex(Stamm, e, ~dat, pl).  % + Endung + Kasus
# noun_flex_('1_1', Stamm) :: #n_ce_flex(Stamm, en, dat, pl).  % + Endung + Kasus



% es wird nur der Genitiv mit -s und der Dativ - gebildet

base 'Sultan' :<< subst_eintrag >>: #subst_macro('Sultan' , '1_2').

# noun_flex_('1_2', Stamm) :: #n_c_flex(Stamm, ~gen, sg).     % - Endung + Kasus
# noun_flex_('1_2', Stamm) :: #n_ce_flex(Stamm, s, gen, sg).  % + Endung + Kasus

# noun_flex_('1_2', Stamm) :: #n_ce_flex(Stamm, e, ~dat, pl). % + Endung + Kasus
# noun_flex_('1_2', Stamm) :: #n_ce_flex(Stamm, en, dat, pl). % + Endung + Kasus
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Bus' :<< subst_eintrag >>: #subst_macro('Bus', 2).

# noun_flex_(2, Stamm) :: #n_c_flex(Stamm, ~gen, sg).      % - Endung + Kasus
# noun_flex_(2, Stamm) :: #n_ce_flex(Stamm, ses, gen, sg). % + Endung + Kasus
# noun_flex_(2, Stamm) :: #n_ce_flex(Stamm, se, dat, sg).  % + Endung + Kasus

# noun_flex_(2, Stamm) :: #n_ce_flex(Stamm, se, ~dat, pl).
# noun_flex_(2, Stamm) :: #n_ce_flex(Stamm, sen, dat, pl).




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 3
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Bach' :<< subst_eintrag >>: #subst_macro('Bach', 3).
base 'Flo"s' :<< subst_eintrag >>: #subst_macro('Flo"s', 3).


# noun_flex_(3, Stamm) :: #n_c_flex(Stamm, ~gen, sg).         % - Endung + Kasus
# noun_flex_(3, Stamm) :: top &&
                          tmp: #atomdiffz(Stamm,'"s') &       % kein "s
                          #n_ce_flex(Stamm, s, gen, sg).      % + Endung + Kasus
# noun_flex_(3, Stamm) :: #n_ce_flex(Stamm, es, gen, sg).     % + Endung + Kasus
# noun_flex_(3, Stamm) :: #n_ce_flex(Stamm, e, dat, sg).      % + Endung + Kasus

# noun_flex_(3, Stamm) :: #n_uce_flex(Stamm, e, ~dat, pl).
# noun_flex_(3, Stamm) :: #n_uce_flex(Stamm, en, dat, pl).




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Leib' :<< subst_eintrag >>: #subst_macro('Leib', 4).
```

```
base 'Feld' :<< subst_eintrag >>: #subst_macro('Feld', 4).

# noun_flex_(4, Stamm) :: #n_c_flex(Stamm, ~gen, sg).
# noun_flex_(4, Stamm) :: #n_ce_flex(Stamm, s, gen, sg).
# noun_flex_(4, Stamm) :: #n_ce_flex(Stamm, es, gen, sg).
# noun_flex_(4, Stamm) :: #n_ce_flex(Stamm, e, dat, sg).

# noun_flex_(4, Stamm) :: #n_ce_flex(Stamm, er, ~dat, pl).
# noun_flex_(4, Stamm) :: #n_ce_flex(Stamm, ern, dat, pl).




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 5
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Wald' :<< subst_eintrag >>: #subst_macro('Wald', 5).
base 'Dorf' :<< subst_eintrag >>: #subst_macro('Dorf', 5).

# noun_flex_(5, Stamm) :: #n_c_flex(Stamm, ~gen, sg).
# noun_flex_(5, Stamm) :: #n_ce_flex(Stamm, es, gen, sg).
# noun_flex_(5, Stamm) :: #n_ce_flex(Stamm, s, gen, sg).
# noun_flex_(5, Stamm) :: #n_ce_flex(Stamm, e, dat, sg).

# noun_flex_(5, Stamm) :: #n_uce_flex(Stamm, er, ~dat, pl).
# noun_flex_(5, Stamm) :: #n_uce_flex(Stamm, ern, dat, pl).




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 6
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Deckel' :<< subst_eintrag >>: #subst_macro('Deckel', 6).
base 'Lappen' :<< subst_eintrag >>: #subst_macro('Lappen', 6).
base 'Fenster' :<< subst_eintrag >>: #subst_macro('Fenster', 6).
base 'M"adchen' :<< subst_eintrag >>: #subst_macro('M"adchen', 6).

# noun_flex_(6, Stamm) :: #n_c_flex(Stamm, ~gen, sg).
# noun_flex_(6, Stamm) :: #n_ce_flex(Stamm, s, gen, sg).

# noun_flex_(6, Stamm) :: #n_c_flex(Stamm, ~dat, pl).
# noun_flex_(6, Stamm) :: top &&
                          tmp: #atomdiffz(Stamm,n) &
```

```
                          #n_ce_flex(Stamm, n, dat, pl).
# noun_flex_(6, Stamm) :: top &&
                          tmp: #atomcmpz(Stamm,n) &
                          #n_c_flex(Stamm, dat, pl).




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 7
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Vater' :<< subst_eintrag >>: #subst_macro('Vater', 7).
base 'Faden' :<< subst_eintrag >>: #subst_macro('Faden', 7).
base 'Kloster' :<< subst_eintrag >>: #subst_macro('Kloster', 7).

# noun_flex_(7, Stamm) :: #n_c_flex(Stamm, ~gen, sg).
# noun_flex_(7, Stamm) :: #n_ce_flex(Stamm, s, gen, sg).

# noun_flex_(7, Stamm) :: #n_uc_flex(Stamm, ~dat, pl).
# noun_flex_(7, Stamm) :: top &&
                          tmp: #atomdiffz(Stamm, n) &
                          #n_uce_flex(Stamm, n, dat, pl).
# noun_flex_(7, Stamm) :: top &&
                          tmp: #atomcmpz(Stamm, n) &
                          #n_uc_flex(Stamm, dat, pl).




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 8
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Gummi' :<< subst_eintrag >>: #subst_macro('Gummi', 8).
base 'Auto' :<< subst_eintrag >>: #subst_macro('Auto', 8).

# noun_flex_(8, Stamm) :: #n_c_flex(Stamm, ~gen, sg).
# noun_flex_(8, Stamm) :: #n_ce_flex(Stamm, s, gen, sg).

# noun_flex_(8, Stamm) :: #n_e_flex(Stamm, s, pl).
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 9
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Mensch' :<< subst_eintrag >>: #subst_macro('Mensch', 9).

# noun_flex_(9, Stamm) :: #n_c_flex(Stamm, nom, sg).
# noun_flex_(9, Stamm) :: #n_ce_flex(Stamm, en, ~nom, sg).
# noun_flex_(9, Stamm) :: #n_ce_flex(Stamm, ens, gen, sg).

# noun_flex_(9, Stamm) :: #n_e_flex(Stamm, en, pl).




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 10
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Gedanke' :<< subst_eintrag >>: #subst_macro('Gedanke', 10).

# noun_flex_(10, Stamm) :: #n_c_flex(Stamm, nom, sg).
# noun_flex_(10, Stamm) :: #n_ce_flex(Stamm, n, ~nom, sg).
# noun_flex_(10, Stamm) :: #n_ce_flex(Stamm, ns, gen, sg).

# noun_flex_(10, Stamm) :: #n_e_flex(Stamm, n, pl).




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 11
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Staat' :<< subst_eintrag >>: #subst_macro('Staat', 11).
base 'Bett' :<< subst_eintrag >>: #subst_macro('Bett', 11).

# noun_flex_(11, Stamm) :: #n_c_flex(Stamm, ~ gen, sg).     % - Endung + Kasus
# noun_flex_(11, Stamm) :: #n_ce_flex(Stamm, s, gen, sg).   % + Endung + Kasus
# noun_flex_(11, Stamm) :: #n_ce_flex(Stamm, es, gen, sg).  % + Endung + Kasus
# noun_flex_(11, Stamm) :: #n_ce_flex(Stamm, e, dat, sg).   % + Endung + Kasus

# noun_flex_(11, Stamm) :: #n_e_flex(Stamm, en, pl).        % - Endung + Kasus
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 12
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Direktor' :<< subst_eintrag >>: #subst_macro('Direktor', 12).

# noun_flex_(12, Stamm) :: #n_c_flex(Stamm, ~gen, sg).      % - Endung + Kasus
# noun_flex_(12, Stamm) :: #n_ce_flex(Stamm, s, gen, sg).   % + Endung + Kasus

# noun_flex_(12, Stamm) :: #n_e_flex(Stamm, en, pl).        % - Endung + Kasus




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 13
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'See' :<< subst_eintrag >>: #subst_macro('See', 13).
base 'Auge' :<< subst_eintrag >>: #subst_macro('Auge', 13).

# noun_flex_(13, Stamm) :: #n_c_flex(Stamm, ~gen, sg).      % - Endung + Kasus
# noun_flex_(13, Stamm) :: #n_ce_flex(Stamm, s, gen, sg).   % + Endung + Kasus

# noun_flex_(13, Stamm) :: #n_e_flex(Stamm, n, pl).         % - Endung + Kasus




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 20
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Drangsal' :<< subst_eintrag >>: #subst_macro('Drangsal', 20).

# noun_flex_(20, Stamm) :: #n_flex(Stamm, sg).               % - Endung - Kasus

# noun_flex_(20, Stamm) :: #n_ce_flex(Stamm, e, ~dat, pl).   % + Endung + Kasus
# noun_flex_(20, Stamm) :: #n_ce_flex(Stamm, en, dat, pl).   % - Endung + Kasus
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 21
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Nacht' :<< subst_eintrag >>: #subst_macro('Nacht', 21).

# noun_flex_(21, Stamm) :: #n_flex(Stamm, sg).                % - Endung - Kasus

# noun_flex_(21, Stamm) :: #n_uce_flex(Stamm, e, ~dat, pl).   % + Endung + Kasus
# noun_flex_(21, Stamm) :: #n_uce_flex(Stamm, en, dat, pl).   % + Endung + Kasus




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 22
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Kenntnis' :<< subst_eintrag >>: #subst_macro('Kenntnis', 22).

# noun_flex_(22, Stamm) :: #n_flex(Stamm, sg).               % - Endung - Kasus

# noun_flex_(22, Stamm) :: #n_ce_flex(Stamm, se, ~dat, pl).  % + Endung + Kasus
# noun_flex_(22, Stamm) :: #n_ce_flex(Stamm, sen, dat, pl).  % + Endung + Kasus




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 23
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Mutter' :<< subst_eintrag >>: #subst_macro('Mutter', 23).

# noun_flex_(23, Stamm) :: #n_flex(Stamm, sg).               % - Endung - Kasus

# noun_flex_(23, Stamm) :: #n_uc_flex(Stamm, ~dat, pl).      % - Endung - Kasus
# noun_flex_(23, Stamm) :: #n_uce_flex(Stamm, n, dat, pl).   % + Endung + Kasus




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 24
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
base 'Kamera' :<< subst_eintrag >>: #subst_macro('Kamera', 24).

# noun_flex_(24, Stamm) :: #n_flex(Stamm, sg).          % - Endung - Kasus

# noun_flex_(24, Stamm) :: #n_e_flex(Stamm, s, pl).     % + Endung - Kasus




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 25
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Zahl' :<< subst_eintrag >>: #subst_macro('Zahl', 25).

# noun_flex_(25, Stamm) :: #n_flex(Stamm, sg).          % - Endung - Kasus

# noun_flex_(25, Stamm) :: #n_e_flex(Stamm, en, pl).    % + Endung - Kasus




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 26
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Blume' :<< subst_eintrag >>: #subst_macro('Blume', 26).

# noun_flex_(26, Stamm) :: #n_flex(Stamm, sg).          % - Endung - Kasus

# noun_flex_(26, Stamm) :: #n_e_flex(Stamm, n, pl).     % + Endung - Kasus




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 27
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Lehrerin' :<< subst_eintrag >>: #subst_macro('Lehrerin', 27).

# noun_flex_(27, Stamm) :: #n_flex(Stamm, sg).          % - Endung - Kasus

# noun_flex_(27, Stamm) :: #n_e_flex(Stamm, nen, pl).   % + Endung - Kasus
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 40
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Herz' :<< subst_eintrag >>: #subst_macro('Herz', 40).

# noun_flex_(40, Stamm) :: #n_c_flex(Stamm, nom \ acc, sg).
# noun_flex_(40, Stamm) :: #n_ce_flex(Stamm, ens, gen, sg).
# noun_flex_(40, Stamm) :: #n_ce_flex(Stamm, en, gen \ dat, sg).

# noun_flex_(40, Stamm) :: #n_e_flex(Stamm, en, pl).
```

B Deklination (Variante 2)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Verbmobil Lexikon - Version 0.32
% Deklination, Variante2
% HUB - LeX4
% Autoren: G.Gebhardi, J.Heinecke
%          Computerlinguistik
%          Humboldt Universit"at zu Berlin
%          {gebhardi|heinecke}@compling.hu-berlin.de
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Macros zur Verknuepfung des Stamms, der Endung, des Kasus und des
%%% Numerus unter Beachtung etwaiger Umlautung
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Endung % Umlaut % ein Kasus % Praedikat
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   -    %   -    %     -     % n_flex/3
%   -    %   -    %     +     % n_c_flex/4
%   +    %   -    %     +     % n_ce_flex/5
%   +    %   -    %     -     % n_e_flex/4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   -    %   +    %     -     % n_u_flex/3
%   -    %   +    %     +     % n_uc_flex/4
%   +    %   +    %     +     % n_uce_flex/5
%   +    %   +    %     -     % n_ue_flex/4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% n_ce_flex/5
% neue Endung dran
#n_ce_flex(Stamm,Endung,Kasus,Numerus) ::
        syn_ibm: phon: #atomcat(Stamm,Endung) &
        case: Kasus &
        num: Numerus.

% n_c_flex/4
% nix dran
#n_c_flex(Stamm,Kasus,Numerus) ::
        syn_ibm: phon: Stamm &
        case: Kasus &
        num: Numerus.
```

```
% n_e_flex/4
% neue Endung dran und alles ein Kasus
#n_e_flex(Stamm,Endung,Numerus) ::
        syn_ibm: phon: #atomcat(Stamm,Endung) &
        case: case_v &
        num: Numerus.

% n_flex/3
% nix dran und alles ein Kasus
#n_flex(Stamm,Numerus) ::
        syn_ibm: phon: Stamm &
        case: case_v &
        num: Numerus.

% --- Umlaut ---

% n_uce_flex/5
% umlauten und neue Endung dran
#n_uce_flex(Stamm,Endung,Kasus,Numerus) ::
        top &&
        tmp: (#atomumlaut(Stamm) & NeuerStamm) &
        syn_ibm: phon: #atomcat(NeuerStamm,Endung) &
        case: Kasus &
        num: Numerus.

% n_uc_flex/4
% umlauten und nix dran
#n_uc_flex(Stamm,Kasus,Numerus) ::
        syn_ibm: phon: #atomumlaut(Stamm) &
        case: Kasus &
        num: Numerus.

% n_ue_flex/4
% umlauten und alles ein Kasus
#n_ue_flex(Stamm,Endung,Numerus) ::
        top &&
        tmp: (#atomumlaut(Stamm) & NeuerStamm) &
        syn_ibm: phon: #atomcat(NeuerStamm,Endung) &
        case: case_v &
        num: Numerus.

% n_u_flex/3
% umlauten und nix dran und alles ein Kasus
#n_u_flex(Stamm,Numerus) ::
        syn_ibm: phon: #atomumlaut(Stamm) &
        case: case_v &
        num: Numerus.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Zusatzmakros
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% subst_flex/2
#subst_flex(String, FleKlasse) ::
  (#noun_flex_(FleKlasse, String) & num:sg) \
  (#noun_flex_(FleKlasse, String) & num:pl).





%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Klassendefinition zu Anschauungszwecken
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

class test_flex :< top >: syn_ibm: phon: top &    % flektierte Form
                          case: top &             % fuer LeX4 V 1.xx
                          num: number_v &
                          gender:gender_v.

class gender_v :< top >: (masc \ fem \ neut).
class number_v :< top >: (sg \ pl).
class case_v :< top >: (nom \ gen \ dat \ acc).

class test_c :< test_flex >: case: case_v &
                             num: sg.
class test_c :< test_flex >: case: case_v &
                             num: pl -
                             gender.



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Klassendefinitionen
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% die base-Definitionen dienen als Beispiel
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% es wird sowohl der Genitiv mit -s/-es als auch der Dativ
% mit -/-e gebildet

base 'Tag' :<< test_flex >>: #subst_flex('Tag', 1).
base 'Brot' :<< test_flex >>: #subst_flex('Brot', 1).

# noun_flex_(1, Stamm) :: #n_c_flex(Stamm, ~ gen, sg).      % - Endung + Kasus
# noun_flex_(1, Stamm) :: #n_ce_flex(Stamm, s, gen, sg).    % + Endung + Kasus
# noun_flex_(1, Stamm) :: #n_ce_flex(Stamm, es, gen, sg).   % + Endung + Kasus
# noun_flex_(1, Stamm) :: #n_ce_flex(Stamm, e, dat, sg).    % + Endung + Kasus

# noun_flex_(1, Stamm) :: #n_ce_flex(Stamm, e, ~dat, pl).   % + Endung + Kasus
# noun_flex_(1, Stamm) :: #n_ce_flex(Stamm, en, dat, pl).   % + Endung + Kasus



% es wird nur der Genitiv mit -es und der Dativ -e gebildet

base 'Index' :<< test_flex >>: #subst_flex('Index', '1_1').
% der Dativ ist hier eher ungebraeuchlich

# noun_flex_('1_1', Stamm) :: #n_c_flex(Stamm, nom \ acc, sg). % - Endung + Kasus
# noun_flex_('1_1', Stamm) :: #n_ce_flex(Stamm, es, gen, sg).  % + Endung + Kasus
# noun_flex_('1_1', Stamm) :: #n_ce_flex(Stamm, e, dat, sg).   % + Endung + Kasus

# noun_flex_('1_1', Stamm) :: #n_ce_flex(Stamm, e, ~dat, pl).  % + Endung + Kasus
# noun_flex_('1_1', Stamm) :: #n_ce_flex(Stamm, en, dat, pl).  % + Endung + Kasus



% es wird nur der Genitiv mit -s und der Dativ - gebildet

base 'Sultan' :<< test_flex >>: #subst_flex('Sultan' , '1_2').

# noun_flex_('1_2', Stamm) :: #n_c_flex(Stamm, ~gen, sg).     % - Endung + Kasus
# noun_flex_('1_2', Stamm) :: #n_ce_flex(Stamm, s, gen, sg).  % + Endung + Kasus

# noun_flex_('1_2', Stamm) :: #n_ce_flex(Stamm, e, ~dat, pl). % + Endung + Kasus
# noun_flex_('1_2', Stamm) :: #n_ce_flex(Stamm, en, dat, pl). % + Endung + Kasus
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Bus' :<< test_flex >>: #subst_flex('Bus', 2).

# noun_flex_(2, Stamm) :: #n_c_flex(Stamm, ~gen, sg).      % - Endung + Kasus
# noun_flex_(2, Stamm) :: #n_ce_flex(Stamm, ses, gen, sg). % + Endung + Kasus
# noun_flex_(2, Stamm) :: #n_ce_flex(Stamm, se, dat, sg).  % + Endung + Kasus

# noun_flex_(2, Stamm) :: #n_ce_flex(Stamm, se, ~dat, pl).
# noun_flex_(2, Stamm) :: #n_ce_flex(Stamm, sen, dat, pl).




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 3
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Bach' :<< test_flex >>: #subst_flex('Bach', 3).
base 'Flo"s' :<< test_flex >>: #subst_flex('Flo"s', 3).

# noun_flex_(3, Stamm) :: #n_c_flex(Stamm, ~gen, sg).         % - Endung + Kasus
# noun_flex_(3, Stamm) :: top &&
                          tmp: #atomdiffz(Stamm,'"s') &      % kein "s
                          #n_ce_flex(Stamm, s, gen, sg).     % + Endung + Kasus
# noun_flex_(3, Stamm) :: #n_ce_flex(Stamm, es, gen, sg).    % + Endung + Kasus
# noun_flex_(3, Stamm) :: #n_ce_flex(Stamm, e, dat, sg).     % + Endung + Kasus

# noun_flex_(3, Stamm) :: #n_uce_flex(Stamm, e, ~dat, pl).
# noun_flex_(3, Stamm) :: #n_uce_flex(Stamm, en, dat, pl).




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Leib' :<< test_flex >>: #subst_flex('Leib', 4).
base 'Feld' :<< test_flex >>: #subst_flex('Feld', 4).

# noun_flex_(4, Stamm) :: #n_c_flex(Stamm, ~gen, sg).
# noun_flex_(4, Stamm) :: #n_ce_flex(Stamm, s, gen, sg).
```

```
# noun_flex_(4, Stamm) :: #n_ce_flex(Stamm, es, gen, sg).
# noun_flex_(4, Stamm) :: #n_ce_flex(Stamm, e, dat, sg).

# noun_flex_(4, Stamm) :: #n_ce_flex(Stamm, er, ~dat, pl).
# noun_flex_(4, Stamm) :: #n_ce_flex(Stamm, ern, dat, pl).




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 5
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Wald' :<< test_flex >>: #subst_flex('Wald', 5).
base 'Dorf' :<< test_flex >>: #subst_flex('Dorf', 5).

# noun_flex_(5, Stamm) :: #n_c_flex(Stamm, ~gen, sg).
# noun_flex_(5, Stamm) :: #n_ce_flex(Stamm, es, gen, sg).
# noun_flex_(5, Stamm) :: #n_ce_flex(Stamm, s, gen, sg).
# noun_flex_(5, Stamm) :: #n_ce_flex(Stamm, e, dat, sg).

# noun_flex_(5, Stamm) :: #n_uce_flex(Stamm, er, ~dat, pl).
# noun_flex_(5, Stamm) :: #n_uce_flex(Stamm, ern, dat, pl).




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 6
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Deckel' :<< test_flex >>: #subst_flex('Deckel', 6).
base 'Lappen' :<< test_flex >>: #subst_flex('Lappen', 6).
base 'Fenster' :<< test_flex >>: #subst_flex('Fenster', 6).
base 'M"adchen' :<< test_flex >>: #subst_flex('M"adchen', 6).

# noun_flex_(6, Stamm) :: #n_c_flex(Stamm, ~gen, sg).
# noun_flex_(6, Stamm) :: #n_ce_flex(Stamm, s, gen, sg).

# noun_flex_(6, Stamm) :: #n_c_flex(Stamm, ~dat, pl).
# noun_flex_(6, Stamm) :: top &&
                          tmp: #atomdiffz(Stamm,n) &
                          #n_ce_flex(Stamm, n, dat, pl).
# noun_flex_(6, Stamm) :: top &&
                          tmp: #atomcmpz(Stamm,n) &
                          #n_c_flex(Stamm, dat, pl).
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 7
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Vater' :<< test_flex >>: #subst_flex('Vater', 7).
base 'Faden' :<< test_flex >>: #subst_flex('Faden', 7).
base 'Kloster' :<< test_flex >>: #subst_flex('Kloster', 7).

# noun_flex_(7, Stamm) :: #n_c_flex(Stamm, ~gen, sg).
# noun_flex_(7, Stamm) :: #n_ce_flex(Stamm, s, gen, sg).

# noun_flex_(7, Stamm) :: #n_uc_flex(Stamm, ~dat, pl).
# noun_flex_(7, Stamm) :: top &&
                          tmp: #atomdiffz(Stamm, n) &
                          #n_uce_flex(Stamm, n, dat, pl).
# noun_flex_(7, Stamm) :: top &&
                          tmp: #atomcmpz(Stamm, n) &
                          #n_uc_flex(Stamm, dat, pl).




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 8
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Gummi' :<< test_flex >>: #subst_flex('Gummi', 8).
base 'Auto' :<< test_flex >>: #subst_flex('Auto', 8).

# noun_flex_(8, Stamm) :: #n_c_flex(Stamm, ~gen, sg).
# noun_flex_(8, Stamm) :: #n_ce_flex(Stamm, s, gen, sg).

# noun_flex_(8, Stamm) :: #n_e_flex(Stamm, s, pl).




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 9
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
base 'Mensch' :<< test_flex >>: #subst_flex('Mensch', 9).

# noun_flex_(9, Stamm) :: #n_c_flex(Stamm, nom, sg).
# noun_flex_(9, Stamm) :: #n_ce_flex(Stamm, en, ~nom, sg).
# noun_flex_(9, Stamm) :: #n_ce_flex(Stamm, ens, gen, sg).

# noun_flex_(9, Stamm) :: #n_e_flex(Stamm, en, pl).




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 10
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Gedanke' :<< test_flex >>: #subst_flex('Gedanke', 10).

# noun_flex_(10, Stamm) :: #n_c_flex(Stamm, nom, sg).
# noun_flex_(10, Stamm) :: #n_ce_flex(Stamm, n, ~nom, sg).
# noun_flex_(10, Stamm) :: #n_ce_flex(Stamm, ns, gen, sg).

# noun_flex_(10, Stamm) :: #n_e_flex(Stamm, n, pl).




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 11
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Staat' :<< test_flex >>: #subst_flex('Staat', 11).
base 'Bett' :<< test_flex >>: #subst_flex('Bett', 11).

# noun_flex_(11, Stamm) :: #n_c_flex(Stamm, ~ gen, sg).     % - Endung + Kasus
# noun_flex_(11, Stamm) :: #n_ce_flex(Stamm, s, gen, sg).   % + Endung + Kasus
# noun_flex_(11, Stamm) :: #n_ce_flex(Stamm, es, gen, sg).  % + Endung + Kasus
# noun_flex_(11, Stamm) :: #n_ce_flex(Stamm, e, dat, sg).   % + Endung + Kasus

# noun_flex_(11, Stamm) :: #n_e_flex(Stamm, en, pl).        % - Endung + Kasus




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 12
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Direktor' :<< test_flex >>: #subst_flex('Direktor', 12).

# noun_flex_(12, Stamm) :: #n_c_flex(Stamm, ~gen, sg).      % - Endung + Kasus
# noun_flex_(12, Stamm) :: #n_ce_flex(Stamm, s, gen, sg).   % + Endung + Kasus

# noun_flex_(12, Stamm) :: #n_e_flex(Stamm, en, pl).        % - Endung + Kasus




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 13
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'See' :<< test_flex >>: #subst_flex('See', 13).
base 'Auge' :<< test_flex >>: #subst_flex('Auge', 13).

# noun_flex_(13, Stamm) :: #n_c_flex(Stamm, ~gen, sg).      % - Endung + Kasus
# noun_flex_(13, Stamm) :: #n_ce_flex(Stamm, s, gen, sg).   % + Endung + Kasus

# noun_flex_(13, Stamm) :: #n_e_flex(Stamm, n, pl).         % - Endung + Kasus




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 20
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Drangsal' :<< test_flex >>: #subst_flex('Drangsal', 20).

# noun_flex_(20, Stamm) :: #n_flex(Stamm, sg).               % - Endung - Kasus

# noun_flex_(20, Stamm) :: #n_ce_flex(Stamm, e, ~dat, pl).   % + Endung + Kasus
# noun_flex_(20, Stamm) :: #n_ce_flex(Stamm, en, dat, pl).   % - Endung + Kasus




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 21
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
base 'Nacht' :<< test_flex >>: #subst_flex('Nacht', 21).

# noun_flex_(21, Stamm) :: #n_flex(Stamm, sg).                % - Endung - Kasus

# noun_flex_(21, Stamm) :: #n_uce_flex(Stamm, e, ~dat, pl).   % + Endung + Kasus
# noun_flex_(21, Stamm) :: #n_uce_flex(Stamm, en, dat, pl).   % + Endung + Kasus




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 22
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Kenntnis' :<< test_flex >>: #subst_flex('Kenntnis', 22).

# noun_flex_(22, Stamm) :: #n_flex(Stamm, sg).               % - Endung - Kasus

# noun_flex_(22, Stamm) :: #n_ce_flex(Stamm, se, ~dat, pl).  % + Endung + Kasus
# noun_flex_(22, Stamm) :: #n_ce_flex(Stamm, sen, dat, pl).  % + Endung + Kasus




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 23
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Mutter' :<< test_flex >>: #subst_flex('Mutter', 23).

# noun_flex_(23, Stamm) :: #n_flex(Stamm, sg).               % - Endung - Kasus

# noun_flex_(23, Stamm) :: #n_uc_flex(Stamm, ~dat, pl).      % - Endung - Kasus
# noun_flex_(23, Stamm) :: #n_uce_flex(Stamm, n, dat, pl).   % + Endung + Kasus




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 24
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Kamera' :<< test_flex >>: #subst_flex('Kamera', 24).

# noun_flex_(24, Stamm) :: #n_flex(Stamm, sg).           % - Endung - Kasus
```

```
# noun_flex_(24, Stamm) :: #n_e_flex(Stamm, s, pl).      % + Endung - Kasus



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 25
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Zahl' :<< test_flex >>: #subst_flex('Zahl', 25).

# noun_flex_(25, Stamm) :: #n_flex(Stamm, sg).           % - Endung - Kasus

# noun_flex_(25, Stamm) :: #n_e_flex(Stamm, en, pl).     % + Endung - Kasus



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 26
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Blume' :<< test_flex >>: #subst_flex('Blume', 26).

# noun_flex_(26, Stamm) :: #n_flex(Stamm, sg).           % - Endung - Kasus

# noun_flex_(26, Stamm) :: #n_e_flex(Stamm, n, pl).      % + Endung - Kasus



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 27
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Lehrerin' :<< test_flex >>: #subst_flex('Lehrerin', 27).

# noun_flex_(27, Stamm) :: #n_flex(Stamm, sg).           % - Endung - Kasus

# noun_flex_(27, Stamm) :: #n_e_flex(Stamm, nen, pl).    % + Endung - Kasus
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Klasse 40
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

base 'Herz' :<< test_flex >>: #subst_flex('Herz', 40).

# noun_flex_(40, Stamm) :: #n_c_flex(Stamm, nom \ acc, sg).
# noun_flex_(40, Stamm) :: #n_ce_flex(Stamm, ens, gen, sg).
# noun_flex_(40, Stamm) :: #n_ce_flex(Stamm, en, gen \ dat, sg).

# noun_flex_(40, Stamm) :: #n_e_flex(Stamm, en, pl).
```