INSTITUT FÜR INFORMATIK
Lehr- und Forschungseinheit für
Programmier- und Modellierungssprachen

Oettingenstraße 67 D–80538 München

Ludwig-
Maximilians-
Universität
München

Eine deduktive Datenbank für die Linguistik

Gerhard Hingerl

Diplomarbeit

Beginn der Arbeit:	15. Mai 1996
Abgabe der Arbeit:	14. November 1996
Aufgabensteller:	Prof. Dr. François Bry
Betreuer:	Dr. Heribert Schütz
	Prof. Dr. Dietmar Zaefferer (Institut für Deutsche Philologie)

Erklärung

Hiermit versichere ich, daß ich diese Arbeit selbstständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 13. November 1996 Gerhard Hingerl

Zusammenfassung

Anhand einer allgemeinen Definition von Klassifikationssystemen, Klassifikatoren und Anfragen an ein Klassifikationssystem wird die Struktur einer Sprachbeschreibung als Spezialfall eines solchen Klassifikationssystems konzipiert. Klassifikationssysteme sind als UND-ODER-Bäume definiert, deren Klassifikatoren bestimmte Konsistenzbedingungen erfüllen. Eine Implementierung eines Klassifikationssystems als deduktive Datenbank wird vorgestellt und es wird gezeigt, daß mit Mitteln der Logikprogrammierung Klassifikationssysteme, deren Klassifikatoren Konsistenzbedingungen unterworfen sind, elegant realisiert werden können. Insbesondere zeigt sich, daß zum Erstellen von Einträgen in die Datenbank und zur Anfragestellung an die Datenbank derselbe Algorithmus verwendet wird. Die Einhaltung der Konsistenzbedingungen bei der Bildung eines Klassifikators wird durch Memoisierung realisiert. Die memoisierten Fakten bilden zugleich die Grundlage zur Erstellung eines dem Klassifikator zugeordneten Beschreibungsschemas, welches die Beschreibung linguistischer Phänomene unterstützt.

Welcome to the Language Database!

Inhaltsverzeichnis

1 Einleitung 2

2 Sprache und Sprachbeschreibungen 3
  2.1 Linguistische Phänomene 4
  2.2 Beschreibung linguistischer Phänomene 5

3 Einteilungen und Klassifikationssysteme 12

4 Ein Klassifikationssystem für Sprachbeschreibungen 31

5 Eine deduktive Datenbank für Sprachbeschreibungen 33
  5.1 Darstellung von Klassifikationssystemen in Prolog 34
    5.1.1 Erstellung eines Klassifikators in Prolog 41
    5.1.2 Definition von Beschreibungsschemata 47
    5.1.3 Erstellen von Anfragen 50

6 Phonotaktik: Ein erster Versuch 51

7 Phonotaktik: Ein zweiter Versuch 57

8 Rückblick und Ausblick 59

1 Einleitung

In der Linguistik ist man an Beschreibungen von menschlichen Sprachen aus verschiedenen Gründen interessiert. Zwei wichtige Gründe sind Sprachvergleiche und Universalienforschung[1]. Um dem Forscher dabei die Arbeit zu erleichtern, hat man Anstrengungen die zwei Richtungen unternommen: die Sprachbeschreibungen sollen standardisiert und rein deskriptiv[2] sein. [Comrie and Smith, 1977] haben hierzu einen Fragenkatalog erarbeitet, der eine strukturierte Handhabe bietet, um Sprachbeschreibungen durchzuführen. Auf diese Weise erstellte Sprachbeschreibungen haben eine einheitliche Struktur. Ein im Umgang mit diesen Sprachbeschreibungen geübter Sprachforscher findet so in der Regel schnell, ohne das Inhaltsverzeichnis konsultieren zu müssen, die gewünschte Information.

Da Sprachbeschreibungen jedoch meist in Buchform vorliegen, sind Sprachvergleiche immer noch mit einem hohen Aufwand an manueller Tätigkeit verbunden. Dies ist insbesondere dann der Fall, wenn viele Sprachen auf ein oder mehrere *linguistische Phänomene* hin untersucht werden sollen. Die Ursache für diesen hohen Aufwand steckt in einer gewissen Orthogonalität, die zwischen dem *Erstellen* und dem *Benutzen* von Sprachbeschreibungen vorliegt: Beim Erstellen einer Sprachbeschreibung werden alle (oder doch möglichst viele) linguistischen Phänomene *einer* Sprache erforscht und in einem Medium - in der Regel einem Buch - zusammengefaßt. Beim Benutzen von Sprachbeschreibungen für Sprachvergleiche werden hingegen mehrere Sprachbeschreibungen in Bezug auf *ein linguistisches Phänomen* hin untersucht. Der gleiche Zusammenhang wird auch für eine Datenbank für Sprachbeschreibungen die Regel sein: Eingelesen werden die linguistischen Phänomene nach Sprachen geordnet. Ausgelesen werden die Sprachen nach linguistischen Phänomenen geordnet.

In dieser Arbeit geht es darum, eine *deduktive Datenbank für die Linguistik* zu entwerfen. Die Daten, welche diese Datenbank aufnehmen soll, sind *Beschreibungen natürlicher Sprachen.* Daraus ergeben sich im wesentlichen zwei Aufgaben:

1. Der Begriff *Sprachbeschreibung* ist soweit zu explizieren, daß klar ist, welche Datenstrukturen zur Darstellung von Sprachbeschreibungen nötig und angemessen sind, und welche Benutzeranforderungen an eine

---

[1] "Grammatische Universalien sind Eigenschaften (bzw. Hypothesen über solche Eigenschaften), die allen menschlichen Sprachen gemeinsam sind". ([Bußmann, 1990], Stichwort *Universalien*Seite 819).

[2] Da die Sprachforscher verschiedenen linguistischen Schulen angehören, sollen die Sprachbeschreibungen möglichst frei von theoretischen Annahmen sein, die nicht allgemein akzeptiert sind.

mit Sprachbeschreibungen gefüllte Datenbank gestellt werden.

2. Abhängig von den Ergebnissen der ersten Aufgabe ist eine deduktive Datenbank zu entwerfen und auf ihre Brauchbarkeit hin zu bewerten.

2 Sprache und Sprachbeschreibungen

Sprache ist laut [Bußmann, 1990] ein "System von freigeschaffenen, aber konventionell überlieferten Zeichen bzw. Symbolen und Kombinationsregeln" .

Ziel einer *Sprachbeschreibung* ist es, das System einer gegebenen Sprache explizit zu machen.

Sprachbeschreibungen sind deskriptiv in dem Sinne, daß es ihr Ziel ist, den *tatsächlichen* Gebrauch einer gegebenen Sprache zu beschreiben. "deskriptiv" stellt hier den Gegensatz zu "normativ" dar, denn viele in der Vergangenheit erstellte Sprachbeschreibungen beschrieben nicht den tatsächlichen Gebrauch einer Sprache, sondern "den korrekten" Gebrauch. Damit stellten sich normative Sprachbeschreibungen bewußt gegen den tatsächlichen Gebrauch einer Sprache, wenn dieser mit "dem korrekten" Gebrauch nicht übereinstimmte.

In der Linguistik wird der Terminus "deskriptiv" aber noch mit einem anderen Terminus kontrastiert, nämlich mit "theoretisch" im Sinne von "einer (bestimmten) linguistischen Theorie zugehörig". [Comrie and Smith, 1977] meinen damit"(...) descriptions of languages or parts of languages which are (...) structured in terms of one of the current theories of language (...)".[3] Eine Sprachbeschreibung kann nur dann "deskriptiv" in diesem zweiten Sinne genannt werden, wenn sie *nicht* im begrifflichen Rahmen einer der linguistischen Theorien formuliert ist. Die Verwendung des Plurals weißt schon darauf hin, daß es "die" linguistische Theorie nicht gibt, welche von der Mehrzahl der Sprachforscher anerkannt würde, sondern mehrere konkurrierende Theorien. Eine im zweiten Sinne deskriptive Sprachbeschreibung soll theorienneutral und dadurch für alle Sprachforscher, unabhängig von deren theoretischen Neigungen, verwendbar sein.

Schließlich wird der Begriff "deskriptiv" noch mit dem Begriff "nicht beobachtbar" in Gegensatz gesetzt. Eine Sprachbeschreibung in diesem dritten Sinn ist deskriptiv, wenn sie sich "auf die Klassifizierung und Systematisierung von Beobachtungen (...) beschränkt".[4]

Während es keine Schwierigkeiten bereiten dürfte, eine im ersten Sinne

---

[3] [Comrie and Smith, 1977], Seite 5.

[4] [Glück, 1993], Stichwort *Deskription*, Seite 129.

deskriptive Sprachbeschreibung von einer normativen Sprachbeschreibung zu unterscheiden, gibt es in den anderen beiden Fällen keine klare Abgrenzung. Der Gegensatz "deskriptiv" vs. "theoretisch" unterstellt jedoch, daß es eine Menge von Begriffen gibt, welche einerseits keiner speziellen linguistischen Theorie entstammen, andererseits aber dennoch ausreichen, um Sprachbeschreibungen erstellen zu können.

Dieses *deskriptive Beschreibungsvokabular* ist ein wesentlicher Bestandteil jeder zur Sprachbeschreibung verwendeten *Metasprache*.

2.1 Linguistische Phänomene

"Ziel einer deskriptiven Sprachbeschreibung ist es, die Grammatik einer Sprache aufzudecken und darzustellen, d. h. deren phonetisch-phonologische[5], morphologische[6] und syntaktische[7] Einheiten in ihren korrespondierenden Strukturen und Regeln zu registrieren sowie Angaben zu Semantik[8] und Pragmatik[9] zu machen".[10]

Diese Einheiten, Strukturen und Regeln seien zusammengefaßt *linguistische Phänomene* genannt. Eine Sprachbeschreibung besteht also in einer Beschreibung der in der Sprache auftretenden linguistischen Phänomene.

Grundlage für eine Sprachbeschreibung ist ein *Korpus* sprachlicher Äußerungen. Ein Korpus ist "eine Ansammlung linguistischer Daten in Form von geschriebenen Texten oder transkribierter Rede, welche aus Ausgangspunkt einer linguistischen Beschreibung oder als Mittel zum Überprüfen von Hypothesen über die Sprache benutzt wird."[11]

Mittels geeigneter Analyseverfahren werden die Bestandteile und Strukturen der Elemente des Korpus aufgedeckt.

Als Ergebnis erhält man (relativ zum Korpus) eine Beschreibung der in der untersuchten Sprache vorliegenden linguistischen Phänomene; "darüberhinaus werden Generalisierungen vorgenommen, indem die Gültigkeit der Aussagen über das Korpus hinaus erweitert wird (...)".[12]

---

[5] *Phonetik* ist die Wissenschaft von der Hervorbringung, der akustischen Struktur sowie der Wahrnehmung lautsprachlicher Äußerungen.
Phonologie ist eine linguistische Teildisziplin, die sich mit den bedeutungsunterscheidenden Sprachlauten beschäftigt.

[6] *Morphologie* untersucht die interne Struktur der Wörter und Wortformen natürlicher Sprachen.

[7] *Syntax* ist die Lehre vom Bau der Sätze einer bestimmten Sprache.

[8] *Semantik* befaßt sich mit den Bedeutungen von Ausdrücken natürlicher Sprachen.

[9] *Pragmatik* ist die Lehre vom sprachlichen Handeln.

[10] [Glück, 1993], Stichwort *Sprachbeschreibung*, Seite 568.

[11] [Crystal, 1991],Stichwort *corpus*, Seite 96.

[12] [Glück, 1993], Stichwort *Sprachbeschreibung*, Seite 568.

2.2 Beschreibung linguistischer Phänomene

Die Beschreibung einer Sprache setzt eine weitere Sprache voraus: eine Metasprache, in der die untersuchte Sprache - die Objektsprache - beschrieben wird. Als Metasprache wird häufig Englisch oder eine andere natürliche Sprache verwendet, die mit dem notwendigen linguistischen Fachvokabular - dem oben erwähnten *deskriptiven Beschreibungsvokabular* - angereichert ist. Aber auch stärker formal ausgerichtete Beschreibungssprachen finden Verwendung. Stärker formale Beschreibungssprachen haben gegenüber eher weniger formalen Beschreibungssprachen den Vorteil, daß die Beschreibungen präziser sind; allerdings steigt durch zunehmende Formalisierung auch die Gefahr, daß die direkte Verständlichkeit der Beschreibungen abnimmt und deshalb vom Leser ein zunehmendes Maß an Vorwissen über den Formalismus erwartet wird. Ein zusätzliches Problem ergibt sich daraus, daß durch die Formalisierung eine Vorstrukturierung der Beschreibungen vorgenommen wird, über deren Angemessenheit Linguisten durchaus verschiedener Meinung sein können.

Linguistische Phänomene werden also anhand von in der Metasprache formulierten Aussagen über die Objektsprache **L** beschrieben. Eine Beschreibung **B** einer Objektsprache im Rahmen der Metasprache **M** ist dann eine Menge von in der Metasprache formulierten Aussagen über die Objektsprache, die (vom Beschreiber) als wahr, als auf die Objektsprache zutreffend unterstellt werden. Da natürliche Sprachen hochkomplexe Gebilde sind, die zudem ständig in Veränderung begriffen sind[13], ist eine erschöpfende vollständige Beschreibung einer natürlichen Sprache nicht möglich. Beschreibungen natürlicher Sprachen sind deshalb immer mehr oder weniger partiell.

Beispiele für linguistische Phänomene einer gegebenen Objektsprache **L** beschrieben in der Metasprache Deutsch:

1. Das Lautinventar[14] der Sprache **L** besteht aus den Phonemen $P_1, \ldots, P_n$.
2. Das Vokalinventar der Sprache **L** besteht aus den acht Vokalen, welche in der folgenden Tabelle[15] mit ihren Merkmalen aufgeführt sind:

---

[13] Außer es handelt sich um tote Sprachen wie Latein.

[14] Das sind unter anderem die in der Sprache vorkommenden Vokale und Konsonanten.

[15] vgl. [Wendt, 1972], Seite 24f.

Name	Artikulationsstelle	Lippengestalt	Kieferabstand
e	vorne	flach	weit
i	vorne	flach	eng
ö	vorne	rund	weit
ü	vorne	rund	eng
a	hinten	flach	weit
I	hinten	flach	eng
o	hinten	rund	weit
u	hinten	rund	eng

3. In der Sprache **L** gibt es nur Wörter, die mit einem Vokal enden.

4. In der Sprache L folgt in jedem Ausdruck nach zwei Konsonanten immer ein Vokal.

5. In der Sprache **L** werden Verben durch Anhängen von Suffixen an den Stamm der Verben konjugiert.

6. In der Sprache **L** besteht ein einfacher Satz aus einer Nominalphrase gefolgt von einer Verbphrase.

7. Jede Wortform der Sprache **L**, die nicht aus einer anderen Sprache entlehnt ist, ist vokalharmonisch in dem Sinne, daß jeder Suffixnukleus, der nicht zum Wortstamm der Wortform gehört, die gleiche *Artikulationsstelle* wie der ihm vorhergehende Nukleus hat und jeder *enge* Suffixnukleus darüber hinaus auch die gleiche *Lippengestalt* wie der ihm vorhergehende Nukleus.

8. In der Sprache **L** unterliegt jede grammatische Endung, welche an einen Wortstamm angefügt wird, entweder der *kleinen* oder der *großen* Vokalharmonie. Unterliegt sie der kleinen Vokalharmonie, so ist der Vokal der Endung entweder e oder a. Es wird derjenige Vokal angenommen, welcher dieselbe Artikulationsstelle wie der Vokal der letzten Silbe des Wortstamms hat. Unterliegt sie der großen Vokalharmonie, so ist der Vokal der Endung entweder i oder ü oder I oder u. Es wird derjenige Vokal angenommen, welcher dieselbe Artikulationsstelle und dieselbe Lippengestalt wie der Vokal der letzten Silbe des Wortstamms hat.[16]

Es ist natürlich nicht sehr sinnvoll, eine Sprachbeschreibung zu erstellen, die nur aus einer ungeordneten Abfolge von Beschreibungen linguistischer Phänomene besteht. Sondern man möchte, daß die Art und Weise wie linguistische Phänomene *miteinander zusammenhängen* auch in der Sprachbeschreibung deutlich wird.

---

[16] vgl. hierzu [Wendt, 1972], Seite 24f sowie [Rühl, 1970], Seite 13ff.

Wie man an den Beispielen sieht, sind linguistische Phänomene nicht an die Existenz einer *bestimmten* Sprache gebunden in denen sie vorkommen oder nicht vorkommen. Vielmehr definiert jedes linguistische Phänomen eine Teilmenge der natürlichen Sprachen, nämlich die Menge derjenigen Sprachen, in denen dieses linguistische Phänomen auftritt. Eine einzelne Sprache wiederum ist definiert durch die Menge der linguistischen Phänomene, welche diese Sprache aufweist. Dabei ist weder ausgeschlossen, daß die Menge der von einem linguistischen Phänomen definierten Sprachen leer ist, noch daß sie identisch mit der Menge der natürlichen Sprachen selbst ist. Ist letzteres der Fall, so handelt es sich bei dem linguistischen Phänomen um eine Universalie. Setzt man für die in einer Beschreibung eines linguistischen Phänomens vorkommende Variable **L** den Namen einer konkreten Sprache (z.B. Chinesisch) ein, so geht die Beschreibung in einen Satz über, der wahr oder falsch ist, je nachdem, ob das beschriebene Phänomen in der Sprache vorkommt oder nicht. Ein Beispiel für ein linguistisches Phänomen, welches in keiner natürlichen Sprache vorkommt, ist etwa durch die folgende Beschreibung gegeben:

> Das Lautinventar der Sprache **L** besteht nur aus Konsonanten.

An Feststellungen der Art, in welchen Sprachen ein linguistisches Phänomen auftritt, bzw. daß ein linguistisches Phänomen in keiner, genau einer, mehreren oder allen natürlichen Sprachen vorkommt, ist man in der Linguistik ja gerade interessiert.

Im allgemeinen sind linguistische Phänomene nicht auf einzelne Sprachen beschränkt. Vielmehr treten *gleiche* oder *ähnliche* linguistische Phänomene in verschiedenen Sprachen auf. Der Begriff der Ähnlichkeit zweier linguistischer Phänomene muß noch erklärt werden. Als Beispiel benutze ich das linguistische Phänomen *Vokalharmonie*. Vokalharmonie gibt es in der türkischen Sprache. Die Beschreibung dieses Phänomens, so wie es im Türkischen vorliegt, lautet folgendermaßen:

> Jede Wortform der Sprache **L**, die nicht aus einer anderen Sprache entlehnt ist, ist vokalharmonisch in dem Sinne, daß jeder Suffixnukleus, der nicht zum Wortstamm der Wortform gehört, die gleiche *Artikulationsstelle* wie der ihm vorhergehende Nukleus hat und jeder *enge* Suffixnukleus darüber hinaus auch die gleiche *Lippengestalt* wie der ihm vorhergehende Nukleus.

Diese Beschreibung trifft auf das Türkische zu. Die Vokalharmonie im Türkischen ist also eine *bestimmte* Einschränkung des gemeinsamen Vorkommens von

Vokalen in türkischen Wörtern. Das gemeinsame Vorkommen von Vokalen einzuschränken ist jedoch noch auf viele andere Weisen möglich.[17]

Die Vokalharmonie im Türkischen ist somit lediglich *ein* Spezialfall eines allgemeineren linguistischen Phänomens, welches nur Vokalharmonie genannt wird. Vokalharmonie wird von [Bußmann, 1990] definiert als:

> (1) Im weiteren Sinne: Jede Form von qualitativer Angleichung zwischen Vokalen hinsichtlich ihres Artikulationsortes durch gegenseitige Beeinflussung (...). Vokalharmonie ist ein assimilatorischer Vorgang, der phonetisch als Artikulationserleichterung erklärbar ist.
>
> (2) Im engeren Sinne: Qualitative Abhängigkeit des Suffixvokals vom Wurzelvokal (...).[18]

Dieses allgemeinere Phänomen ist in gewisser Weise abstrakt, da es lediglich zum Ausdruck bringt, *daß* eine Einschränkung des gemeinsamen Vorkommens von Vokalen vorliegt, aber nicht, *wie* diese realisiert ist.

Aber auch diese allgemeinere Form der Vokalharmonie ist wiederum ein Spezialfall eines noch allgemeineren Phänomens, welches *Assimilation* genannt wird. Diese wird von [Bußmann, 1990] definiert als:

> Vorgang und Ergebnis der artikulatorischen Anpassung eines Sprachlautes an einen benachbarten Laut in bezug auf ein oder mehrere Merkmale.

Linguistische Phänomene können also durch eine *Spezialisierungsrelation* zueinander in Beziehung stehen. Ein linguistisches Phänomen P ist eine Spezialisierung eines linguistischen Phänomens P', wenn für jede Sprache L gilt: tritt P in L auf, dann tritt auch P' in L auf. Die Spezialisierungsrelation ist transitiv. Zwei linguistische Phänomene sind einander ähnlich, wenn das eine eine Spezialisierung des anderen ist oder wenn sie beide Spezialisierungen eines allgemeineren linguistischen Phänomens sind. Wie allerdings die beiden Beschreibungen zur Vokalharmonie allgemein und zur türkischen Vokalharmonie zeigen, liegt dieses Folgeverhältnis aufgrund der natürlichsprachlichen Beschreibung nicht in dem Maße offen, wie dies in einer formalisierten Fassung der Fall wäre. Dann nämlich würde die Spezialisierungsrelation durch den Folgerungsbegriff ausgedrückt werden. Insbesondere in einem *Klassifikationssystem* für linguistische Phänomene muß die Spezialisierungsrelation berücksichtigt werden.

---

[17] Vokalharmonie gibt es außer im Türkischen bespielsweise auch im Finnischen und Mongolischen.

[18] Im Türkischen liegt also eine Vokalharmonie im engeren Sinne vor.

Ich möchte das Verhältnis zwischen einer natürlichsprachlichen und einer formalen Beschreibung am Beispiel der Assimilation näher erläutern.

Beispiel 1 Bei Assimilation unterscheidet man (unter anderem) zwei einander ausschließende Spezialfälle, nämlich die totale und die partielle Assimilation. Der Einfachheit halber behandle ich nicht den Vorgang, sondern nur das Ergebis einer Assimilation. Das Ergebnis einer Assimilation schlägt sich in einer bestimmten Beziehung zwischen zwei Sprachlauten nieder. Im folgenden sind die natürlichsprachlichen und die formalsprachlichen Fassungen von Phänomenbeschreibungen direkt nacheinander aufgeführt.

Als formale Beschreibungssprache benutze ich eine Sprache 1. Stufe, welche das folgende nicht-logische Vokabular enthält:

- Eine Menge von Variablen $\{s, s_1, s_2, \ldots\}$ der Sorte **Sprachlaut**.
- Eine Menge von Variablen $\{m, m_1, m_2, \ldots\}$ der Sorte **Merkmal**.
- Eine Menge von Variablen $\{w, w', \ldots\}$ der Sorte **Wert**.
- Die Prädikate *assimilation/2*, *partielle_assimilation/2*, *totale_assimilation/2*, mit $assimilation(s_1, s_2)$ bzw. $partielle\_assimilation(s_1, s_2)$ bzw. $totale\_assimilation(s_1, s_2)$ genau dann, wenn zwischen s_1 und s_2 liegt Assimilation bzw. partielle Assimilation bzw. totale Assimilation vor.
- Eine (möglicherweise partielle) zweistellige Funktion *merkmal*, so daß $merkmal(s, m) = w$ genau dann, wenn der Sprachlaut s das Merkmal m mit dem Wert w hat.

Sprachlaute sind durch eine Menge von Merkmalen und deren Werte vollständig definiert. Das heißt, es gilt

Axiom 1

- Zwei Sprachlaute sind identisch genau dann, wenn sie dieselben Merkmale mit denselben Werten haben.
- $\forall s_1 \forall s_2 (s_1 = s_2 \Leftrightarrow \forall m (merkmal(s_1, m) = merkmal(s_2, m)))$.

Eine interessante Frage ist, ob alle Sprachlaute die gleichen Merkmale haben (mit je unterschiedlichen Werten). Das heißt mit anderen Worten: Ist die Funktion *merkmal* partiell oder total. Sollte letzteres der Fall sein, dann gilt folgendes Axiom:

Axiom 2

- Alle Sprachlaute haben dieselben Merkmale (mit je unterschiedlichen Werten).
- $\forall s \forall m \exists w (merkmal(s, m) = w)$.

Ich betrachte im folgenden Assimilation im Sinne von

Definition 1

- Assimilation liegt zwischen zwei Sprachlauten vor, wenn sie ein gemeinsames Merkmal haben und dieses gemeinsame Merkmal denselben Wert hat.
- $\forall s_1 \forall s_2 (assimilation(s_2, s_1) :\Leftrightarrow$
  $\exists m (merkmal(s_1, m) = merkmal(s_2, m)))$.

Der einfachere Spezialfall ist die totale Assimilation, die sich formal sehr einfach darstellen läßt:

Definition 2

- Eine Assimilation zwischen zwei Sprachlauten ist total, wenn die Sprachlaute identisch sind.
- $\forall s_1 \forall s_2 (totale\_assimilation(s_1, s_2) :\Leftrightarrow s_1 = s_2)$

Aus dieser Definition folgt unmittelbar wegen Axiom 1.2:

Korollar 1

- Eine Assimilation zwischen zwei Sprachlauten ist total, wenn die Sprachlaute dieselben Merkmale mit denselben Werten haben.
- $\forall s_1 \forall s_2 (totale\_assimilation(s_1, s_2) \Leftrightarrow$
  $\forall m (merkmal(s_1, m) = merkmal(s_2, m)))$.

Eine Definition der partiellen Assimilation wäre etwa

Definition 3

- Eine Assimilation zwischen zwei Sprachlauten ist partiell, wenn die Sprachlaute nicht identisch sind.
- $\forall s_1 \forall s_2 (partielle\_assimilation(s_1, s_2) :\Leftrightarrow$
  $assimilation(s_1, s_2) \wedge s_1 \neq s_2)$.

Es folgt analog zu oben

Korollar 2

- Eine Assimilation zwischen zwei Sprachlauten ist partiell, wenn die Sprachlaute nicht dieselben Merkmale mit denselben Werten haben.
- $\forall s_1 \forall s_2 (partielle\_assimilation(s_1, s_2) \Leftrightarrow assimilation(s_1, s_2) \wedge \neg(\forall m(merkmal(s_1, m) = merkmal(s_2, m))))).$

Angenommen Axiom 2 gilt nicht. Das heißt die Funktion *merkmal* ist partiell. Für $\neg(merkmal(s, m) = w)$ müssen dann die folgenden zwei Fälle berücksichtigt werden: zum einen kann es sein, daß das Merkmal m für den Sprachlaut s nicht definiert ist, zum anderen kann es sein, daß s zwar das Merkmal m hat, aber dieses Merkmal einen von w verschiedenen Wert hat.

Lemma 1 $\forall s \forall m \forall w(\neg(merkmal(s, m) = w) \Leftrightarrow \neg\exists w'(merkmal(s, m) = w' \vee \exists w'(w' \neq w \wedge merkmal(s, m) = w')))).$

Die Annahme, daß *merkmal* partiell ist, hat allerdings eine unerfreuliche Konsequenz. Dies zeigt sich, wenn man einige logische Umformulierungen vornimmt:

$$partielle\_assimilation(s_1, s_2)$$

$$\overset{(Kor.2)}{\Longleftrightarrow} assimilation(s_1, s_2) \wedge \neg(\forall m(merkmal(s_1, m) = merkmal(s_2, m)))$$

$$\Longleftrightarrow assimilation(s_1, s_2) \wedge \exists m \neg(merkmal(s_1, m) = merkmal(s_2, m)))$$

$$\overset{(Def.1)}{\Longleftrightarrow} assimilation(s_1, s_2) \wedge \exists m(\neg\exists w'(merkmal(s_1, m) = w') \vee \exists w'(w' \neq merkmal(s_2, m) \wedge merkmal(s_1, m) = w')))$$

Setzt man jedoch die Gültigkeit von Axiom 2 voraus. Denn dann läßt sich die Äquivalenzenkette fortsetzen zu

$$\overset{(Def.2)}{\Longleftrightarrow} assimilation(s_1, s_2) \wedge \exists m(\exists w'(w' \neq merkmal(s_2, m) \wedge merkmal(s_1, m) = w')))$$

Und daraus ergibt sich

Theorem 1

- Eine Assimilation zwischen zwei Sprachlauten ist partiell, wenn es ein Merkmal gibt bei dem die Sprachlaute unterschiedliche Werte haben.

- $\forall s_1 \forall s_2 (partielle\_assimilation(s_1, s_2) \Leftrightarrow assimilation(s_1, s_2) \wedge \exists m(merkmal(s_1, m, w) \neq merkmal(s_2, m, w')))$.

Ohne die Voraussetzung von Axiom 2 wäre Theorem 1 nicht gültig und dies wäre gleichbedeutend damit, daß eine partielle Assimilation zwischen zwei Sprachlauten auch dann vorliegt, wenn sie bei mindestens einem Merkmal den gleichen Wert haben und alle anderen Merkmale für wenigstens einen der beiden Sprachlaute undefiniert sind. Theorem 1.2 müßte dann ersetzt werden durch

2' $\forall s_1 \forall s_2 (partielle\_assimilation(s_1, s_2) \Leftrightarrow assimilation(s_1, s_2) \wedge \exists m(merkmal(s_1, m, w) \neq merkmal(s_2, m, w'))) \vee \neg \exists w'(merkmal(s_1, m) = w' \vee merkmal(s2, m) = w')$.

An diesem Beispiel ist deutlich geworden, daß eine formalisierte Beschreibung von linguistischen Phänomenen an Präzision gewinnt, aber an Lesbarkeit einbüßt. Es ist ein großer Vorteil der Verwendung einer formalen Darstellungssprache, daß diese das Ziehen von Schlüssen aus gegebenen Beschreibungen auf ein solides Fundament stellt.

Andererseits gibt es viele linguistische Phänomene, die bislang noch nicht befriedigend formal behandelt werden können. Es können also lediglich Teilbereiche formal dargestellt werden.

3 Einteilungen und Klassifikationssysteme

In Abschnitt 2.1 wurde gesagt, daß eine Sprache L beschrieben wird, indem man die linguistischen Phänomene beschreibt, welche in der Sprache L auftreten. Man wird darüber hinaus auch solche linguistische Phänomene erwähnen, die in der Sprache *nicht* auftreten, wenn dieses Nichtauftreten bemerkenswert erscheint. Des weiteren wurde in Abschnitt 2.2 schon darauf hingewiesen, daß es notwendig ist, für diese Beschreibungen ein *Klassifikationssystem* zu finden, welches der zwischen Beschreibungen bestehenden Spezialisierungsrelation gerecht wird.

Klassifikationssysteme[19] zu einer gegebenen Menge M werden üblicherweise als Bäume definiert. Ein solcher Baum hat als Wurzel die Menge M selbst und für jeden Knoten gilt, daß seine unmittelbaren Nachfolger eine Menge von nicht-leeren, echten Teilmengen von M ist. Man fordert, daß ein

---

[19] Ausgangspunkt für die folgenden Definitionen waren die unter dem Stichwort *Klassifikation* aus [Dorn, 1980], Band2, vorgefundenen Definitionen. Insbesondere habe ich davon den Begriff der *Einteilung* übernommen.

Knoten, der kein Blatt ist, mindestens zwei Nachfolger hat, da man sonst nicht ernsthaft von Klassifikation sprechen kann. Dieses Verhältnis zwischen einem Knoten im Klassifikationsbaum und seinen unmittelbaren Nachfolgern ist in der folgenden Definition von *Einteilungen* festgehalten. Eine Einteilung einer Menge M ist eine endliche, mindestens zweielementige Menge von Teilmengen von M, genannt *Einteilungsklassen*. Diese Bedingungen sind sehr schwach. Nach ihnen ist jede mindestens zweielementige endliche Teilmenge von $Pot(M)\backslash\{\emptyset, M\}$ eine Einteilung von M.[20] Diese Bedingungen können in zweierlei Hinsicht verschärft werden. Die erste besteht darin, zu fordern, daß alle Einteilungsklassen einer Einteilung paarweise zueinander *disjunkt* sind. Die zweite fordert, daß die Menge M von den Einteilungsklassen der Einteilung *überdeckt* wird. Das heißt, daß jedes Element von M in mindestens einer Einteilungsklasse enthalten ist. Schließlich kann man noch beides zusammen fordern, dann erhält man eine *Partition* der Menge M.

Definition 4 Sei M eine Menge

1. $\mathbf{M}$ ist eine *Einteilung* von M genau dann, wenn
   (a) $2 \leq |\mathbf{M}| < \infty$
   (b) $\forall M_i \in \mathbf{M} : \emptyset \neq M_i \subsetneq M$

   M heißt *Einteilungsganzes*. Die Elemente von $\mathbf{M}$ werden *Einteilungsklassen* von M bzw. aus $\mathbf{M}$ genannt.

2. $\mathbf{M} = \{M_1, \ldots, M_n\}$ ist eine *scharfe Einteilung* von M genau dann, wenn die Einteilungsklassen $M_1, \ldots, M_n$ paarweise disjunkt sind. Andernfalls heißt $\mathbf{M}$ *unscharfe Einteilung* von M.

3. $\mathbf{M} = \{M_1, \ldots, M_n\}$ ist eine *vollständige Einteilung* von M genau dann, wenn $\bigcup\{M_1, \ldots, M_n\} = M$. Andernfalls heißt $\mathbf{M}$ *unvollständige Einteilung* von M.

4. Eine scharfe und vollständige Einteilung $\mathbf{M}$ von M wird *korrekte Einteilung* von M genannt. Andernfalls heißt $\mathbf{M}$ *unkorrekte Einteilung* von M.

Lemma 2 Sei M eine Menge und $\mathbf{M}$ Einteilung von M. Dann gilt:

1. $\forall M_i \in \mathbf{M}\ \exists x, x' \in M : x \in M_i \wedge x' \notin M_i$.

---

[20] Damit ist zum Beispiel auch zugelassen, daß zwei Einteilungsklassen in der (echten) Teilmengenrelation zueinander stehen.

2. Ist **M** korrekte Einteilung von M, dann ist **M** eine Partition von M[21].

Beweis:
(1): . Für jedes Element M_i von $Pot(M) \backslash \{\emptyset, M\}$ gibt es Elemente $x, x' \in M$, so daß $x \in M_i$ und $x' \notin M_i$.
(2) folgt direkt aus Definition 4.

□

In konkreten Anwendungen fassen Einteilungen *in gewisser Hinsicht gleiche Elemente* in Einteilungsklassen zusammen. Eine Einteilungsklasse wird dabei normalerweise mittels eines einstelligen Prädikates definiert. Nimmt man beispielsweise die Menge der Menschen als Einteilungsganzes, so definieren die Prädikate *weiblich/1* und *maennlich/1* die Einteilung $\mathbf{M} = \{\{x \mid weiblich(x)\}, \{x \mid maennlich(x)\}\}$ auf der Menge der Menschen. Allerdings wird hier vorausgesetzt, daß $\{\{weiblich(x), maennlich(x)\}\}$ ein *Einteilungskriterium* von M gemäß der folgenden Definition ist.

Definition 5 Sei M eine Menge. Eine Menge $\mathbf{P} = \{P_1, \ldots, P_n\}$ von einstelligen Prädikaten ist ein *(scharfes, vollständiges, korrektes) Einteilungskriterium* von M genau dann, wenn die Menge $\{P_1(M), \ldots, P_n(M)\}$ eine (scharfe vollständige, korrekte) Einteilung von M ist.[22]
Die Prädikate $P_1, \ldots, P_n$ heißen *Einteilungsprädikate* von M bzw. aus **P**.

Es können aber auch beliebige *Merkmale* (z.b. ALTER, AUGENFARBE, INTELLIGENZ[23].), welche auf Elementen von M definiert bzw. definierbar sind zur Definition von Einteilungen benutzt werden. Ein auf einer Menge M definiertes Merkmal mit Werten in W ist eine Funktion $f : M \longrightarrow W$. Ist das Einteilungsganze M die Menge der Menschen, so sind die Wertebereiche der Merkmale ALTER bzw. AUGENFARBE die Mengen $\mathbb{N}$ bzw. {gruen, blau, braun, schwarz, grau, rot}.

Das nächste Theorem formuliert die Bedingung, unter der es möglich ist, eine Einteilung von einer Menge M mithilfe einer Funktion $f : M \longrightarrow W$ zu definieren. Diese Bedingung lautet, daß auf der Menge der $f(M)$[24], also der Menge der Elemente aus W, auf welche f ein Element von M abbildet, eine Einteilung definierbar sein muß. Und dies ist immer der Fall, sobald $f(M)$

---

[21] Jede korrekte Einteilungen ist eine Partition, aber nicht umgekehrt, da Partitionen mit nur einer Einteilungsklasse oder unendlich vielen Einteilungsklassen keine Einteilungen sind.

[22] Ist P ein einstelliges Prädikat, dann ist $P(M) := \{x \in M \mid P(x)\}$.

[23] vgl. [Dorn, 1980], Band2, Stichwort *Klassifikation*

[24] Für jede Funktion $f : M \longrightarrow W$ ist $f(M) := \{y \in W \mid \exists x \in M : y = f(x)\}$. $f(M)$ heißt Bildmenge von M unter f oder f-Bild von M.

mindestens zwei Elemente hat. Jede solche Einteilung des Bildbereichs definiert eindeutig eine Einteilung von M. Darüber hinaus ist eine so erhaltene Einteilung von M scharf bzw. vollständig bzw. korrekt, wenn die Einteilung von $f(M)$ scharf bzw. vollständig bzw. korrekt ist.

Theorem 2 Ist M eine Menge und f eine Funktion von M nach W und ist $\mathbf{Q} = \{Q_1, \ldots, Q_n\}$ (scharfe, vollständige, korrekte) Einteilung von $f(M)$, dann ist $\mathbf{P} = \{\{x \mid f(x) \in Q_1\}, \ldots, \{x \mid f(x) \in Q_n\}\}$ eine (scharfe, vollständige, korrekte) Einteilung von M. $\mathbf{P}$ heißt die von f bezüglich $\mathbf{Q}$ auf M induzierte Einteilung.

Beweis: Seien die Voraussetzungen erfüllt.
(1)

$$\mathbf{Q} = \{Q_1, \ldots, Q_n\} \text{ ist Einteilung von } f(M)$$

$$\overset{(Def.4.1)}{\iff} 2 \leq |\mathbf{Q}| < \infty \wedge \forall Q \in \mathbf{Q} : \emptyset \neq Q \subsetneqq f(M)$$

Sei $\mathbf{Q} \ni Q_i \neq Q_j \in \mathbf{Q}$. Dann gibt es nach Lemma 2.1 ein $y \in f(M)$, so daß $(y \in Q_i \wedge y \notin Q_j) \vee (y \notin Q_i \wedge y \in Q_j)$. Ohne Beschränkung der Allgemeinheit sei $(y \in Q_i \wedge y \notin Q_j)$. Da $Q_i \subsetneqq f(M)$, gibt es ein $x \in M : f(x) = y$. Also ist $f(x) \in Q_i \wedge f(x) \notin Q_j$ und damit ist $\{x \mid f(x) \in Q_i\} \neq \{x \mid f(x) \in Q_j\}$. Also ist $|\mathbf{P}| = |\mathbf{Q}|$ und folglich $2 \leq |\mathbf{P}| < \infty$. Aus $\emptyset \neq Q_i \subsetneqq f(M)$ folgt $\emptyset \neq \{x \mid f(x) \in Q_i\} \subsetneqq M$.

(2)

$$\mathbf{Q} \text{ ist scharfe Einteilung von } f(M)$$

$$\overset{(Def.4.2)}{\Longrightarrow} \forall i \neq j \in \{1, \ldots, n\} \; \forall y \in f(M) : (y \in Q_i \Rightarrow y \notin Q_j).$$

$$\Longrightarrow \forall i \neq j \in \{1, \ldots, n\} \; \forall x \in M : (f(x) \in Q_i \Rightarrow f(x) \notin Q_j).$$

$$\Longrightarrow \forall i \neq j \in \{1, \ldots, n\} : \; \{x \mid f(x) \in Q_i\} \cap \{x \mid f(x) \in Q_j\} = \emptyset$$

$$\overset{(Def.4.2)}{\Longrightarrow} \mathbf{P} \text{ ist scharfe Einteilung von } M$$

(3)

$$\mathbf{Q} \text{ ist vollständige Einteilung von } f(M)$$

$$\overset{(Def.4.3)}{\Longrightarrow} \bigcup \{Q_1, \ldots, Q_n\} = f(M)$$

$$\Longrightarrow \forall y \in f(M) \; \exists i \in \{1, \ldots, n\} : y \in Q_i$$

$$\Longrightarrow \forall x \in M \; \exists i \in \{1, \ldots, n\} : f(x) \in Q_i$$

$$\Longrightarrow \bigcup\{\{x \mid f(x) \in Q_1\}, \ldots, \{x \mid f(x) \in Q_n\}\} = M$$

$$\overset{(Def. 4.3)}{\Longrightarrow} \mathbf{P} \text{ ist vollständige Einteilung von } M$$

□

Hat man zu einer Menge M mehrere Einteilungen $\mathbf{E_1}, \ldots, \mathbf{E_n}$ so möchte man unter Umständen jede Einteilung verwenden. Hierbei gibt es zwei Vorgehensweisen, die mit den Schlagwörtern *sequentiell* und *parallel* charakterisierbar sind. Was dies bedeutet, soll zunächst an einem Beispiel erläutert werden.

Beispiel 2 Einteilungsganzes sei die Menge M der Kookkurrenzrestriktionen in Phonemfolgen. Zu M seien die nachfolgend definierten Einteilungen $\mathbf{P} = \{P_1, P_2\}, \mathbf{Q} = \{Q_1, Q_2\}$ und $\mathbf{R} = \{R_1, R_2, R_3\}$ gegeben:

1. $x \in Q_1 :\Leftrightarrow x$ ist eine Kookkurrenzrestriktion zwischen benachbarten Phonemen.
2. $x \in Q_2 :\Leftrightarrow x$ ist eine Kookkurrenzrestriktion zwischen nicht benachbarten Phonemen.
3. $x \in P_1 :\Leftrightarrow x$ ist eine assimilierende Kookkurrenzrestriktion.
4. $x \in P_2 :\Leftrightarrow x$ ist eine dissimilierende Kookkurrenzrestriktion.
5. $x \in R_1 :\Leftrightarrow x$ ist eine progressive Kookkurrenzrestriktion.
6. $x \in R_2 :\Leftrightarrow x$ ist eine regressive Kookkurrenzrestriktion.
7. $x \in R_3 :\Leftrightarrow x$ ist eine reziproke Kookkurrenzrestriktion.

Diese drei Einteilungen sind korrekt. Sie sind auch voneinander logisch unabhängig: daß eine Phonemfolge in einer bestimmten Einteilungsklasse der einen Einteilung fällt, sagt nichts darüber aus, in welche Einteilungsklasse einer anderen Einteilung diese Phonemfolge fällt. Es ist sinnvoll, nach allen drei Einteilungen zu klassifizieren.

Die sequentielle Vorgehensweise besteht in der Konstruktion eines Baums gemäß des nachfolgenden Verfahrens.

Verfahren 1 Sei M das Einteilungsganze und seien $\mathbf{E_1} \ldots \mathbf{E_n}$ Einteilungen von M.

1. M ist die Wurzel des Baums, $\mathbf{K} := \{M\}$ ist die Menge der expandierbaren Knoten und $\mathbf{E} := \{\mathbf{E_1} \ldots \mathbf{E_n}\}$ ist die Menge der noch nicht angewendeten Einteilungen.
2. Solange $\mathbf{E} \neq \emptyset$: Wähle beliebig eine Einteilung E_i aus $\mathbf{E}$ aus, und definiere zu jedem expandierbaren Knoten $K \in \mathbf{K}$ und jeder Einteilungsklasse E aus $\mathbf{E_i}$ den Durchschnitt $K \cap E$ als unmittelbaren Nachfolger von K. Ist ein so erhaltener Durchschnitt die leere Menge, dann breche das Verfahren als gescheitert ab. Ist letzteres nicht der Fall, dann entferne $\mathbf{E_i}$ aus $\mathbf{E}$ und ersetze alle Elemente in $\mathbf{K}$ durch deren unmittelbare Nachfolger.

Bei dem so konstruierten Baum handelt es sich um einen ODER-Baum. Diese sind wie folgt definiert:

Definition 6 Sei S eine Menge.

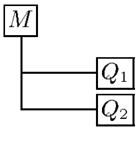
1. Ist $x \in S$, dann ist $(x, \emptyset)$ ein ODER-Baum (über S).
2. Ist $x \in S$ und sind $B_1, \ldots, B_n$ ODER-Bäume (über S), dann ist $(x, \{B_1, \ldots, B_n\})$ ein ODER-Baum (über S).

Im Beispiel ist die Menge S die Potenzmenge von M.

Die Anwendung des sequentiellen Verfahrens auf das Beispiel ergibt folgendes:
1. Schritt: M ist Wurzel des Baums, $\mathbf{K} := \{M\}, \mathbf{E} := \{\mathbf{Q}, \mathbf{P}, \mathbf{R}\}$.

Der Baum nach Schritt 1

2. Schritt: $\mathbf{Q}$ wird aus $\mathbf{E}$ gewählt, $\mathbf{K} := \{Q_1, Q_2\}$[25], $\mathbf{E} := \{\mathbf{P}, \mathbf{R}\}$.

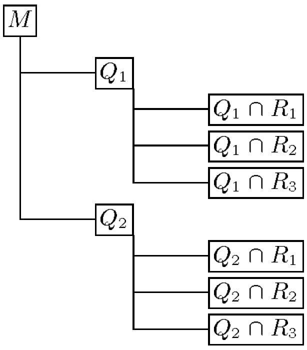


Der Baum nach Schritt 2

---

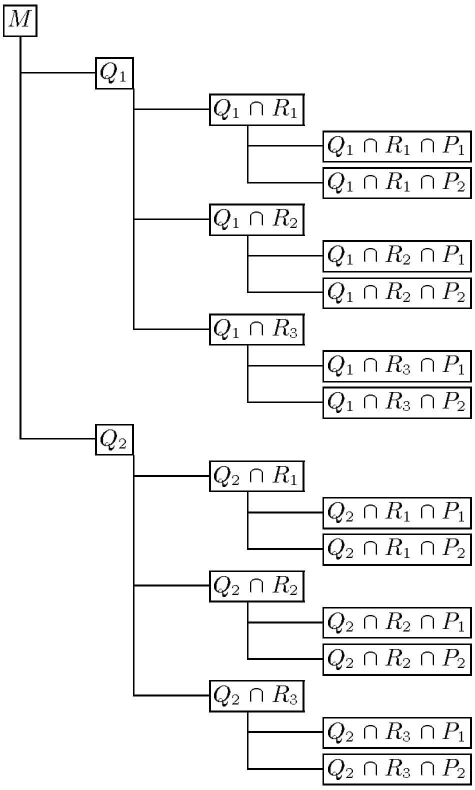
[25] Korrekterweise sollten hier und auch in den Abbildungen anstelle von Q_1 und Q_2 die Durchschnitte $M \cap Q_1$ und $M \cap Q_2$ stehen. Da der Schnitt einer Einteilungsklasse mit dem Einteilungsganzen aber wiederum die Einteilungsklasse ergibt, habe ich die Durchschnitte mit M durch deren Ergebnis ersetzt. Dies gilt auch für die nächsten Schritte und Abbildungen.

3. Schritt: $\mathbf{R}$ wird aus $\mathbf{E}$ gewählt, $\mathbf{K} := \{Q_1 \cap R_1, Q_1 \cap R_2, Q_1 \cap R_3, Q_2 \cap R_1, Q_2 \cap R_2, Q_2 \cap R_3\}$, $\mathbf{E} := \{\mathbf{P}\}$.



Der Baum nach Schritt 3

4. Schritt: $\mathbf{P}$ wird aus $\mathbf{E}$ gewählt, $\mathbf{K} := \{Q_1 \cap R_1 \cap P_1, Q_1 \cap R_1 \cap P_2, Q_1 \cap R_2 \cap P_1, M \cap Q_1 \cap R_2 \cap P_2, Q_1 \cap R_3 \cap P_1, Q_1 \cap R_3 \cap P_2, Q_2 \cap R_1 \cap P_1, Q_2 \cap R_1 \cap P_2, M \cap Q_2 \cap R_2 \cap P_1, Q_2 \cap R_2 \cap P_2, Q_2 \cap R_3 \cap P_1, Q_2 \cap R_3 \cap P_2\}$, $\mathbf{E} := \emptyset$.



Der Baum nach Schritt 4

Die Reihenfolge, in der die Einteilungen verwendet werden, spielt keine Rolle.

Die Anzahl der Blätter bleibt immer gleich. Lediglich die Anzahl der inneren Knoten kann minimiert werden, wenn man immer eine Einteilung aus $\mathbf{E}$ auswählt, welche eine im Vergleich zu den anderen Einteilungen in $\mathbf{E}$ minimale Anzahl von Einteilungsklassen enthält.

Ergibt sich im Laufe der Anwendung des Verfahrens die leere Menge beim Schnitt eines expandierbaren Knotens und einer Einteilungsklasse, beweist dies, daß doch Abhängigkeiten zwischen den Einteilungen bestehen.

Die parallele Vorgehensweise besteht ebenfalls in der Definition eines Baums. Der Baum wird folgendermaßen konstruiert:

Verfahren 2 Sei M das Einteilungsganze und seien $\mathbf{E_1} \ldots \mathbf{E_n}$ Einteilungen von M.

> M ist die Wurzel des Baums und $\mathbf{E_1} \ldots \mathbf{E_n}$ sind die unmittelbaren Nachfolger von M.

Bei einem nach Verfahren 1 erstellten Baum ist jeder unmittelbare Nachfolger eines Knotens eine Teilmenge dieses Knotens. Ein nach Verfahren 2 erstellter Baum hat immer die Höhe 2 und jeder unmittelbare Nachfolger der Wurzel ist eine Menge von Teilmengen der Wurzel. Nach Verfahren 2 erstellte Bäume sind keine ODER-Bäume, sondern UND-ODER-Bäume. Allerdings sind die nach Verfahren 2 erstellten Bäume Spezialfälle von UND-ODER-Bäumen, wie aus der folgenden Definition zu ersehen ist. Die Definition von UND-ODER-Bäume ist wie folgt:

Definition 7 Sei S eine Menge.

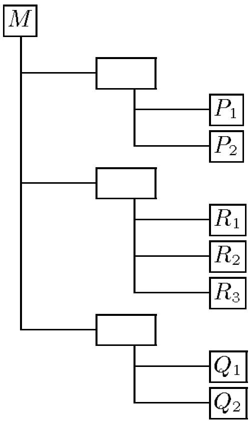
1. Ist $x \in S$, dann ist $(x, \emptyset)$ ein UND-ODER-Baum (über S).
2. Ist $x \in S$ und sind $B_1^1, \ldots, B_n^1, \ldots, B_1^i, \ldots, B_m^i$ UND-ODER-Bäume (über S), dann ist $(x, \{\{B_1^1, \ldots, B_n^1\}, \ldots, \{B_1^i, \ldots, B_m^i\}\}$ ein UND-ODER-Baum (über S).

Wie man an Definition 7 sieht, haben bei einem UND-ODER-Baum $und(M, \mathbf{M})$ nicht die Elemente von $\mathbf{M}$ Nachfolger, sondern die Elemente der Elemente von $\mathbf{M}$. Beim Baumdurchlauf ist von jedem Nachfolger ein Element zu wählen. UND-ODER-Bäume sind eine Verallgemeinerung der ODER-Bäume. Die Klasse der Bäume, die man erhält, wenn man die Bedingung 2 von Definition 7 ersetzt durch die Bedingung

2' Ist $x \in S$ und sind $B_1, \ldots, B_n$ UND-ODER-Bäume (über S), dann ist $(x, \{\{B_1, \ldots, B_n\}\})$ ein UND-ODER-Baum (über S).

ist isomorph zur Klasse der ODER-Bäume gemäß Definition 6.

Die Anwendung des parallelen Verfahrens ergibt die nachfolgend abgebildete Struktur. Die leeren Kästchen symbolisieren die Trennlinie zwischen der UND- (links) und der ODER-Ebene (rechts).



Der mit dem zweiten Verfahren erhaltene Baum

Wie das Beispiel zeigt, hat die Verwendung von UND-ODER-Bäumen gegenüber der Verwendung von ODER-Bäumen zwar den Nachteil, daß die Struktur der UND-ODER-Bäume komplexer ist als die der ODER-Bäume. Dafür sind UND-ODER-Bäume ökonomischer, denn sie benötigen zur Darstellung derselben Information deutlich weniger Knoten, als ODER-Bäume, und sie sind inhaltlich adäquater, denn sie zwingen nicht zu einer willkürlichen Hierarchisierung von Einteilungen. Allerdings kommen diese Vorteile nur zum Tragen, wenn man *mehrere* Einteilungen derselben Menge verwenden will. Dies ist jedoch nur dann möglich, wenn

nicht in jedem Fall möglich. Diese lautet, daß die Einteilungen gemäß der nachfolgenden Definition *orthogonal* sein müssen. Aus der Orthogonalität zweier Einteilungen **E** und **E'** folgt insbesondere, daß der Schnitt einer beliebigen Einteilungsklasse aus **E** mit einer beliebigen Einteilungsklasse aus **E'** nicht-leer ist und alle so erhaltenen Schnitte paarweise verschieden sind. Die erste Eigenschaft ist gleichbedeutend damit, daß Widersprüche ausgeschlossen sind. Die zweite Eigenschaft besagt, daß keine Redundanzen vorkommen.

Ausgangspunkt bildet die Orthogonalität einer Menge von Funktionen $\mathbf{F} = \{f_1 : M \longrightarrow W_1, \ldots, f_n : M \longrightarrow W_n\}$ mit *gleichem* Definitionsbereich. Diese sind orthogonal, wenn es zu beliebigen Elementen der Bildbereiche $w_1 \in f_1(W_1), \ldots, w_n \in f_n(W_n)\}$ immer ein Element M in der Menge M gibt, welches auf diese Werte abgebildet wird. Für dieses Element m gilt also $f_1(m) = w_1 \wedge \ldots \wedge f_n(m) = w_n$. Es gibt somit keinerlei Abhängigkeiten zwischen den Funktionen. Für beliebige Elemente $m \in M$ und beliebige Teilmengen $\mathbf{F}' \subsetneqq \mathbf{F}$ kann aus den Werten, auf die m von den Funktionen in **F'** abgebildet wird, nicht geschlossen werden, daß die Werte, auf die m von den Funktionen in $\mathbf{F}\backslash\mathbf{F}'$ abgebildet wird, in *echten* Teilmengen der Bildbereiche der Funktionen in $\mathbf{F}\backslash\mathbf{F}'$ liegen. Dies bedeutet insbesondere, daß aus dem Wert, den eine Funktion aus **F** für ein gegebenes Argument annimmt, nicht geschlossen werden kann, daß der Wert, den eine andere Funktion aus **F** zu demselben Argument annimmt, in einer *echten* Teilmenge des Bildbereichs dieser zweiten Funktion liegt. Da letztere Bedingung von Funktionen, deren Bildmenge nur ein Element enthält, trivialerweise erfüllt wird, wird explizit gefordert, daß nur solche Funktionen zueinander orthogonal sein können, deren Bildmengen mindestens zwei Elemente enthalten.

Eine Menge von Teilmengen einer gegebenen Menge M wird als orthogonal bezeichnet, wenn die Menge der zugehörigen charakteristischen Funktionen orthogonal ist. Diese Definition beruht auf derselben Motivation, wie die Definition der Orthogonalität einer Menge von Funktionen: Aus der Zugehörigkeit oder Nichtzugehörigkeit eines Elements zur einen Teilmenge sollen keinerlei Rückschlüsse auf die Zugehörigkeit oder Nichtzugehörigkeit dieses Elements zu einer anderen Teilmenge möglich sein.

Eine Menge von Einteilungen ist orthogonal, wenn jede Menge von Einteilungsklassen, die genau eine Einteilungsklasse von jeder Einteilung enthält, orthogonal ist.

Schließlich wird in analoger Weise zu Definition 5 die Orthogonalität zweier Prädikate bzw. zweier Einteilungskriterien definiert.

Definition 8

1. Eine Menge $\{f_1 : M \longrightarrow W_1, \ldots, f_n : M \longrightarrow W_n\}$ von Funktionen ist *orthogonal* genau dann,wenn

   (a) $|f_1(M)| \geq 2 \wedge \ldots \wedge |f_n(M)| \geq 2$.

   (b) $\forall w_1 \in f_1(M), \ldots, \forall w_n \in f_n(M) \exists x \in M : (f_1(x) = w_1 \wedge \ldots \wedge f_n(x) = w_n))$.

   $\{f_1 : M \longrightarrow W_1, \ldots, f_n : M \longrightarrow W_n\}$ heißt *orthogonale Funktionenmenge (bezüglich M).*

2. Eine Menge $\{M_1, \ldots, M_n\}$ von Teilmengen von M ist *orthogonal* genau dann, wenn die Menge der charakteristischen Funktionen $\{\chi_{M_1} : M \longrightarrow \{0,1\}, \ldots, \chi_{M_n} : M \longrightarrow \{0,1\}\}$ orthogonal ist.

   $\{M_1, \ldots, M_n\}$ heißt *orthogonale Menge (bezüglich M).*

3. Eine Menge $\{P_1, \ldots, P_n\}$ von einstelligen Prädikaten, so daß $\{x \mid P_1(x)\} \subseteq M \wedge \ldots \wedge \{x \mid P_n(x)\} \subseteq M$, ist *orthogonal* genau dann, wenn $\{P_1(M), \ldots, P_n(M)\}$ orthogonal ist.

4. Eine Menge $\{\mathbf{M_1}, \ldots, \mathbf{M_n}\}$ von Einteilungen von M ist *orthogonal* genau dann, wenn $\forall M_1 \in \mathbf{M_1} \ldots \forall M_n \in \mathbf{M_n} : \{M_1, \ldots, M_n\}$ ist orthogonal.

5. Eine Menge $\{\mathbf{P_1}, \ldots, \mathbf{P_n}\}$ von Einteilungskriterien von M ist *orthogonal* genau dann, wenn $\forall P_1 \in \mathbf{P_1} \ldots \forall P_n \in \mathbf{P_n} : \{P_1(M), \ldots, P_n(M)\}$ ist orthogonal.

Die folgenden zwei Theoreme geben die wichtigste Eigenschaft von orthogonalen Mengen bzw. Funktionenmengen an. Aus dem ersten Theorem folgt, daß eine Menge N von nicht-leeren echten Teilmengen einer Menge M orthogonal ist genau dann, wenn für beliebige $M_1 \in N, \ldots M_n \in N$ gilt, daß $\pm M_1 \cap \ldots \cap \pm M_n \neq \emptyset$. Dabei symbolisiert $\pm M_i$, daß an dieser Stelle sowohl die Menge M_i, als auch das Komplement $M \backslash M_i$ stehen kann. Insbesondere ist eine Menge von zwei nicht-leeren echten Teilmengen einer Menge M orthogonal genau dann, wenn für die zwei Teilmengen gilt, daß sie ein gemeinsames Element haben, daß beide jeweils ein Element enthalten, welches in der anderen nicht enthalten ist, und daß es ein Element von M gibt, das in beiden nicht enthalten ist. Negativ formuliert bedeutet dies, daß eine Menge von zwei nicht-leeren echten Teilmengen einer Menge M orthogonal ist genau, wenn die Elemente *nicht* disjunkt sind, *keines* im anderen enthalten ist und beide zusammengenommen die Menge M *nicht* vollständig überdecken.

Das zweite Theorem besagt, daß die Menge der Einteilungen, welche von den Elementen einer orthogonalen Funktionenmenge induziert werden, orthogonal ist.

Theorem 3 Eine Menge $\{M_1, \ldots, M_n\}$ von Teilmengen von M ist orthogonal genau dann, wenn $\forall (b_1, \ldots, b_n) \in \{0, 1\}^n \ \exists x \in M : \chi_{M_1}(x) = b_1 \wedge \ldots \wedge \chi_{M_n}(x) = b_n$.

Beweis: Seien Voraussetzungen erfüllt.

$$\{M_1, \ldots, M_n\} \text{ ist orthogonal}$$

$$\overset{(Def.8.2)}{\Longleftrightarrow} \{\chi_{M_1} : M \longrightarrow \{0,1\}, \ldots, \chi_{M_n} : M \longrightarrow \{0,1\}\} \text{ ist orthogonal}$$

$$\overset{(Def.8.1)}{\Longleftrightarrow} |\chi_{M_1}(M)| \geq 2 \wedge \ldots \wedge |\chi_{M_n}(M)| \geq 2 \text{ und } \forall b_1 \in \chi_{M_1}(M) \ldots \forall b_n \in \chi_{M_n}(M) \exists x \in M : (\chi M_1(x) = b_1 \wedge \ldots \wedge \chi_{M_n}(x) = b_n)).$$

$\Longleftrightarrow$ $|\chi_{M_1}(M)| = \{0,1\}\wedge\ldots\wedge|\chi_{M_n}(M)| = \{0,1\}$ und $\forall b_1 \in \chi_{M_1}(M)\ldots\forall b_n \in \chi_{M_n}(M)\exists x \in M : (\chi M_1(x) = b_1 \wedge \ldots \wedge \chi_{M_n}(x) = b_n))$.

$\Longleftrightarrow$ $\forall(b_1,\ldots,b_n) \in \{0,1\}^n\ \exists x \in M : \chi_{M_1}(x) = b_1 \wedge \ldots \wedge \chi_{M_n}(x) = b_n$.

□

Hilfssatz 1

Sei $f : M \longrightarrow W$ eine Funktion, sei $\mathbf{Q}$ eine Einteilung von $f(M)$ und sei für $Q_i \in \mathbf{Q} : M_i = \{x \mid f(x) \in Q_i\}$. Dann gilt:

$$\forall m \in M : \chi_{M_i}(m) = b \Leftrightarrow \chi_{M_i}(\{m' \mid f(m) = f(m')\}) = \{b\}.$$

Beweis: Seien Voraussetzungen erfüllt.
"$\Longrightarrow$" Angenommen $\chi_{M_i}(m) = b$ und $\chi_{M_i}(\{m' \mid f(m) = f(m')\}) \neq \{b\}$. Dann gibt es ein $m^\star$, so daß $f(m^\star) = f(m) \wedge \chi_{M_i}(m^\star) \neq b$. Nun gilt aber $\chi_{M_i}(m^\star) = \chi_{Q_i}(f(m^\star)) = \chi_{Q_i}(f(m)) = \chi_{M_i}(m)$ im Widerspruch zur Annahme. "$\Longleftarrow$" Angenommen $\chi_{M_i}(m) \neq b$ und $\chi_{M_i}(\{m' \mid f(m) = f(m')\}) = \{b\}$. Dann ist $\chi_{M_i}(m) = b$ im Widerspruch zur Annahme.

□

Theorem 4 Sei $\{f_1 : M \longrightarrow W_1, \ldots, f_n : M \longrightarrow W_n\}$ orthogonal und seien $\mathbf{W_1}, \ldots, \mathbf{W_n}$ Einteilungen von $f_1(M), \ldots, f_n(M)$. Dann ist die Menge der von $\mathbf{W_1}, \ldots, \mathbf{W_n}$ induzierten Einteilungen orthogonal.

Beweis: Die von $\mathbf{W_1}, \ldots, \mathbf{W_n}$ induzierten Einteilungen von M seien $\mathbf{M_1}, \ldots, \mathbf{M_n}$.

$\mathbf{M_1}, \ldots, \mathbf{M_n}$ ist orthogonal

$\overset{(Def.8.4)}{\Longleftrightarrow}$ $\forall M_1 \in \mathbf{M_1} \ldots \forall M_n \in \mathbf{M_n} : \{M_1, \ldots, M_n\}$ ist orthogonal

$\overset{(Th.3)}{\Longleftrightarrow}$ $\forall M_1 \in \mathbf{M_1} \ldots \forall M_n \in \mathbf{M_n} \forall(b_1, \ldots, b_n) \in \{0,1\}^n\ \exists x \in M : \chi_{M_1}(x) = b_1 \wedge \ldots \wedge \chi_{M_n}(x) = b_n$.

Sei nun $M_1 \in \mathbf{M_1} \wedge \ldots \wedge M_n \in \mathbf{M_n}$ und sei $(b_1, \ldots, b_n) \in \{0,1\}$. Da $M_1, \ldots, M_n$ Einteilungsklassen sind, gibt es $m_1 \in M_1 \wedge \ldots \wedge m_n \in M_n$ so daß $\chi_{M_1}(m_1) = b_1 \wedge \ldots \wedge \chi_{M_n}(m_n) = b_n$.
Zu $M_1, \ldots, M_n$ gibt es nach Theorem 2 Mengen $W'_1 \subseteq W_1, \ldots, W'_n \subseteq W_n$, so daß $M_1 = \{x \mid f_1(x) \in W'_1\} \wedge \ldots \wedge M_n = \{x \mid f_n(x) \in W'_n\}$. Dann gilt:

$\chi_{M_1}(m_1) = b_1 \wedge \ldots \wedge \chi_{M_n}(m_n) = b_n$

$\overset{(Hs.1)}{\Longleftrightarrow}$ $\chi_{M_1}(\{m \mid f_1(m) = f_1(m_1)\}) = \{b_1\} \wedge \ldots \wedge \chi_{M_n}(\{m \mid f_n(m) = f_n(m_n)\}) = \{b_n\}$

Da $f_1(m_1) \in f(M) \wedge \ldots \wedge f_n(m_n) \in f(M)$ und nach Voraussetzung die Funktionenmenge $\{f_1, \ldots, f_n\}$ orthogonal ist, folgt

$$\overset{(Def.8.1.a)}{\Longrightarrow} \exists m \in M : (f_1(m) = f_1(m_1) \wedge \ldots \wedge f_n(m) = f_n(m_n)$$

$$\overset{(Def.8.1.b)}{\Longrightarrow} \exists m \in M : (\chi_{M_1}(m) = b_1 \wedge \ldots \wedge \chi_{M_n}(m) = b_n$$

$\square$

Orthogonalität als Voraussetzung zur Anwendung von Verfahren 2 (siehe Seite 19) ist eine sehr starke Bedingung, die von den meisten Mengen von Einteilungen nicht erfüllt wird. Möchte man Verfahren 2 auch auf Einteilungsmengen anwenden, die nicht orthogonal sind[26] so muß man dafür Sorge tragen, daß Widersprüche nicht auftreten können. Eine Einteilungsmenge ist widersprüchlich, wenn die Anwendung von Verfahren 1 (siehe Seite 17) auf die Einteilungsmenge scheitert. Dies ist der Fall, wenn sich bei der Durchschnittbildung die leere Menge ergibt. Um nun Einteilungsmengen verwenden zu können, die nicht orthogonal sind, müssen die leeren Durchschnitte ausgeschlossen werden.[27] Die nächste Definition stellt hierfür die Ausdrucksmittel zur Verfügung.

Definition 9 Seien $\mathbf{M_1}, \ldots, \mathbf{M_n}$ Einteilungen der Menge M. Dann ist

1. $\Cap\{\mathbf{M_1}, \ldots, \mathbf{M_n}\} := \{(M_1 \cap \ldots \cap M_n) \neq \emptyset \mid M_1 \in \mathbf{M_1} \wedge \ldots \wedge M_n \in \mathbf{M_n}\}$ der *(komponentenweise) Schnitt* von $\mathbf{M_1}, \ldots, \mathbf{M_n}$
2. $\{\mathbf{M_1}, \ldots, \mathbf{M_n}\}$ *erfüllbar (bzgl. M)* genau dann, wenn $\Cap\{\mathbf{M_1}, \ldots, \mathbf{M_n}\}$ Einteilung von M ist.

$\Cap$ ist eine Funktion, die eine (nichtleere) Menge von Einteilungen einer Menge M auf eine Teilmenge der Potenzmenge von M abbildet, die nicht notwendigerweise wiederum eine Einteilung von M ist. Wie man jedoch leicht sieht, ist der Schnitt einer orthogonalen Menge von Einteilungen wieder eine Einteilung, da jede Durchschnittbildung nach Theorem 3 eine nicht-leere Menge ergibt. Handelt es sich um eine orthogonale Menge von *korrekten* Einteilungen, sind die Durchschnitte darüber hinaus auch noch paarweise verschieden, so daß der Schnitt die maximal mögliche Anzahl an Elementen enthält.

Theorem 5 Ist $\{\mathbf{M_1}, \ldots, \mathbf{M_n}\}$ eine orthogonale Menge von korrekten Einteilungen von M, dann ist $| \Cap \{\mathbf{M_1} \cap \ldots \cap \mathbf{M_n}\}| = |\mathbf{M_1}| \cdot \ldots \cdot |\mathbf{M_n}|$.

---

[26] Dies ist durchaus sinnvoll, wenn die Einteilungsmengen $\{\mathbf{M_1}, \ldots, \mathbf{M_n}\}$ *fast* orthogonal sind, also es vergleichsweise wenige Einteilungsklassen $M_1 \in \mathbf{M_1}, \ldots, M_n \in \mathbf{M_n}$ gibt, so daß $M_1 \cap \ldots \cap M_n = \emptyset$.

[27] Dies wird in der später vorgestellten Implementation in Form von *Constraints* geschehen.

Beweis:

$$\{\mathbf{M_1}, \ldots, \mathbf{M_n}\} \text{ ist orthogonal}$$

$$\overset{(Def.8.4)}{\Longrightarrow} \forall M_1 \in \mathbf{M_1} \ldots \forall M_n \in \mathbf{M_n} : \{M_1, \ldots, M_n\} \text{ ist orthogonal}$$

$$\overset{(Th.3)}{\Longrightarrow} \forall M_1 \in \mathbf{M_1} \ldots \forall M_n \in \mathbf{M_n} : M_1 \cap \ldots \cap M_n \neq \emptyset$$

Es bleibt zu zeigen ist, daß *alle* erhaltenen Schnitte paarweise voneinander verschieden sind. Angenommen es gibt $M_1, M_1' \in \mathbf{M_1} \ldots M_n, M_n' \in \mathbf{M_n}$ so daß $M_1 \cap \ldots \cap M_n = M_1' \cap \ldots \cap M_n'$ und $\mathbf{M_i} \ni M_i \neq M_i' \in \mathbf{M_i}$. Da $\mathbf{M_i}$ nach Voraussetzung korrekte Einteilung ist, folgt $M_i \cap M_i' = \emptyset$ und damit $M_1 \cap \ldots \cap M_n \neq M_1' \cap \ldots \cap M_n'$ im Widerspruch zur Annahme.

□

Bemerkung 1 Aus der Orthogonalität jeder zwei-elementigen Teilmenge einer Menge $\mathbf{M} = \{\mathbf{M_1}, \ldots, \mathbf{M_n}\}$ von Einteilungen folgt nicht, daß $\mathbf{M}$ orthogonal ist, wie an dem folgenden einfachen Beispiel zu sehen ist: : Einteilungsganzes sei die Menge $M := \{a, \ldots, z\}.\mathbf{M_1} := \{\{a, b, c, d\}, \{e, f, g, h\}\}, \mathbf{M_2} := \{\{a, i, e, j\}, \{g, c, k, l\}\}, \mathbf{M_3} := \{\{b, i, f, k\}, \{h, l, j, d\}\}$ sind scharfe, unvollständige Einteilungen von M. $\{\mathbf{M_1}, \mathbf{M_2}\}, \{\mathbf{M_1}, \mathbf{M_3}\}$ und $\{\mathbf{M_2}, \mathbf{M_2}\}$ sind orthogonal, jedoch nicht $\{\mathbf{M_1}, \mathbf{M_2}, \mathbf{M_3}\}$, denn jeder Durchschnitt von drei Einteilungsklassen, die aus den verschiedenen Einteilungen stammen ist leer.

Mithilfe der bis hierher in diesem Abschnitt erarbeiteten Begriffe ist es nun möglich eine Definition von Klassifikationssystemen anzugeben. Klassifikationssysteme werden rekursiv auf der Basis des Einteilungsbegriffs definiert. Diese Klassifikationssysteme sind Mengen mit der Struktur von UND-ODER-Bäumen, die sich von den UND-ODER-Bäumen gemäß Definition 7 darin unterscheiden, daß zum einen reine ODER-Knoten vorkommen dürfen und zum anderen, daß für die Rekursionsbasis nicht die leere Menge verwendet wird, sondern Einteilungen bzw. erfüllbare Mengen von Einteilungen.

Auf die ODER-Knoten könnte man ohne weiteres verzichten, da sie zu Spezialfällen von UND-Knoten isomorph sind. Wie aus Definition 10 leicht zu ersehen ist, erhält man aus einem Baum, der ODER-Knoten enthält, einen Baum, der nur UND-ODER-Knoten enthält, indem man jeden Knoten der Form $oder(M, \mathbf{M})$ durch einen Knoten der Form $und(M, \{\mathbf{M}\})$ ersetzt.

Der zweite Unterschied entspricht der Intuition, daß eine Einteilung $\mathbf{M}$ bzw. eine erfüllbare Menge $\{M_1, \ldots, M_n\}$ von Einteilungen einer Menge M ein *minimales* oder *elementares* oder *atomares* Klassifikationssystem von M sein soll. Die Notation hierfür ist $\{oder(M, \mathbf{M})\}$ bzw. $\{und(M, \{M_1, \ldots, M_n\})\}$. Elementare Klassifikationssysteme sind also Einermengen von ODER- bzw.

UND-Knoten. Ausgehend von diesen elementaren Klassifikationssystemen erhält man rekursiv komplexere Klassifikationssysteme, indem man Klassifikationssysteme von $M_i \in \mathbf{M}$ bzw. $M_i \in \mathbf{M_i} \in \{M_1, \ldots, M_n\}$ vereinigt mit Klassifikationssystemen von M. Im folgenden bezeichne ich die Elemente von Klassifikationssystemen als *Knoten*, *ODER-Knoten* bzw. *UND-Knoten*.

Damit sichergestellt werden kann, daß die so erhaltenen Klassifikationssysteme keine Widersprüche enthalten, wird zu jedem Klassifikationssystem eine Basiseinteilung (ebenfalls rekursiv) definiert. Diese ist im Falle des elementaren Klassifikationssystems $\{oder(M, \mathbf{M})\}$ die Einteilung $\mathbf{M}$ bzw. im Falle des elementaren Klassifikationssystems $\{und(M, \{M_1, \ldots, M_n\})\}$ der *Schnitt* $\Cap\{M_1, \ldots, M_n\}$. Letzterer ist eine Einteilung von M, da $\{M_1, \ldots, M_n\}$ nach Voraussetzung erfüllbar ist. Ein elementares Klassifikationssystem kann nur dann mit einem weiteren Klassifikationssystem vereinigt werden, wenn das Ergebnis dieser Vereinigung wiederum eine Basiseinteilung hat.

Definition 10 (Klassifikationssystem) Sei M eine Menge.

1. Ist $\mathbf{M}$ eine Einteilung von M, dann:
   (a) $KS = \{oder(M, \mathbf{M})\}$ ist ein *(elementares) Klassifikationssystem* von M
   (b) $\mathbf{M}$ ist die *Basiseinteilung* von KS
2. Ist $\{\mathbf{M_1}, \ldots, \mathbf{M_n}\}$ eine erfüllbare Menge von Einteilungen von M, dann:
   (a) $KS = \{und(M, \{\mathbf{M_1}, \ldots, \mathbf{M_n}\})\}$ ist ein *(elementares) Klassifikationssystem* von M
   (b) $\Cap\{\mathbf{M_1}, \ldots, \mathbf{M_n}\}$ ist die *Basiseinteilung* von KS
3. Ist $\{oder(M, \mathbf{M})\}$ ein elementares Klassifikationssystem von M und ist KS_i ein Klassifikationssystem von $M_i \in \mathbf{M}$ mit Basiseinteilung $\mathbf{B_i}$, so daß $\mathbf{B_i} \cup \mathbf{M} \backslash \{M_i\}$ eine Einteilung von M ist, dann:
   (a) $KS = \{oder(M, \mathbf{M})\} \cup KS_i$ ist ein *(komplexes) Klassifikationssystem* von M
   (b) $\mathbf{B_i} \cup \mathbf{M} \backslash \{M_i\}$ ist die *Basiseinteilung* von KS
4. Ist $\{und(M, \{\mathbf{M_1}, ..., \mathbf{M_i}, ..., \mathbf{M_n}\})\}$ ein elementares Klassifikationssystem von M und ist KS_i ein ein Klassifikationssystem für $M_i \in \mathbf{M_i}$ mit Basiseinteilung $\mathbf{B_i}$, so daß $\Cap\{\mathbf{M_1}, ..., \mathbf{B_i} \cup \mathbf{M_i} \backslash \{M_i\}, ..., \mathbf{M_n}\}$ eine Einteilung von M ist, dann:
   (a) $KS = \{und(M, \{\mathbf{M_1}, ..., \mathbf{M_i}, ..., \mathbf{M_n}\})\} \cup KS_i$ ist ein *(komplexes) Klassifikationssystem* von M

(b) $\Cap\{\mathbf{M_1}, ..., \mathbf{B_i} \cup \mathbf{M_i} \backslash \{M_i\}, ..., \mathbf{M_n}\}$ ist die *Basiseinteilung* von KS.

5. Nur nach 1. - 4. gebildete Strukturen sind Klassifikationssysteme bzw. Basiseinteilungen.

[28] Ersetzt man ODER-Knoten in der oben beschriebene Weise durch UND-ODER-Knoten, kann man auf die Prädikate *oder/2* und *und/2* verzichten und Knoten als geordnete Paare der Form $(M, \mathbf{M})$ darstellen.

Analog zu den Klassifikationssystemen werden rekursiv die zu diesen gehörenden Klassifikatoren definiert, denn mit der Definition von Klassifikationssystemen allein ist noch nicht klar, was es heißt mithilfe eines solchen Klassifikationssystems zu klassifizieren. Intuitiv erwartet man etwa folgendes: Sei K Klassifikationssystem von M. Dann gibt es zu jedem $m \in M$ eine *Stelle* in K, an der m untergebracht werden kann. Diese Stellen im Klassifikationssystem sind gerade die Klassifikatoren. Ein Klassifikator ist im Falle der hier definierten Klassifikationssysteme, also im Falle von UND-ODER-Bäumen, ein Wurzelpfad, der an Blättern[29] endet. Ein Element $m \in M$ wird klassifiziert, indem das Klassifikationssystem von der Wurzel beginnend durchlaufen wird. Die Knoten im Baum, die hierbei durchlaufen werden, definieren den Klassifikator, der m zugeordnet ist. Ein Wurzelpfad durch einen UND-ODER-Baum ist im allgemeinen ein UND-Baum.

---

[28] Klassifikationssysteme können auch direkt als UND-ODER-Bäume definiert werden. Man muß nur Definition 10.1.a - 10.4.a ersetzen durch 1'.a - 4'.a.

Sei M eine Menge.

1'.a Ist $\mathbf{M}$ eine Einteilung von M, dann ist $oder(M, \mathbf{M})$ ein *(elementares) Klassifikationssystem* von M

2'.a Ist $\{\mathbf{M_1}, \ldots, \mathbf{M_n}\}$ eine erfüllbare Menge von Einteilungen von M, dann ist$und(M, \{\mathbf{M_1}, \ldots, \mathbf{M_n}\})$ ein *(elementares) Klassifikationssystem* von M

3'.a Ist $oder(M, \mathbf{M})$ ein elementares Klassifikationssystem von M und ist KS_i ein Klassifikationssystem von $M_i \in \mathbf{M}$ mit der Basiseinteilung $\mathbf{B_i}$, so daß $\mathbf{B_i} \cup \mathbf{M} \backslash \{M_i\}$ eine Einteilung von M ist, dann ist $oder(M, KS_i \cup \mathbf{M} \backslash \{M_i\})$ ein *(komplexes) Klassifikationssystem* von M

4'.a Ist $und(M, \{\mathbf{M_1}, ..., \mathbf{M_i}, ..., \mathbf{M_n}\})$ ein elementares Klassifikationssystem von M und ist KS_i ein Klassifikationssystem von $M_i \in \mathbf{M_i}$ mit der Basis $\mathbf{B}$, so daß $\Cap\{\mathbf{M_1}, ..., \mathbf{B} \cup \mathbf{M_i} \backslash \{M_i\}, ..., \mathbf{M_n}\}$ eine Einteilung von M ist, dann ist $und(M, \{\mathbf{M_1}, ..., KS_i \cup \mathbf{M_i} \backslash \{M_i\}, ..., \mathbf{M_n}\})$ ein *(komplexes) Klassifikationssystem* von M

5 Nur nach 1 - 4. gebildete Strukturen sind Klassifikationssysteme bzw. Basiseinteilungen.

[29] Die Verwendung des Plurals ist an dieser Stelle berechtigt, denn wenn ein Pfad über einen UND-Knoten läuft, teilt er sich auf und endet also notwendigerweise an mehreren Blättern.

Auch bei den Klassifikatoren ist wie bei den Klassifikationssystemen sicherzustellen, daß sie keinen Widerspruch enthalten (andernfalls würden sie nichts klassifizieren). Darum wird zu jedem Klassifikator eine Basismenge (wiederum rekursiv) definiert. Die Basismenge eines Klassifikators ist nichts anderes als die Menge der Objekte, welche durch den Klassifikator klassifiziert werden. Diese Menge darf nicht leer sein.

Definition 11 (Klassifikator) Sei M eine Menge.

1. Ist $KS = \{oder(M, \mathbf{M})\}$ ein Klassifikationssystem von M und ist $M_i \in \mathbf{M}$, dann:
   (a) $K = (M, M_i)$ ist ein Klassifikator von KS
   (b) $\mathbf{M_i}$ ist die *Basismenge* von K
2. Ist $KS = \{und(M, \{\mathbf{M_1}, \ldots, \mathbf{M_n}\})\}$ ein Klassifikationssystem von M und sind $M_1 \in \mathbf{M_1}$ und … und $M_n \in \mathbf{M_n}$, so daß $\bigcap\{M_1, \ldots, M_n\} \in \Cap\{\mathbf{M_1}, \ldots, \mathbf{M_n}\}$, dann:
   (a) $K = (M, \{M_1, \ldots, M_n\})$ ist ein *Klassifikator* von KS
   (b) $\bigcap\{M_1, \ldots, M_n\}$ ist die *Basismenge* von K
3. Ist $\{oder(M, \mathbf{M})\} \cup KS_i$ ein Klassifikationssystem von M mit Basiseinteilung $\mathbf{B}$ und ist K_i Klassifikator von KS_i mit Basismenge B_i, so daß $B_i \in \mathbf{B}$, dann:
   (a) $K = (M, K_i)$ ist Klassifikator von M
   (b) B_i ist die Basismenge von K
4. Ist $\{und(M, \{\mathbf{M_1}, ..., \mathbf{M_i}, ..., \mathbf{M_n}\})\} \cup KS_i$ ein Klassifikationssystem von M und ist K_i Klassifikator von KS_i und ist B_i die Basismenge von K_i und sind $M_1 \in \mathbf{M_1}$ und … und $M_i \in \mathbf{M_i}$ und … $M_n \in \mathbf{M_n}$, so daß $\bigcap\{M_1, ..., B_i, ..., M_n\} \in \mathbf{B}$, dann:
   (a) $K = (M, \{M_1, ..., K_i, ...M_n\}$ ist *Klassifikator* von M
   (b) $\bigcap\{M_1, ..., B_i, ..., M_n\}$ ist *Basismenge* von K
5. Nur nach 1. - 4. gebildete Strukturen sind Klassifikatoren bzw. Basismengen.

Ein Klassifikator eines Klassifikationssystems KS ist nach Definition 11 *kein* Teilbaum von KS. Zwar könnte die Definition ohne weiteres so geändert werden, daß dies der Fall ist, aber dadurch würden die Klassifikatoren mit unnötigem syntaktischen Ballast befrachtet.

Schließlich wird ebenfalls parallel zu den Klassifikationssystemen definiert, was eine Anfrage an ein Klassifikationssystem ist und wie die zugehörige Antwortmenge definiert ist. Die Intuition hierbei ist, daß Anfragen Klassifikatoren sind oder aus Klassifikatoren hervorgehen, indem man Teilbäume dieser Klassifikatoren (welche selbst wieder Klassifikatoren sind) wegläßt. Die Antwortmengen ergeben sich aus den Basismengen der Klassifikatoren bzw. rekursiv aus Antwortmengen der Anfragen, welche den weggelassenen Klassifikatoren zugeordnet sind. Im Falle eines elementaren Klassifikationssystems sind Anfragen bzw. Antwortmengen identisch mit den Klassifikatoren bzw. Basismengen.

Definition 12 (Anfrage) Sei M eine Menge.

1. Ist KS ein Klassifikationssystem von M und ist K ein Klassifikator von KS mit Antwortmenge M_i, dann:
   (a) K ist eine Anfrage an KS
   (b) M_i ist die Antwortmenge auf K
2. Ist $KS = \{oder(M, \mathbf{M})\} \cup KS_i$ ein Klassifikationssystem von M, dann:
   (a) Ist A eine Anfrage an $\{oder(M, \mathbf{M})\}$ mit Antwortmenge A_i, dann:
      i. A ist eine Anfrage an KS
      ii. A_i ist die Antwortmenge auf A
   (b) Ist A eine Anfrage an KS_i mit Antwortmenge A_i, dann:
      i. (M, A) ist eine Anfrage an KS
      ii. A_i ist die Antwortmenge auf A
3. Ist $KS = \{und(M, \{\mathbf{M_1}, ..., \mathbf{M_i}, ..., \mathbf{M_n}\})\} \cup KS_i$ ein Klassifikationssystem von M, dann:
   (a) Ist A eine Anfrage an $\{und(M, \{\mathbf{M_1}, \ldots, \mathbf{M_n}\})\}$ mit Antwortmenge A_i, dann:
      i. A ist eine Anfrage an KS
      ii. A_i ist die Antwortmenge auf A
   (b) Ist A eine Anfrage an KS_i mit Antwortmenge A_i und ist $(M, \{M_1, ..., M_i, ..., M_n\})$ eine Anfrage an $\{und(M, \{\mathbf{M_1}, \ldots, \mathbf{M_n}\})\}$ mit Antwortmenge A_i, dann:
      i. $(M, \{M_1, ..., A, ..., M_n\})$ ist eine Anfrage an KS
      ii. $A \cap A_i$ ist Antwortmenge auf $(M, \{M_1, ..., A, ..., M_n\})$
4. Nur nach 1. - 3. gebildete Strukturen Anfragen bzw. Antwortmengen.

Mithilfe des Begriffs des Klassifikators eines Klassifikationssystems ist es möglich, die Begriffe *scharfes, unscharfes, vollständiges, unvollständiges, korrektes, unkorrektes Klassifikationssystem* zu definieren:

Definition 13 Sei KS ein Klassifikationssystem von M.

1. KS ist *scharfes* Klassifikationssystem von M genau dann, wenn es zu jedem $m \in M$ höchstens einen Klassifikator K von KS gibt, so daß m in der Basismenge von K enthalten ist. Andernfalls heißt KS unscharfes Klassifikationssystem von M.

2. KS ist *vollständiges* Klassifikationssystem von M genau dann, wenn es zu jedem $m \in M$ mindestens einen Klassifikator K von KS gibt, so daß m in der Basismenge von K enthalten ist. Andernfalls heißt KS unvollständiges Klassifikationssystem von M.

3. KS ist *korrektes* Klassifikationssystem von M genau dann, wenn KS scharfes und vollständiges Klassifikationssystem von M ist. Andernfalls heißt KS unkorrektes Klassifikationssystem von M.

Wer sich nun allgemein mit der Struktur von Sprachbeschreibungen beschäftigt, dem geht es vor allem darum, ein Klassifikationssystem für die Menge der linguistischen Phänomene zu finden, welches (möglichst) korrekt ist. Das heißt, daß *jedes* linguistische Phänomen in diesem System an einer *eindeutig bestimmten* Stelle untergebracht werden kann. Jedes tatsächlich formulierte Klassifikationssystem für linguistische Phänomene wird sich diesem Ziel nur bis zu einem gewissen Grad annähern können. Dies liegt daran, daß die Klasse der linguistischen Phänomene nicht exakt definiert werden kann, wie dies beispielsweise für mathematische Gegenstände wie die natürlichen Zahlen möglich ist. Wollte man dies erreichen, so müßte man eine Menge von mathematischen Strukturen definieren, welche genau die Menge der linguistischen Phänomene modelliert. Die in der Linguistik bislang erarbeiteten theoretischen Ansätze konntem einem solchen Anspruch bisher nicht gerecht werden. Seriöse Ansätze beschränken sich daher explizit auf einen Teilbereich der linguistischen Phänomene, in der Regel die Syntax. Für Sprachbeschreibungen ist aber eine möglichst vollständige Beschreibung der in der Sprache auftretenden linguistischen Phänomene einer durch die Einhaltung von formalen Standards zur Unvollständigkeit gezwungenen Beschreibung der linguistischen Phänomene vorzuziehen. Es gibt keine präzisen Kriterien, um in allen Fällen linguistische Phänomene von nicht-linguistischen Phänomenen unterscheiden zu können. Deshalb ist jedes Klassifikationssystem für linguistische Phänomene vorläufig und bei gegebenem Anlaß zu ändern.

4 Ein Klassifikationssystem für Sprachbeschreibungen

Sprachbeschreibungen setzten ein Klassifikationssystem für die Menge P der linguistischen Phänomene voraus. Was darüber hinaus noch dazu kommt, ist nicht viel.

Zum einen ist dies die Menge L der *Sprachen*[30]. Wie bereits auf Seite 7 erwähnt wurde, sind Sprachen definierbar durch die Menge der sie konstituierenden linguistischen Phänomene.

Des weiteren kommt die Menge der *Sprachbeschreiber* S hinzu. Diese müssen Erwähnung finden, da ein und dieselbe Sprache von verschiedenen Forschern beschrieben werden kann und es von Interesse ist, konkurrierende Sprachbeschreibungen verfügbar zu haben. Die Sprachbeschreiber sind sowohl von den Sprachen als auch den linguistischen Phänomenen (logisch) unabhängig.

Schließlich kommt noch die Menge B der eigentlichen *Beschreibungen* der linguistischen Phänomene hinzu. Idealerweise sollte zwar durch die Klassifikation eines linguistischen Phänomens *alles* über dieses Phänomen gesagt sein, aber ein Klassifikationssystem für linguistische Phänomene, welches diese Eigenschaft hat, ist noch nicht gefunden. Die Klassifikation eines linguistischen Phänomens wird in der Regel noch einen *Beschreibungsbedarf* hinterlassen. Es ist also noch eine Menge von Beschreibungen B zu berücksichtigen. B ist *nicht* unabhängig von der Menge der durch Klassifikatoren erfaßten linguistischen Phänomene, sondern ganz im Gegenteil höchst abhängig. Denn durch die Klassifikation wird ein linguistisches Phänomen ja ebenfalls (zumindest partiell) beschrieben. Eine darüber hinausgehende Beschreibung sollte mit der durch die Klassifikation geleisteten Beschreibung *inhaltlich* im Einklang stehen. Dies bedeutet, daß durch die Beschreibung eine nicht-leere echte Teilmenge der Basismenge des Klassifikators ausgezeichnet[31] wird. Allerdings gibt es *keine* Methode, um dies in jedem Fall zu gewährleisten. Dazu müßten die Beschreibungen in formaler Sprache erstellt werden und dies ist nach dem oben auf Seite 30 gesagten gerade *nicht* in jedem Fall möglich. Es ist allerdings schon möglich, die Phänomenbeschreibung dahingehend zu *unterstützen*, daß die durch die Klassifikation geleistete Beschreibung als Grundlage für jede weitere Beschreibung in diese übernommen wird. Dies bedeutet, daß die Klassifikatoren als Grundlage dienen, um einen Beschreibungsrahmen oder Beschreibungsschemata oder Beschreibungsprämissen zu definieren, wel-

---

[30] Ob dies natürliche oder künstliche, tote oder lebende Sprachen sind, spielt keine Rolle.

[31] Den Terminus *definieren* möchte ich an dieser Stelle nicht verwenden, da Klassifikatoren und Beschreibungen nicht Sätze von ein und derselben (Meta-) Sprache sind.

che den Sprachbeschreibern angeboten werden. Da das Ausfüllen dieses Rahmens jedoch im allgemeinen *nicht* auf rein formaler Basis stattfinden kann, kann auch nicht sichergestellt werden, daß durch die erhaltene Beschreibung (das Ergebis des Ausfüllens) eine nicht-leere echte Teilmenge der Basismenge des Klassifikators ausgezeichnet wird.

Die (im allgemeinen) informelle Natur der Beschreibungen hat für eine Datenbank, in welche diese Beschreibungen aufgenommen werden, noch eine weitere wichtige Konsequenz: Die Beschreibungen sind (im allgemeinen) nur als Ganzes abrufbar. Es ist (im allgemeinen) nicht möglich eine Anfrage an die Datenbank zu stellen dergestalt, daß zu einem gegebenen Klassifikator genau die Beschreibungen ausgegeben werden, die aus der Basismenge des Klassifikators eine bestimmte Teilmenge aussondern. Der Anfragebegriff von Definition 12 kann dies nicht leisten.

Sei $\mathbf{L}$ die Menge der Sprachen, $\mathbf{S}$ die Menge der Sprachbeschreiber, $\mathbf{KS}$ ein Klassifikationssystem für linguistische Phänomene und $\mathbf{K}$ die Menge der Klassifikatoren von $\mathbf{KS}$, sei schließlich $\mathbf{B}$ die Menge der Beschreibungen von linguistischen Phänomenen.

Eine Sprachbeschreibung ist eine Teilmenge von $\mathbf{L} \times \mathbf{S} \times \mathbf{K} \times \mathbf{B}$

Damit ist die Struktur einer Datenbank für Sprachbeschreibungen festgelegt. Diese besteht aus einer einzigen vierstelligen Relation. Jedes Element dieser Relation ist ein Datenbankeintrag.

Datenbankeinträge können aber auch anders modelliert werden:

Die Menge der potentiellen Datenbankeinträge **DBE** wird definiert durch die Funktionen

- SPRACHE: $\mathbf{DBE} \longrightarrow \mathbf{L}$
- BESCHREIBER: $\mathbf{DBE} \longrightarrow \mathbf{S}$
- KLASSIFIKATOR: $DBE \longrightarrow \mathbf{K}$
- BESCHREIBUNG: $\mathbf{DBE} \longrightarrow \mathbf{B}$

Wobei ich voraussetze, daß die Bildbereiche dieser Funktionen endlich sind.[32] Da jede Bildmenge endlich ist, sind die Mengen der Einermengen von Elementen in den Bildmengen korrekte Einteilungen der Bildmengen.

---

[32] Diese Annahme ist meines Erachtens von einem praktischen Standpunkt aus gerechtfertigt, da die Menge der Datenbankeinträge in der Praxis immer endlich ist. Andererseits räume ich gerne ein, daß diese Annahme von einem rein formalen Standpunkt aus sehr in Zweifel gezogen werden kann. Aber da Sprachbeschreibung nicht rein formal abgehandelt werden kann, ist man gezwungen immer wieder solche lediglich plausiblen oder auch nur nützlichen Annahmen zu machen.

- **Sprache** $:= \{E \subseteq \textsc{sprache}(\mathbf{DBE}) \mid |E| = 1\}$ ist korrekte Einteilung von **DBE**.
- **Beschreiber** $:= \{E \subseteq \textsc{beschreiber}(\mathbf{DBE}) \mid |E| = 1\}$ ist korrekte Einteilung von **DBE**.
- **Klassifikator** $:= \{E \subseteq \textsc{klassifikator}(\mathbf{DBE}) \mid |E| = 1\}$ ist korrekte Einteilung von **DBE**.
- **Beschreibung** $:= \{E \subseteq \textsc{beschreibung}(\mathbf{DBE}) \mid |E| = 1\}$ ist korrekte Einteilung von **DBE**.

Damit ist

$$\mathrm{SBKS} =$$

$$und(\mathbf{DBE}, \{\mathbf{Sprache}, \mathbf{Beschreiber}, \mathbf{Klassifikator}, \mathbf{Beschreibung}\})$$

ein Klassifikationssystem von **DBE**, genannt Sprachbeschreibungsklassifikationssystem, abgekürzt **SBKS**. Die Menge der *tatsächlichen* Datenbankeinträge ist somit eine Teilmenge der Menge der Klassifikatoren dieses Klassifikationssystems von **DBE**.

5 Eine deduktive Datenbank für Sprachbeschreibungen

Ein Datenbanksystem besteht aus einer Datenbank und einem Datenbankmanagementsystem. Dies trifft auch auf ein *deduktives* Datenbanksystem zu. In der Datenbank werden zweierlei Arten von Daten abgelegt:

1. Das Klassifikationssystem **SBKS**.
2. Klassifikatoren des Klassifikationssystems **SBKS**.

Das Datenbankmanagementsystem ist ein Programmsystem, das die Datenbank verwaltet, fortschreibt und alle Zugriffe auf die Datenbank regelt. Es muß daher die folgenden Möglichkeiten bieten:

1. Erstellung und Modifikation von Klassifikationssystemen.
2. Eingabe, Ausgabe von Klassifikatoren.
3. Stellen von Anfragen an die in der Datenbank hinterlegten Klassifikatoren und Ausgabe der Antworten.

5.1 Darstellung von Klassifikationssystemen in Prolog

Wie aus Definition 10 hervorgeht, hat ein Klassifikationssystem die Struktur eines UND-ODER-Baums. Für die Implementation wird dieser UND-ODER-Baum als *Menge* von UND- und ODER-Knoten dargestellt.[33] Die in einem Klassifikationssystem vorkommenden Mengen werden durch Namen ersetzt.

KS bezeichne ein vorgegebenes Klassifikationssystem. Ein Knoten aus **KS** der Gestalt $oder(M, \{M_1, \ldots, M_n\})$ wird zu einer Prolog-Klausel der Gestalt $node(oder, N, [N_1, ..., N_n])$, wobei N ein *Name*[34] für die Einteilung $\{M_1, \ldots, M_n\}$ ist und für $i \in \{1, \ldots, n\}$ gilt:

- N_i ist ein Name für eine Einteilung von M_i, falls es in **KS** einen Knoten $oder(M_i, \mathbf{M_i})$ gibt
- N_i ist ein Name für eine *Menge* von Einteilungen von M_i, falls es in **KS** einen Knoten $und(M_i, \{\mathbf{M_{i_1}}, \ldots, \mathbf{M_{i_k}}\})$ gibt
- N_i ist ein Name für eine Einteilungsklasse, falls es *keinen* Knoten von M_i in **KS** gibt.

Ein Knoten aus **KS** der Gestalt $und(M, \{\mathbf{M_1}, \ldots, \mathbf{M_n}\})$ wird zu einer Prolog-Klausel der Gestalt $node(and, N, [N_1, \ldots, N_n])$, wobei N hier Name für eine Menge von Einteilungen ist und für $i \in \{1, \ldots, n\}$ gilt:

- N_i ist ein Name für eine Einteilung von M_i, falls es in **KS** einen Knoten $oder(M_i, \mathbf{M_i})$ gibt
- N_i ist ein Name für eine *Menge* von Einteilungen von M_i, falls es in **KS** einen Knoten $und(M_i, \{\mathbf{M_{i_1}}, \ldots, \mathbf{M_{i_l}}\})$ gibt

N_i kann in diesem Fall kein Name einer Einteilungsklasse sein.

Das tatsächlich verwendete Prolog-Prädikat `node` hat noch eine weitere Argumentstelle, die ich *Message*. Das Argument *Message* soll Definitionen, Erklärungen oder sonstwie angebracht erscheinende Mitteilungen aufnehmen, die am Bildschirm ausgegeben werden, wenn bei der Klassifikation der zugehörige Knoten durchlaufen wird. Dies soll die Klassifikation unterstützen. Die Datenbank enthält also Einträge der Gestalt:

```
node(Typ,Name,Message,Liste),
```

---

[33] Ein UND-Knoten ist ein Knoten der Gestalt $oder(M, \{\mathrm{M_1}, \ldots, \mathrm{M_n}\})$, ein ODER-Knoten ist ein Knoten der Gestalt $oder(M, \mathrm{M})$.

[34] Ein Atom in Prolog.

wobei `Typ` den Wert `or` oder den Wert `and` haben kann, `Name` der Name des Einteilungsganzen ist, `Message` wie gerade besprochen zu interpretieren ist, und `Liste` im Fall, daß `Typ` den Wert `or` hat, eine *Liste von Namen von Einteilungsklassen* ist, und im Fall, daß `Typ` den Wert `and` hat, eine *Liste von Namen von Einteilungen* ist.

Eine wichtige Integritätsbedingung lautet, daß es zu jedem Namen $E \in$ `Liste` genau eine weitere Klausel geben muß, deren `Name` E ist. Eine Verletzung dieser Integritätsbedingung bedeutet, daß nicht mehr jeder Knoten von der Wurzel aus erreichbar ist. Und damit bildet die Menge der Knoten nicht mehr *ein* Klassifikationssystem.

Im folgenden nenne ich diese Einträge auch *Knoten* und, da jeder Knoten anhand seines Namens eindeutig identifizierbar ist, beziehe ich mich manchmal mit dem Namen des Knotens auf den ganzen Eintrag. Was jeweils gemeint ist, wird aus dem Kontext hervorgehen. Ich nenne jedes Element in der Liste *Nachfolger* des Knotens. Einen Knoten vom Typ `or` bzw. `and` nenne ich auch *UND-Knoten* bzw. *ODER-Knoten*. Desweiteren werde ich manchmal davon sprechen, daß ein Knoten *angelaufen* wird. Damit ist eine Situation beim Baumdurchlauf gemeint, daß der Knoten zwar expandiert *wird*, dies aber noch nicht geschehen ist.

Bevor ich ein (kleines) Klassifikationssystem als Beispiel gebe, möchte ich auf ein praktisches und gleichzeitig grundsätzliches Problem hinweisen, das sich bei der Erstellung von Klassifikationssystemen stellt. Ich möchte dies an den folgenden Beispielen erläutern. Dieses Problem erzwingt vom Ersteller eines Klassifikationssystems manchmal eine gewisse sprachschöpferische Fähigkeit. Es wird sich zeigen, daß die Argumentstelle `Message` sehr nützlich sein kann, um dieses Problem zu entschärfen.

Zunächst ein unproblematisches Beispiel.

Beispiel 3 Einteilungsganzes sei die Menge M der Menschen. Zur Definition einer Einteilung wird das Merkmal ALTER: $M \longrightarrow \mathbb{N}$ benutzt. Nach Theorem 2 gibt es zu jeder Einteilung $\mathbf{Q}$ von ALTER(M) eine induzierte Einteilung $\mathbf{A}$ von M. $\mathbf{Q} = \{Q_1, Q_2, Q_3\}$ sei gegeben durch

- $x \in Q_1 :\Leftrightarrow x \in \{0, \ldots, 20\}$.
- $x \in Q_2 :\Leftrightarrow x \in \{21, \ldots, 60\}$.
- $x \in Q_3 :\Leftrightarrow x \geq 61$.

$\mathbf{Q}$ ist offensichtlich sogar eine korrekte Einteilung von ALTER(M) (und von $\mathbb{N}$). Die von $\mathbf{Q}$ und ALTER induzierte Einteilung $\mathbf{A}$ ist dann nach Theorem 2 definiert durch $\mathbf{A} = \{A_1, A_2, A_3\}$ mit $\forall i \in \{1,2,3\} : x \in A_i :\Leftrightarrow \textsc{Alter}(x) \in Q_i$. Schreibt man die Definition aus, so ergibt sich

- $x \in A_1 :\Leftrightarrow \textsc{Alter}(x) \in \{0, \ldots, 20\}$.
- $x \in A_2 :\Leftrightarrow \textsc{Alter}(x) \in \{21, \ldots, 60\}$.
- $x \in A_3 :\Leftrightarrow \textsc{Alter}(x) \geq 61$.

Das nächste Beispiel ist ebenfalls unproblematisch, aber es zeigt sehr schön, worin die Eigenschaft besteht, die das nachfolgende problematische Beispiel nicht hat.

Beispiel 4 Einteilungsganzes sei wiederum die Menge M der Menschen. Zur Definition einer Einteilung wird das Merkmal GESCHLECHT: $M \longrightarrow \{\mathit{weiblich}, \mathit{maennlich}\}$ benutzt. Zum Wertebereich des Merkmals GESCHLECHT gibt es genau *eine* Einteilung, welche auf M die folgende Einteilung $\mathbf{G} = \{G_1, G_2\}$ induziert:

- $x \in G_1 :\Leftrightarrow \textsc{Geschlecht}(x) = \mathit{weiblich}$.
- $x \in G_2 :\Leftrightarrow \textsc{Geschlecht}(x) = \mathit{maennlich}$.

Allerdings könnte man auf den Merkmalsnamen GESCHLECHT verzichten, und folgende Einteilung **G'** $= \{G'_1, G'_2\}$ definieren:

- $x \in G'_1 :\Leftrightarrow \mathit{weiblich}(x)$.
- $x \in G'_2 :\Leftrightarrow \mathit{maennlich}(x)$.

Jetzt kommt ein nahezu problematisches Beispiel.

Beispiel 5 Einteilungsganzes sei die Menge M der phonologischen Formen. Dies ist die Menge der Phonemfolgen einer Sprache, welche korrekte Ausdrücke der Sprache sind. Phonologische Formen kann man sprachunabhängig unterteilen in *elementare* phonologische Formen und *komplexe* phonologische Formen. Also ist $\mathbf{H} = \{H_1, H_2\}$ eine Einteilung von M, definiert durch

- $x \in H_1 :\Leftrightarrow \mathit{elementar}(x)$.
- $x \in H_1 :\Leftrightarrow \mathit{komplex}(x)$.

Die Einteilung **H** unterscheidet sich von den Einteilungen **A** und **G** darin, daß es in der deutschen Sprache zu dem Begriffspaar *elementar/komplex* keinen analogen Begriff zu ALTER bzw. GESCHLECHT gibt. Das heißt, es gibt keinen Namen für das Merkmal, dessen Wertebereich die Menge $\{elementar, komplex\}$ ist. Beispiel 4 zeigt, daß der Merkmalsname, falls vorhanden, nicht in jedem Fall in die Definition einer Einteilung eingehen muß. Beispiel 5 wiederum zeigt, daß es Fälle gibt, in denen die Definition einer Einteilung ohne Verwendung des Merkmalsnamens nicht gut möglich ist. Eine naheliegende Erklärung für diese Tatsache scheint darin zu liegen, daß in der deutschen Sprache die Begriffe *weiblich, maennlich* bzw. *komplex, elementar* lexikalisiert sind, nicht hingegen die Begriffe für $\{\textsc{Alter}(x) \in \{0,\ldots,20\}, \textsc{Alter}(x) \in \{21,\ldots,60\}, \textsc{Alter}(x) \geq 61\}$. Trotzdem gibt es bei der Verwendung dieser Begriffe keine Verständnisprobleme. Dies ist bei dem nächsten Beispiel leider nicht mehr der Fall.

Beispiel 6 Es handelt sich um eine angereicherte Version von Beispiel 2.[35] Klassifiziert werden wieder die Kookkurrenzrestriktionen.

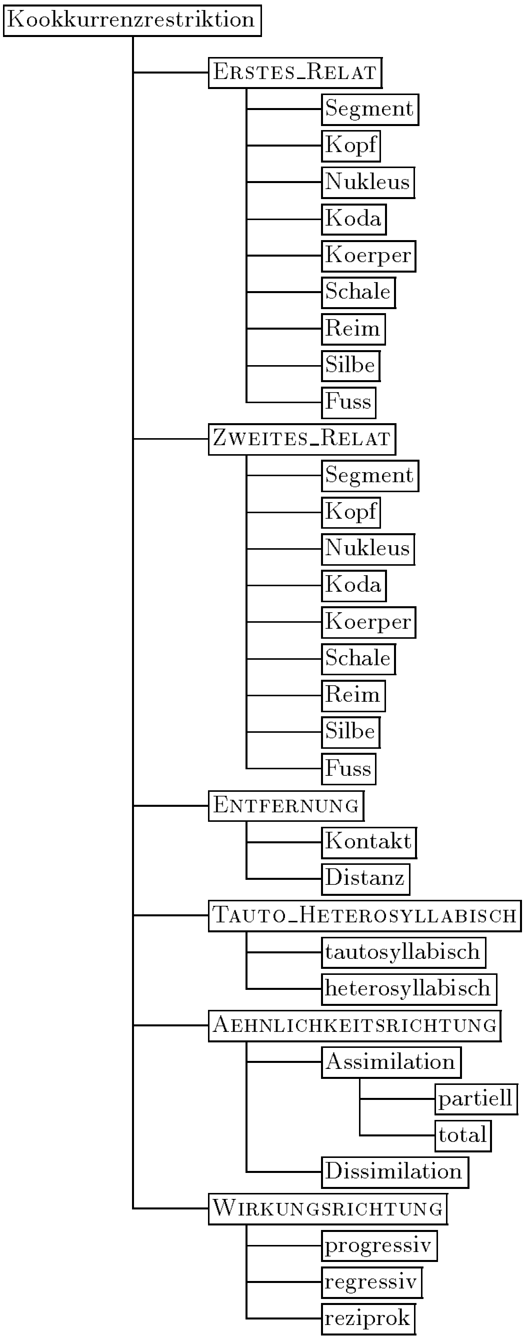
Zur Klassifikation werden die folgenden Knoten verwendet:

1. *und(Kookkurrenzrestriktion, {Erstes_Relat, Zweites_Relat, Entfernung, Tauto_Heterosyllabisch, Aehnlichkeitsrichtung, Wirkungsrichtung})*
2. *oder(Erstes_Relat, {Segment, Kopf, Nukleus, Koda, Koerper, Schale, Reim, Silbe, Fuss})*
   Art des ersten Relats. *Relat* bezeichnet die Konstituente, welche einer Restriktion unterworfen ist.
3. *oder(Zweites_Relat, {Segment, Kopf, Nukleus, Koda, Koerper, Schale, Reim, Silbe, Fuss})*
   Art des zweiten Relats.
4. *oder(Entfernung, {Kontakt, Distanz})*
   Die Kookkurrenzrestriktion besteht zwischen direkt aneinanderstoßenden oder auseinanderliegenden Relaten.
5. *oder(Tauto_Heterosyllabisch, {tautosyllabisch, heterosyllabisch})*
   Die Kookkurrenzrestriktion besteht liegt innerhalb einer Silbe oder geht über Silbengrenzen hinweg.
6. *oder(Aehnlichkeitsrichtung, {Assimilation, Dissimilation})*
   Die Kookkurrenzrestriktion bewirkt, daß die Relate ähnlicher oder unähnlicher werden.
7. *oder(Assimilation, {partiell, total})*
   Die Relate werden ähnlicher, aber nicht gleich (= partiell) oder gleich (= total).

---

[35] vgl. den Baum auf Seite 20.

8. *oder(Wirkungsrichtung, {progressiv, regressiv, reziprok})*
   Das erste Relat wirkt auf das zweite oder umgekehrt oder sie wirken beide aufeinander.



Ein Klassifikationssystem für Kookkurrenzrestriktionen.

Eine Übersetzung dieser Knoten in Prolog ergibt die folgenden Klauseln, die ich der Lesbarkeit halber *nicht* in Prolog-Notation dargestellt habe (außer an einem Beispiel), sondern in tabellarischer Form, aus der man die entsprechende Prolog-Klausel jedoch leicht konstruieren kann.

Typ : and
Name : 'Kookkurrenzrestriktion'
Message : ”[36]
Liste : ['Erstes_Relat', 'Zweites_Relat', 'Entfernung', 'Tauto_Heterosyllabisch', 'Aehnlichkeitsrichtung', 'Wirkungsrichtung']

Als Prolog-Klausel sieht das so aus:

```
node(and,
     ´Kookkurrenzrestriktion´,
     ´´,
     [´Erstes_Relat´,
      ´Zweites_Relat´,
      ´Entfernung´,
      ´Tauto_Heterosyllabisch´,
      ´Aehnlichkeitsrichtung´,
      ´Wirkungsrichtung´]).
```

An dieser Klausel zeigt sich schon die sprachschöpferische Tätigkeit, denn die Bezeichnungen 'Erstes_Relat', 'Zweites_Relat', 'Entfernung', 'Tauto_Heterosyllabisch', 'Aehnlichkeitsrichtung', 'Wirkungsrichtung' sind *ad hoc* geprägte Oberbegriffe für Begriffe, die wie *komplex* und *elementar* eine Einteilung definieren, zu denen jedoch kein Oberbegriff in der Sprache existiert, in der Weise, wie der Begriff *Geschlecht* Oberbegriff von *weiblich* und *maennlich* ist. Da man nicht erwarten kann, daß ad hoc geprägte Begriffe sofort einleuchten, ist es nützlich, wenn nicht notwendig, diesen eine Erklärung beizugeben.

---

[36] ” bezeichnet den leeren String. Für diesen Knoten ist (noch) keine Message eingetragen.

Typ : or
Name : 'Erstes_Relat'
Message : 'Eine Kookkurrenzrestriktion besteht zwischen zwei Konstituenten einer phonologischen Struktur, genannt Relate, wobei das erste Relat links vom zweiten Relat in der phonologischen Struktur angeordnet ist. Von welcher Art ist das erste Relat?'
Liste : ['Segment', 'Kopf', 'Nukleus', 'Koda', 'Koerper', 'Schale', 'Reim', 'Silbe', 'Fuss']

Die Begriffe 'Segment', 'Kopf', 'Nukleus', 'Koda', 'Koerper', 'Schale','Reim', 'Silbe', 'Fuss' finden sich alle in linguistischen Lexika, nicht jedoch der Begriff 'Relat'. Ein analoges Argument gilt für alle folgenden Klauseln.

Typ : or
Name : 'Zweites_Relat'
Message : 'Eine Kookkurrenzrestriktion besteht zwischen zwei Konstituenten einer phonologischen Struktur, genannt Relate, wobei das erste Relat links vom zweiten Relat in der phonologischen Struktur angeordnet ist. Von welcher Art ist das zweite Relat?'
Liste : ['Segment', 'Kopf', 'Nukleus', 'Koda', 'Koerper', 'Schale', 'Reim', 'Silbe', 'Fuss']

Typ : or
Name : 'Entfernung'
Message : 'Folgen die Relate direkt aufeinander (= Kontakt) oder nicht (= Distanz)?'
Liste : ['Kontakt', 'Distanz']

Typ : or
Name : 'Tauto_Heterosyllabisch'
Message : 'Liegen die Relate innerhalb einer Silbe oder in verschiedenen Silben?
Liste : ['tautosyllabisch', 'heterosyllabisch']

---

[36] <Wahl> steht für das aus **Liste** gewählte Element. Als Eintrag im Knoten erscheint <Wahl> nicht, sondern wird beim Baumdurchlauf eingefügt.

Typ : or
Name : Aehnlichkeitsrichtung
Message : 'Bewirkt die Kookkurrenzrestriktion bei den Relaten Assimilation oder Dissimilation?'
Liste : ['Assimilation', 'Dissimilation']

Typ : or
Name : 'Assimilation'
Message : 'Ist die Assimilation partiell oder total?'
Liste : [partiell, total]

Typ : or
Name : 'Wirkungsrichtung'
Message : 'Wirkt das erste Relat auf das zweite? (= progressiv) Oder umgekehrt? (= regressiv) Oder wirken beide aufeinander? (= reziprok)
Liste : [progressiv, regressiv, reziprok]

Für jeden Knoten, dem nicht wieder eine Einteilung zugeordnet ist, ist in der Datenbank ein Eintrag vorhanden, bei dem die Argumentstelle für `Liste` die leere Liste enthält. Zum Beispiel

```
node(or,total ,´´,[]).
```

In früheren Versionen des Programms waren solche *Blattknoten* nicht vorgesehen, aber mit der Einführung des Arguments `Message` schien es mir sinnvoll, diese Information für *jede* Einteilungsklasse zur Verfügung zu haben.

5.1.1 Erstellung eines Klassifikators in Prolog

Beim Durchlaufen eines Klassifikationssystems beginnend bei der Wurzel, die ausgezeichnet ist durch das Prädikat `root(Wurzel)`, wird ein Klassifikator aufgebaut.

Die Grundstruktur des Algorithmus zur Erstellung eines Klassifikators ist sehr einfach. Für die Darstellung dieser Grundstruktur habe ich auf das Argument `Message` verzichtet. Ein Knoten hat also die Struktur `node(Typ,Name,Liste)`, wobei die Argumente die auf Seite 34 angegebenen Bedeutungen haben. Ein Klassifikationssystem wird beginnend bei der Wurzel *depth first* durchlaufen. Bei jedem ODER-Knoten wird ein Nachfolger vom Klassifizierer ausgewählt. Ein Ast ist beendet, wenn ein Nachfolger gewählt wurde, der selbst keine Nachfolger mehr hat. Bei jedem UND-Knoten werden alle Nachfolger in der Reihenfolge in der sie in der Liste `SubCatList` stehen durchlaufen.

```
build_classifier :-
        root(Node),
        traverse_node(Node:Category).

%%% traverse_node(+Node:-Category)
%%% Die Struktur Node:Category ist ein Klassifikator
traverse_node(Node:Category) :-
        node(Type,Node,SubCatList),
        (Type == or ->
             get_choice(Node,SubCatList,SubCat),
             (node(or,SubCat,[]) ->
                  Category = SubCat,
             ;
             Category = [SubCat:Cat],
             traverse_node(SubCat:Cat))
        ;
        Type == and,
        traverse_nodes(SubCatList,Category))

%%% traverse_nodes(+SubCatList,-PathList)
traverse_nodes([],[]).
traverse_nodes([SubCat|SubCatList],[SubCat:Cat|Cats]) :-
        traverse_node(SubCat:Cat),
        traverse_nodes(SubCatList,Cats).
```

Ein Beispiel für einen Klassifikator, der zu dem Klassifikationssystem von Beispiel 6 gehört, ist der folgende:

```
´Kookkurrenzrestriktion´ :
     [´Erstes_Relat´ : ´Kopf´,
      ´Zweites_Relat´ : ´Koda´,
      ´Entfernung´ : ´Kontakt´,
      ´Tauto_Heterosyllabisch´ : heterosyllabisch,
      ´Aehnlichkeitsrichtung´ :
           [´Assimilation´ : partiell],
      ´Wirkungsrichtung´ : regressiv]
```

Die Einteilungen ’Erstes_Relat’, ’Zweites_Relat’, ’Entfernung’, ’Tauto_Heterosyllabisch’, ’Aehnlichkeitsrichtung’, ’Wirkungsrichtung’ bilden *keine* orthogonale Menge. Dies zeigt sich, wenn man die Einteilungen ’Erstes_Relat’, ’Zweites_Relat’ und ’Tauto_Heterosyllabisch’ betrachtet. Zwar ist für ’Erstes_Relat’ und ’Zweites_Relat’ jede Kombination von Einteilungsklassen möglich, wenn die Beziehung heterosyllabisch ist. Dies ist jedoch nicht der Fall, wenn Beziehung tautosyllabisch ist. Die dann noch verbleibenden Möglichkeiten sind in der nachfolgenden Tabelle zusammengestellt. Wie man sieht, kommen ’Silbe’ und ’Fuss’ überhaupt nicht vor. Das liegt daran, daß eine Silbe nicht Bestandteil einer Silbe sein kann und schon gar nicht ein Fuss, der ja eine Folge von Silben ist[37]. Ebensowenig kann eine tautosyllabische Kookkurrenzrestriktion zwischen einem Kopf und einer Schale bestehen, da der Kopf Bestandteil der Schale ist. Es müssen also alle Möglichkeiten ausgeschlossen werden, bei denen eine Kookkurrenzrestriktion zwischen Entitäten behauptet würde, die Bestandteile voneinander sind. Denn jeder Klassifikator, bei dem das nicht der Fall ist, wäre unerfüllbar, das heißt, seine Basismenge wäre die leere Menge.

'Erstes_Relat'	**'Zweites_Relat'**
'Segment'	'Kopf', 'Nukleus', 'Koda', 'Koerper', 'Schale', 'Reim'
'Kopf'	'Segment', 'Nukleus', 'Koda', 'Reim'
'Nukleus'	'Segment', 'Kopf', 'Koda', 'Schale'
'Koda'	'Segment', 'Kopf', 'Nukleus', 'Koerper'
'Koerper'	'Segment', 'Koda'
'Schale'	'Segment', 'Nukleus'
'Reim'	'Segment', 'Kopf'

Zulässige Kombinationen von 'Erstes_Relat' und 'Zweites_Relat', wenn eine tautosyllabische Kookkurrenzrestriktion vorliegt.

Um die Kombinationsmöglichkeiten einzuschränken, gibt es die Prädikate

`positive_constraint(Ursache,Wirkung)`

und

`negative_constraint(Ursache,Wirkung)`

`Ursache` ist eine Prolog-Klausel, die die *Pfade*[38] angibt, welche, wenn sie während der Klassifikation durchlaufen worden sind, die Wahl von bestimmten Nachfolgern eines Knotens verbieten (im Falle eines negativen Constraints), falls dieser Knoten *im weiteren Verlauf* durchlaufen wird, oder diese Wahl einschränken (im Falle eines positiven Constraints).

Im Grenzfall kann sogar *jeder* Nachfolger ausgeschlossen werden, womit erzwungen wird, daß der entsprechende Knoten nicht durchlaufen werden kann. Allerdings sollte man vor Anwendung einer solch drastischen Maßnahme das Klassifikationssystem gründlich überdenken. Durch die Verwendung von Pfaden besteht die Möglichkeit die Bildung von Klassifikatoren beliebig

---

[37] vgl. [Trask, 1996], Seite 147.

[38] Ein Pfad ist eine Folge von (Namen von) Knoten der Form $Name_1 : \ldots : Name_n$, wobei $n \geq 2$ und für alle $i \in \{1, \ldots, n-1\}$ gilt: $Name_{i+1}$ ist unmittelbarer Nachfolger von $Name_i$.

einzuschränken. Das heißt, daß das Durchlaufen eines beliebigen Stücks Pfades während der Klassifikation die Möglichkeiten der weiteren Bildung des Klassifikators beliebig einschränken kann.

Ein Klassifikator des Klassifikationssystem aus Beispiel 6 ist beispielsweise unerfüllbar, wenn das erste oder zweite Relat 'Fuss' oder 'Silbe' ist und die Kookkurrenz tautosyllabisch sein soll. Dies wird unterbunden durch

```
negative_constraint(('Erstes_Relat': 'Fuss';
                     'Erstes_Relat': 'Silbe';
                     'Zweites_Relat': 'Fuss';
                     'Zweites_Relat': 'Silbe'),
                     'Tauto_Heterosyllabisch': [tautosyllabisch]).
```

Allerdings ist die *Reihenfolge* zu beachten, in der die Nachfolger von UND-Knoten durchlaufen werden. Diese ist durch die Reihenfolge festgelegt, in der die Nachfolger in dem Argument `Liste` angeordnet sind. Der eben formulierte Constraint leistet nur deshalb das Gewünschte, weil die Knoten 'Erstes_Relat' bzw. 'Zweites_Relat' wenn überhaupt, dann *vor* dem Knoten 'Tauto_Heterosyllabisch' angelaufen werden.

Das Argument `Wirkung` hat immer die Form `Name:Liste`, wobei `Name` der Name des Knotens ist (das zweite Argument der `node/4`-Prädikate) und `Liste` eine Teilmenge der Nachfolger des entsprechenden Knotens ist. Für das Beispiel bedeutet das, daß dann, wenn der Knoten mit Namen 'Tauto_Heterosyllabisch' angelaufen wird, und vorher schon die entsprechenden Pfade durchlaufen wurden, `tautosyllabisch` nicht gewählt werden kann.

Dieselbe Bedingung kann auch durch einen positiven Constraint formuliert werden.

```
positive_constraint(('Erstes_Relat': 'Fuss';
                     'Erstes_Relat': 'Silbe';
                     'Zweites_Relat': 'Fuss';
                     'Zweites_Relat': 'Silbe'),
                     'Tauto_Heterosyllabisch': [heterosyllabisch]).
```

In diesem Fall wird mit dem Eintreten der Ursache bewirkt, daß beim Anlaufen von `'Tauto_Heterosyllabisch'` nur noch die Wahl `heterosyllabisch` getroffen werden kann.[39]. Hat die gleiche Ursache Wirkungen auf mehrere Knoten, dann muß für jeden Knoten ein eigener Constraint formuliert werden.

---

[39] Wenn nur noch eine Wahl getroffen werden kann, könnte man diese auch stillschweigend vom System treffen lassen, ohne den Klassifizierer davon zu informieren, aber ich denke, man sollte dem Klassifizierer die Kenntnis einer solchen interessanten Situation nicht vorenthalten.

Das Argument `Ursache` kann neben Pfadangaben beliebige Prolog-Terme enthalten, wie das folgende Beispiel zeigt. Dieser Constraint formuliert die Bedingung, daß bei *gleicher* Wahl des Nachfolgers der Knoten 'Erstes_Relat' und 'Zweites_Relat' als Wahl für den Nachfolger von 'Tauto_Heterosyllabisch' *nicht* tautosyllabisch genommen werden kann, wenn die Wahl bei 'Erstes_Relat' und 'Zweites_Relat' *nicht* 'Segment' war.

```
negative_constraint(('Erstes_Relat':X,
                     'Zweites_Relat':X,
                      X \== 'Segment'),
                     'Tauto_Heterosyllabisch':[tautosyllabisch]).
```

Prinzipiell kann auf eines der Prädikate `positive_constraint/2` bzw. `negative_constraint/2` verzichtet werden. Vom Programm werden auch sämtliche negativen Constraints vor dem ersten Aufruf zur Bildung eines Klassifikators in positive Constraints transformiert. Aber ich halte es für eine nützliche Erleichterung, sowohl *verbietende* als auch *gebietende* Constraints formulieren zu können. Bei den transformierten Constraints sind auch sämtliche Pfade $Name_1 : \ldots : Name_n$ mit $n \geq 2$ ersetzt durch eine *Konjunktion* von Pfaden der Form $Name_1 : Name_2, \ldots, Name_{n-1} : Name_n$. Dies ist möglich, da die Namen aller Knoten verschieden sind.

Auch die Integrierung der Constraints in den Klassifikationsalgorithmus ist sehr einfach. Von jedem Knoten, der durchlaufen wurde, wird ein Eintrag der Form `Name:Nachfolger` in die Prolog-Datenbasis gemacht. Handelt es sich um einen ODER-Knoten, so ist `Nachfolger` der Name des gewählten Nachfolgers. Handelt es sich um einen UND-Knoten, so wird von jedem Nachfolger ein solcher Eintrag gemacht. Diese Einträge werden bei Backtracking wieder aus der Datenbasis entfernt. Dies ist zum Beispiel der Fall, wenn der Klassifizierer eine Wahl rückgängig macht.

Bei jedem Knoten, der angelaufen[40] wird, wird nun zuerst geprüft, ob er Constraints unterworfen ist. Dazu werden von sämtlichen Constraints zu diesem Knoten, deren Ursachen anhand der Einträge der Form `Name:Nachfolger` eingetreten sind, die Wirkungen gesammelt. Jede solche Wirkung hat die Form `Knoten:Liste`. Von allen Listen wird der Durchschnitt gebildet und nur aus diesem Durchschnitt kann ein Nachfolger gewählt werden.

---

[40] vgl. die Sprachregelungen auf Seite 35.

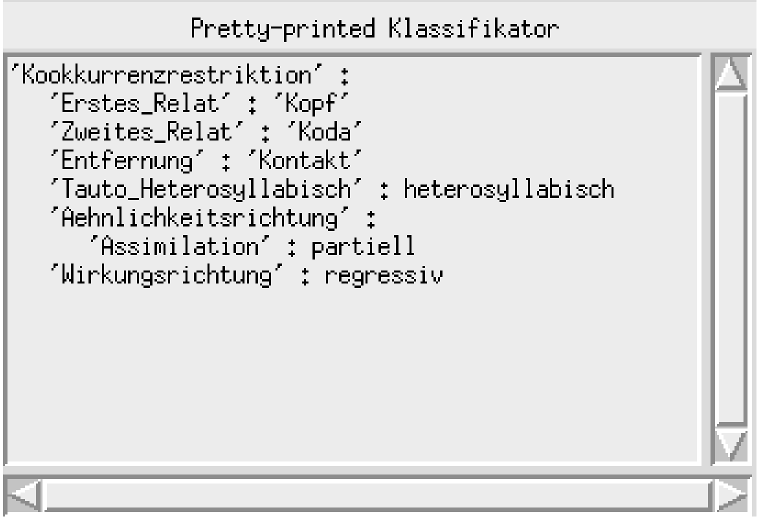
```
traverse_node(Node:Category) :-
        node(Type,Node,OldSubCatList),
        check_constraints(Node,NewSubCatList,NewSubCatList), %%% neu!
        (Type == or ->
             get_choice(Node,NewSubCatList,SubCat),
             (node(or,SubCat,[]) ->
                  Category = SubCat,
             ;
             assert_and_retract_on_backtracking(Node:SubCat), %%% neu!
             Category = [SubCat:Cat],
             traverse_node(SubCat:Cat))
        ;
        Type == and,
        traverse_nodes(Node,NewSubCatList,Category))

%%% traverse_nodes(+Node,+SubCatList,-PathList)
traverse_nodes(_,[],[]).
traverse_nodes(Node,[SubCat|SubCatList],[SubCat:Cat|Cats]) :-
        assert_and_retract_on_backtracking(Node:SubCat),
        traverse_node(SubCat:Cat),
        traverse_nodes(Node,SubCatList,Cats).

%%% check_constraints(+Node,+OldSubCats,-NewSubCats)
check_constraints(Node,OldSubCats,NewSubCats) :-
        (setof(NewCats,check_constraint(Node,NewCats),NewSubCatList) ->
             map_intersection(NewSubCatList,NewSubCats)
        ;
        NewSubCats = OldSubCats).

%%% check_constraint(+Node,-SubCats)
check_constraint(Node,SubCats) :-
        constraint(Conditions,Node:SubCats),
        call(Conditions).
```

Das Prädikat `traverse_nodes` ist um die Argumentstelle `+Node` erweitert worden, damit Einträge in die Datenbasis gemacht werden können, wenn die entsprechenden Knoten angelaufen werden.

Klassifikatoren sind im allgemeinen recht lange Ausdrücke, zu lange jedenfalls, als daß sie in ihrer ganzen *Länge* am Bildschirm dargestellt werden könnten. Dies sieht man auch am folgenden Bild, das einen Ausschnitt aus der grafischen Oberfläche zur Erstellung von Klassifikatoren darstellt. Das Bild zeigt eine Listbox, die mehrere Klassifikatoren enthält.

```
'Kookkurrenzrestriktion' : ['Erstes_Relat' : 'Koerpe
'Kookkurrenzrestriktion' : ['Erstes_Relat' : 'Silbe',
'Kookkurrenzrestriktion' : ['Erstes_Relat' : 'Reim',
'Kookkurrenzrestriktion' : ['Erstes_Relat' : 'Segmen
'Kookkurrenzrestriktion' : ['Erstes_Relat' : 'Kopf',
'Kookkurrenzrestriktion' : ['Erstes_Relat' : 'Reim',
'Kookkurrenzrestriktion' : ['Erstes_Relat' : 'Nukleu
```

Deshalb kann man sich diese optisch aufbereitet in einem eigenen Fenster anzeigen lassen. Der markierte Klassifikator sieht dann so aus:



```
'Kookkurrenzrestriktion' :
   'Erstes_Relat' : 'Kopf'
   'Zweites_Relat' : 'Koda'
   'Entfernung' : 'Kontakt'
   'Tauto_Heterosyllabisch' : heterosyllabisch
   'Aehnlichkeitsrichtung' :
      'Assimilation' : partiell
   'Wirkungsrichtung' : regressiv
```

5.1.2 Definition von Beschreibungsschemata

Beschreibungsschemata werden von den Klassifikatoren definiert.[41] Da für jeden Knoten, welcher zur Bildung eines Klassifikators durchlaufen wurde, ein Eintrag in der Datenbasis der Form `Knoten : Nachfolger` vorliegt[42], kann diese Information direkt in die Bildung eines Beschreibungsschemas einfließen. Die Beschreibungsschemata sind neben den UND- bzw. ODER-Knoten

---

[41] vgl. Seite 4.

[42] Diese Einträge werden dann erst wieder aus der Datenbasis entfernt, wenn mit der Bildung eines neuen Klassifikators begonnen wird.

und den Constraints die dritte Klasse von *Fakten*, welche ein Klassifikationssystem definieren. Sie werden durch das Prädikat

`scheme(Ursache,Wirkung)`

realisiert, wobei die Argumente `Ursache` und `Wirkung` in Analogie zu den gleichnamigen Argumenten der Constraints stehen. Das Argument `Ursache` kann also neben Pfadangaben beliebige Prolog-Terme enthalten und das Argument `Wirkung` kein wiederum ein beliebiger Prolog-Term sein. Wurde ein Klassifikator gebildet, so werden die Wirkungen sämtlicher Schemata, deren Ursachen von der Datenbasis erfüllt werden, aufgesammelt und erscheinen in dem Fenster der Oberfläche, in welchem der Klassifizierer die Phänomenbeschreibung eingibt. Ich nehme als Beispiel die Definitionen von *Assimilation*, *partieller Assimilation* und *totaler Assimilation*.[43] In dem Klassifikationssystem der Kookkurrenzrestriktionen ist Assimilation zwischen beliebigen Relaten (soweit dies nicht durch Constraints ausgeschlossen ist) möglich.

Definition 14

- Assimilation liegt zwischen zwei Relaten vor, wenn sie ein gemeinsames Merkmal haben und dieses gemeinsame Merkmal denselben Wert hat.
- $$\forall s_1 \forall s_2 (assimilation(s_2, s_1) :\Leftrightarrow relat(s_1) \wedge relat(s_2) \wedge \exists m (merkmal(s_1, m) = merkmal(s_2, m))).$$

  Eine Assimilation zwischen zwei Sprachlauten ist partiell, wenn die Sprachlaute nicht identisch sind.
- $$\forall s_1 \forall s_2 (partielle\_assimilation(s_1, s_2) :\Leftrightarrow relat(s_1) \wedge relat(s_2) \wedge assimilation(s_1, s_2) \wedge s_1 \neq s_2).$$

  Eine Assimilation zwischen zwei Sprachlauten ist total, wenn die Sprachlaute dieselben Merkmale mit denselben Werten haben.

$$\forall s_1 \forall s_2 (totale\_assimilation(s_1, s_2) \Leftrightarrow relat(s_1) \wedge relat(s_2) \wedge \forall m (merkmal(s_1, m) = merkmal(s_2, m))).$$

Diese Definitionen werden in die folgenden Schemata übersetzt:

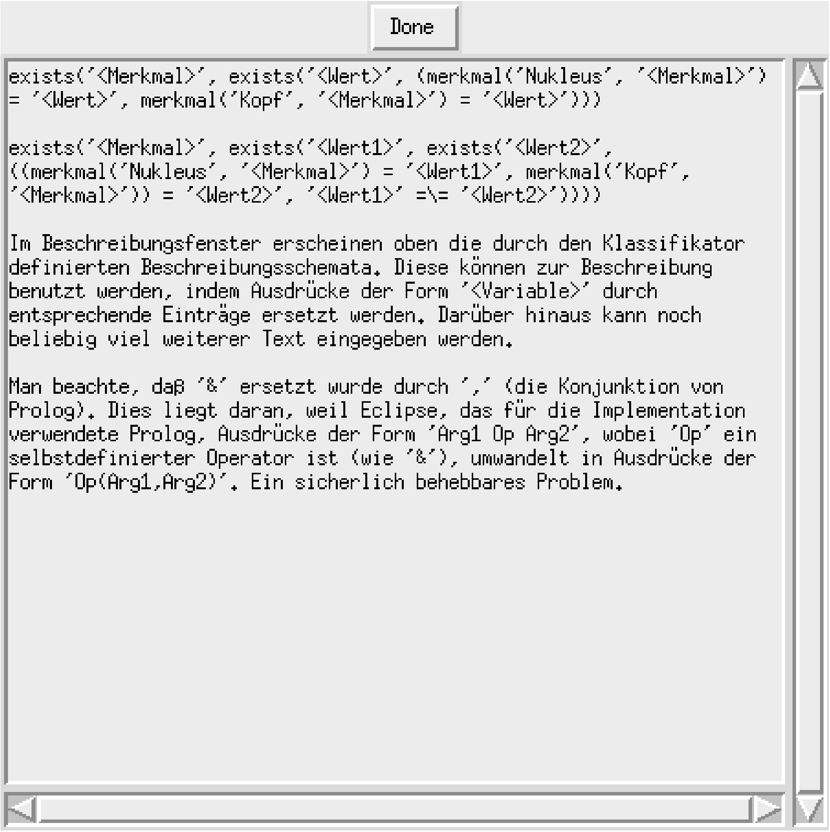
```
scheme(('Aehnlichkeitsrichtung':'Assimilation',
        'Erstes_Relat':R1,
        'Zweites_Relat':R2),
         (exists('<Merkmal>',exists('<Wert>',
```

---

[43] vgl. Def. 1, Seite 10, Def. 3, Seite 10 und Kor. 1, Seite 10.

```
                        ((merkmal(R1,´<Merkmal>´) = ´<Wert>´ &
                          merkmal(R2,´<Merkmal>´) = ´<Wert>´)))))).

scheme((´Assimilation´:partiell,
        ´Erstes_Relat´:R1,
        ´Zweites_Relat´:R2),
        (exists(´<Merkmal>´,exists(´<Wert1>´, exists(´<Wert2>´,
                      ((merkmal(R1,´<Merkmal>´) = ´<Wert1>´ &
                        merkmal(R2,´<Merkmal>´) = ´<Wert2>´ &
                        ´<Wert1>´ =\= ´<Wert2>´))))))).

scheme((´Assimilation´:total,
        ´Erstes_Relat´:R1,
        ´Zweites_Relat´:R2),
        (forall(´<Merkmal>´,forall(´<Wert>´,
             ((merkmal(R1,´<Merkmal>´) = ´<Wert>´ &
               merkmal(R2,´<Merkmal>´) = ´<Wert>´)))))).
```

Möchte man zu einem Klassifikator eine Beschreibung eingeben, so klickt man mit der linken Maustaste auf den markierten Klassifikator und es erscheint das folgende Eingabefenster.



```
exists(´<Merkmal>´, exists(´<Wert>´, (merkmal(´Nukleus´, ´<Merkmal>´)
= ´<Wert>´, merkmal(´Kopf´, ´<Merkmal>´) = ´<Wert>´)))

exists(´<Merkmal>´, exists(´<Wert1>´, exists(´<Wert2>´,
((merkmal(´Nukleus´, ´<Merkmal>´) = ´<Wert1>´, merkmal(´Kopf´,
´<Merkmal>´)) = ´<Wert2>´, ´<Wert1>´ =\= ´<Wert2>´))))
```

Im Beschreibungsfenster erscheinen oben die durch den Klassifikator definierten Beschreibungsschemata. Diese können zur Beschreibung benutzt werden, indem Ausdrücke der Form ´<Variable>´ durch entsprechende Einträge ersetzt werden. Darüber hinaus kann noch beliebig viel weiterer Text eingegeben werden.

Man beachte, daß ´&´ ersetzt wurde durch ´,´ (die Konjunktion von Prolog). Dies liegt daran, weil Eclipse, das für die Implementation verwendete Prolog, Ausdrücke der Form ´Arg1 Op Arg2´, wobei ´Op´ ein selbstdefinierter Operator ist (wie ´&´), umwandelt in Ausdrücke der Form ´Op(Arg1,Arg2)´. Ein sicherlich behebbares Problem.

Die Integration der Beschreibungsschemata in den Klassifikationsalgorithmus besteht aus genau einer Programmzeile. Diese bewirkt nichts anderes, als daß sämtliche Schemata, deren Ursachen von der Datenbasis erfüllt werden, in einer Liste gesammelt werden.

```
build_classifier :-
        root(StartNode),
        traverse_node(StartNode:Category),
        findall(Scheme,(scheme(Condition,Scheme),Condition),Schemata). %%% neu!
```

5.1.3 Erstellen von Anfragen

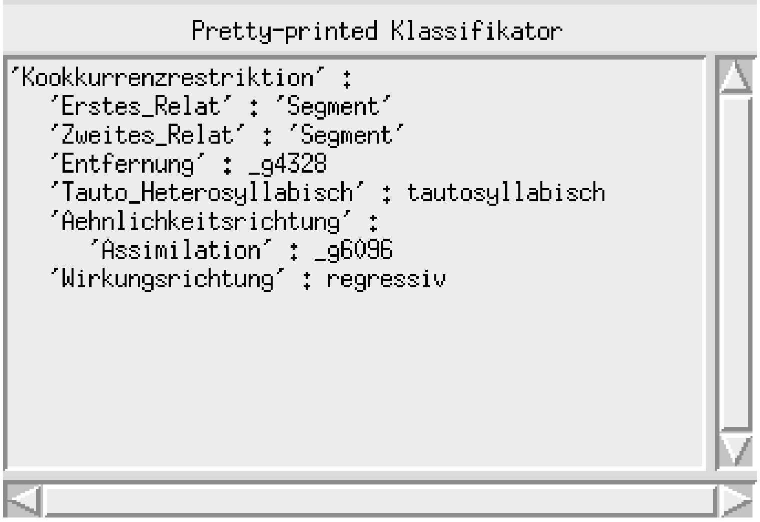
Das Erstellen von Anfragen[44] an ein Klassifikationssystem ist nahezu identisch zum Erstellen eines Klassifikators. Klassifikatoren sind UND-Bäume und man erhält aus einem Klassifikator eine Anfrage, indem man einen beliebigen Teilbaum durch eine Variable ersetzt. Um dies zu ermöglichen bedarf es nur einer einzigen Änderung im Prädikat `traverse_node/2` und der Möglichkeit von der graphischen Oberfläche aus statt eines Nachfolgers den Ausdruck `skip` auszuwählen.

```
traverse_node(Node:Category) :-
        node(Type,Node,OldSubCatList),
        check_constraints(Node,NewSubCatList,NewSubCatList), %%% neu!
        (Type == or ->
             get_choice(Node,NewSubCatList,SubCat),
             (SubCat == skip ->           %%% neu!
                  Category = Category     %%% neu!
             ;
             (node(or,SubCat,[]) ->
                  Category = SubCat,
             ;
             assert_and_retract_on_backtracking(Node:SubCat), %%% neu!
             Category = [SubCat:Cat],
             traverse_node(SubCat:Cat)))
        ;
        Type == and,
        traverse_nodes(Node,NewSubCatList,Category))
```

Die Programmzeile `Category = Category` könnte durch ein einfaches `true` ersetzt werden. Aber so ist klarer, was geschieht. Die folgenden Bilder zeigen zwei so erstellte Anfragen.

---

[44] Zum Begriff *Anfrage* siehe Definition 12 auf Seite 29.



```
'Kookkurrenzrestriktion' :
   'Erstes_Relat' : 'Segment'
   'Zweites_Relat' : 'Segment'
   'Entfernung' : _g4328
   'Tauto_Heterosyllabisch' : tautosyllabisch
   'Aehnlichkeitsrichtung' :
      'Assimilation' : _g6096
   'Wirkungsrichtung' : regressiv
```

```
'Kookkurrenzrestriktion' :
   'Erstes_Relat' : _g21256
   'Zweites_Relat' : _g22396
   'Entfernung' : _g23536
   'Tauto_Heterosyllabisch' : _g24390
   'Aehnlichkeitsrichtung' : _g25252
   'Wirkungsrichtung' : _g26116
```

6 Phonotaktik: Ein erster Versuch

Zunächst einige Definitionen zum Begriff "Phonotaktik":

[Trask, 1996]: In a given language, the set of constraints on the possible sequences of consonant and vowel phonemes within a word, a morpheme or a syllable.

[Crystal, 1991]: A term used in Phonology to refer to the sequential arrangements (or 'tactic behaviour') of phonological units in a language.

[Bußmann, 1990]: Lehre von den für eine bestimmte Sprache zugelassenen Laut- oder Phonemkombinationen. Für jede Sprache gibt es spezifische phonotaktische Regeln, die die Verbindbarkeit von Phonemen in verschiedenen Positionen (An-, In- und Auslaut) beschreiben (...). Die auftretenden Restriktionen sind teils einzelsprachlich, teils universell gültig.

[Glück, 1993]: Phonologische Teildisziplin, die die Regularitäten der Phonemverbindungen einer Sprache zu Silben, Morphemen und Wörtern zum Gegenstand hat.

[Malmkjaer, 1991] (Artikel Phonemics): the study of the permitted or non-permitted arrangements or sequences of phonemes in a given language.

[Lewandowski, 1985]: Die Untersuchung der Distribution bzw. Verbindbarkeit von Phonemen, die Aufstellung von Regeln bzw. Tabellen, die die Einschränkungen der Kombinierbarkeit an bestimmten Positionen des Morphems oder des Wortes beschreiben.

Die Phonotaktik untersucht also Sequenzen (Folgen) von Phonemen. Das Ziel der Phonotaktik besteht darin herauszufinden, welchen Bedingungen (Constraints) Sequenzen von Phonemen genügen müssen, um entweder korrekte "phonologische Phrasen" [Bird and Klein, 1994] der untersuchten Sprache oder korrekte Bestandteile an bestimmten Positionen von "phonologischen Phrasen" zu sein. Phonologische Phrasen sind dabei insbesondere solche Phonemsequenzen, welche Silbenbestandteile, Silben, Morpheme oder Wörter der Sprache sind.

Die Phonotaktik untersucht nun, *welche Phoneme in welchen Strukturen an welchen Positionen unter welchen Bedingungen vorkommen bzw. nicht vorkommen.*

Zunächst ist es angebracht, einiges über die Struktur von Silben, Morphemen und Wörtern zu sagen. Silben bestehen aus

- einem optionalen Silbenkopf, welcher aus einer Folge von einem oder mehreren Konsonanten besteht, gefolgt von
- einem Silbenkern bzw. Nukleus, welcher aus einem Vokal[45], einem Diphthong oder einem vokalischen Konsonanten besteht, gefolgt von
- einer optionalen Silbenkoda, welche aus einer Folge von einem oder mehreren Konsonanten besteht.

Besteht bei einer Silbe der Kopf bzw. die Koda nur aus einem Phonem P, so sagt man, daß dieses Phonem der Kopf bzw. die Koda *ist* und nicht, daß Kopf bzw. Koda eine Folge mit nur einem Element, dem Phonem P, sind.

---

[45] Die Frage, ob im Silbenkern mehrere Vokale vorkommen können, muß ich unbeantwortet lassen. Ein zu diskutierendes Bespiel wäre /leo/.

Als nächstes möchte ich einige Beispiele für phonotaktische Regeln geben[46]:

- In der Sprache O gibt es keine Silbe, deren Kopf aus mehr als drei Konsonanten besteht.
- Wenn in der Sprache O ein Silbenkopf aus drei Konsonanten besteht, dann ist der erste Konsonant ein /s/, der zweite ist ein stimmloser Plosiv (/p/, /t/ oder /k/) und der dritte ist ein Liquida (/l/ oder /r/) oder Gleitlaut (/w/ oder /j/).
- In der Sprache O haben bei sämtlichen in einer Wortform vorkommenden Vokalen das Merkmal *Backness* den gleichen Wert.
- In der Sprache O hat in einer Wortform jeder Suffixnukleus die gleiche *Backness* wie der ihm vorhergehende Nukleus und jeder *hohe* Suffixnukleus hat darüber hinaus auch die gleiche *Rounding* wie der ihm vorhergehende Nukleus.
- In der Sprache O ist das erste Phonem einer Ergänzungsfrage immer ein /w/.

Die beiden letzten Beispiele sind insofern interessant, weil in ihnen Begriffe vorkommen, die nicht zur Phonologie gehören, sondern zur Morphologie bzw. Syntax. Deshalb ist auch umstritten, ob es sich bei ihnen um phonotaktische Regeln handelt oder nicht. In den oben zitierten Phonotaktik-Definitionen verschiedener Linguisten werden sie jedenfalls nicht explizit ausgeschlossen.

Festgehalten kann jedenfalls werden, daß es in der Phonotaktik darum geht, eine Menge von Aussagen zu formulieren, anhand derer Phonemsequenzen (auch einelementige) daraufhin untersucht werden können, ob sie korrekte Silben, Morpheme, Wortformen etc. sind bzw. an welchen Positionen innerhalb von Silben, Morphemen, Wortformen etc. sie korrekterweise als Bestandteil vorkommen können.

Die in der Linguistik betrachteten Phonemsequenzen sind immer endliche Folgen. Auch wenn manche Grammatikregeln prinzipiell die Bildung von Ausdrücken beliebiger Länge erlauben, so ist die Länge jedes tatsächlich gebildeten Ausdrucks doch immer endlich. Zunächst kann man die Aussagen über endliche Phonemfolgen in zwei Arten einteilen:

1. Aussagen über die Länge n der Folgen

---

[46] Die ersten beiden Beispiele habe ich [Trask, 1996] entnommen.

2. Aussagen über die Phoneme, welche an bestimmten Positionen der Folgen stehen, bzw. Aussagen über erlaubte und verbotene Verteilungen von Phonemen an bestimmte Positionen der Folgen

Aussagen über die Länge von Folgen sind relativ leicht charakterisierbar. Endliche Folgen haben endlich viele Elemente. Also ist jede Aussage der Form

$$n \in N, \text{ wobei } N \text{ eine Teilmenge von } \mathbb{N} \text{ ist,}$$

eine Aussage über die Anzahl der Elemente der Folge.

Eine solche Aussage $n \in N$ mit $N = \{n_1, \ldots, n_i\}$ sondert aus der Klasse aller endlichen Folgen die Klasse derjenigen endlichen Folgen aus, deren Länge einen der Werte in N hat. Davon werden jedoch im allgemeinen nur die folgenden Spezialfälle von Belang sein:

- Die Anzahl der Elemente wird exakt angegeben: N ist eine Einermenge.
- Die Anzahl liegt in einem bestimmten Bereich:
  - N ist ein Intervall von natürlichen Zahlen.
  - in N gibt es ein größtes Element (d.h. N ist ein Intervall "mit Lücken").
- Die Anzahl ist beliebig: $N = \mathbb{N}$. Dies ist der Default.

Werden mehrere Folgen betrachtet, so möchte man auch Aussagen formulieren können, welche die Längen dieser Folgen miteinander in Beziehung setzen. Bei zwei Folgen sind dies insbesondere Aussagen der Form:

- Die Anzahl der Elemente der einen Folge ist gleich (größer, kleiner,...) der Anzahl der Elemente der anderen Folge.

Bei endlichen Folgen kann man sich auf zweierlei Weisen auf Folgenelemente beziehen: indem man von vorne beginnend durchzählt, oder indem man von hinten beginnend durchzählt. Für die erste möglichkeit nimmt man einfach die natürlichen Zahlen (ohne die Null). Für die zweite möglichkeit bieten sich die negativen ganzen Zahlen an, wobei die Position des letzten Elements der Folge mit -1 bezeichnet wird.

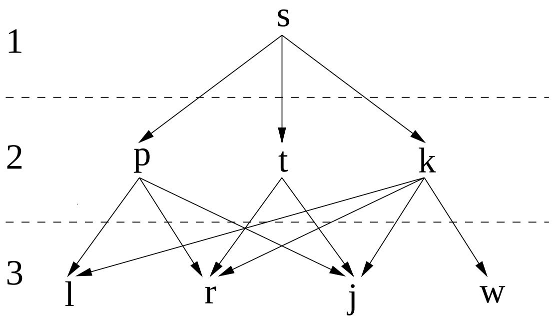
Aussagen über die erlaubten bzw. verbotenen Verteilungen von Phonemen an bestimmte Positionen einer Folge sind nicht so leicht klassifizierbar wie

Aussagen über die Länge einer Folge. Ich möchte dies zuerst an einem sehr einfachen Beispiel deutlich machen.

Angenommen, man möchte die Silbenköpfe einer gegebenen Sprache untersuchen, welche aus drei Konsonanten bestehen. Die in der Sprache vorkommenden Konsonanten seien $K_1, \ldots, K_n$. Eine mögliche systematische Verfahrensweise bestünde darin, zunächst alle dreielementigen Silbenköpfe aufzuzählen und diese dann auf Regelmäßigkeiten hin zu untersuchen. Für das Englische[47] wäre dies die Menge $\{spl, spr, spj, str, stj, skl, skr, skj, skw\}$. Eine Beschreibung, welche nur angibt, an welcher Position welche Konsonanten stehen können, ist das erste Beispiel von oben. Eine tabellarische Darstellung dieses Beispiels ist:

Position	Konsonanten
1	s
2	p,t,k
3	l,r,j,w

Bemerkenswert ist, daß an erster Position nur das Phonem **s** vorkommen kann. Aus dieser Tabelle kann man jedoch nicht ersehen, daß die Folgen *spw, stl, stw* tatsächlich *nicht* als Silbenköpfe im Englischen vorkommen. Eine in dieser Hinsicht vollständige Beschreibung stellt der folgende Graph dar.



Aber auch diese optisch aufbereitete Darstellung zählt die möglichen Phonemkombinationen nur auf und gibt keinen Aufschluß über eventuelle Gemeinsamkeiten der an bestimmten Positionen vorkommenden Phoneme. Diese sieht man erst, wenn man die Merkmale der einzelnen Phoneme betrachtet.

---

[47] vgl. [Scherer and Wollmann, 1986], Seite 270.

Position	Konsonant	Artikulationsart	Artikulationsstelle	Stimmhaftigkeit
1	s	frikativ	alveolar	stimmlos
2	p	plosiv	bilabial	stimmlos
	t	plosiv	alveolar	stimmlos
	k	plosiv	velar	stimmlos
3	l	liquida	alveolar	
	r	liquida	postalveolar	
	j	halbvokal	palatal	
	w	halbvokal	bilabial	

Jetzt kann man folgende Regelmäßigkeiten feststellen:

- An zweiter Position steht immer ein stimmloser Plosiv.
- An dritter Position steht immer ein Liquida oder ein Halbvokal.

Allerdings geben die Merkmale keinen Aufschluß darüber, warum

- an dritter Position *immer* die Phoneme *r* und *j* stehen können,
- das Phonem *w* nur dann an dritter Position vorkommen kann, wenn an zweiter Position das Phonem *k* vorkommt,
- das Phonem *l* nicht nach dem Phonem *t* an dritter Position vorkommen kann.

Ausserdem ist noch nicht klar geworden, wie eine *systematische* Beschreibung der dreielementigen Silbenköpfe auszusehen hat. Eine Möglichkeit besteht darin, zuerst - wie oben - eine Tabelle aufzustellen aus der hervorgeht, welche Phoneme an welchen Positionen stehen können. Anschließend überprüft man anhand der Phonemmerkmale, ob die an einer bestimmten Position erlaubten Phoneme gewisse Merkmale gemeinsam haben und ob diese Merkmale in systematischem Zusammenhang mit den Merkmalen von Phonemen an anderen Positionen zusammenhängen. Schließlich muß man noch die Fälle aufzählen, die unter keine Regelmäßigkeiten fallen. Allerdings ist das eben skizzierte Verfahren ad hoc. Darüber hinaus kann es immer noch sein, daß das Vorkommen von Phonemen an bestimmten Positionen von Entitäten abhängt, die selbst nicht Teil der untersuchten Strukturen (hier die Silbenköpfe) sind. Es ist ja durchaus denkbar, daß z. B. der auf den Silbenkopf folgende Silbenkern das Vorkommen von Phonemen im vorausgehenden Silbenkopf beeinflußt. Die linguistischen Entitäten, welche die Phonotaktik untersucht, sind immer eingebettet in übergeordnete Strukturen, und deshalb kann man nicht ausschließen, daß beliebige andere Teile dieser übergeordneten Strukturen Einfluß auf die möglichen Phonemkombinationen haben.

7 Phonotaktik: Ein zweiter Versuch

Der Begriff "Taktik" entstammt dem Altgriechischen und steht für die Kunst der Anordnung und Aufstellung. [Crystal, 1991] definiert:

> "**tactic(s) (taxis)** A general term used in phonetics and linguistics to refer to the systematic arrangements of units in linear sequence at any linguistic level. The commonest terms based on this notion are: **phonotactics**, dealing with the sequential arrangement of sounds; **morphotactics** (...) and **syntactics** (...)."

Zunächst möchte ich den Begriff "Taktik" etwas deutlicher charakterisieren: Bei einer "Taktik" geht es erstens um

> Entitäten eines bestimmten Typs τ.

Im Falle der Phonotaktik handelt es sich hierbei um Phoneme, oder - mit anderen Worten - um Entitäten des Typs *phonem*.

Zweitens geht es um

> Entitäten eines weiteren Typs σ, welche Entitäten des Typs τ als *Konstituenten* enthalten.

Bei der Phonotaktik können dies zum Beispiel die Strukturen von Silben oder Wortformen sein. Eine Silbe besteht immer aus einem Nukleus und einem optionalen vorangehenden Onset, sowie einer ebenfalls optionalen nachfolgenden Coda. Onset und Coda sind Folgen von Konsonanten, der Nukleus eine (in der Regel einelementige) Folge von Vokalen oder vokalischen Konsonanten.

Drittens geht es um

> eine Menge von *Positionen* der Entitäten des Typs τ innerhalb von Entitäten des Typs σ.

Ein Klassifikationssystem für die Phonotaktik könnte dann etwa wie folgt aussehen:[48] Die "Messages" sind die Kommentare zu den Knoten. Ich habe sie der Übersichtlichkeit halber nicht in die Knoten geschrieben.

Phonotaktik spaltet sich auf in Okkurrenz- und Kookkurrenzphaenomene

```
node(or,´Phonotaktik´,´´,[´Okkurrenz´,Kookkurrenz´]).
```

---

[48] vgl. [Comrie and Smith, 1977], Abschnitt 3.2 *Phonotactics*.

Okkurrenzphaenomene zerfallen in zwei orthogonale Klassen. Die erste bestimmt, ob eine Vorkommenserlaubnis oder eine Vorkommenseinschraenkung vorliegt. Die zweite gibt die Struktur an, welche betrachtet wird.

```
node(and,'Okkurrenz','',[erlaubt_verboten,'Struktur']).
node(and,'Kookkurrenz','',[erlaubt_verboten,'Strukturen']).
node(or,erlaubt_verboten,'',[erlaubt,verboten]).
```

Die Strukturen bei Okkurrenzen werden nach drei Dimensionen eingeteilt. 'Strukturart' gibt die Entitaet an, deren interne Struktur betrachtet wird. 'Konstituente' steht fuer die Konstituente der Entitaet, deren Vorkommen betrachtet wird. 'Position' gibt schliesslich die Position der Konstituente innerhalb der Entitaet an.

```
node(and,'Struktur','',['Strukturart','Konstituente','Position']).

node(or,'Strukturart','',['Wort','Silbe','Silbenfolge']).

node(or,'Position','',[initial,medial,final,beliebig]).
```

Die Strukturen bei Kookkurrenzen werden nach sechs Dimensionen eingeteilt. Nach 'Strukturart','Konstituente' und 'Position' (wie oben). Die zweite 'Konstituente' hat eine 'Relative_Position' zur ersten. Ausgedrueckt wird die Kookkurrenz zweier Konstituenten an bestimmten Positionen. Mit kontakt_distanz wird angegeben, ob die Konstituenten unmittelbar aneinandergrenzen oder nicht.

```
node(and,'Strukturen','',['Strukturart',
                          'Konstituente',
                          'Position',
                          'Konstituente',
                          'Relative_Position',
                           kontakt_distanz]).

node(or,'Relative_Position','',[nachfolgend,vorhergehend]).

node(or,'Konstituente','',['Konsonant',
                           'Vokal',
                           'Konsonanten_cluster',
                           'Silbe',
                           'Kopf',
                           'Nukleus',
                           'Koda']).

node(or,kontakt_distanz,[kontakt,distanz]).
```

Bei den Okkurrenzen wird die Position any ausgeschlossen, da diese durch die anderen Positionen abgedeckt wird.

```
negative_constraint('Phonotaktik':'Okkurrenz','Position':[beliebig]).
```

Wenn als Struktur Silbe gewählt wurde, kann als Konstituente nicht wiederum Silbe gewählt werden.

```
negative_constraint('Struktur':'Silbe','Konstituente':['Silbe']).
```

Bei den Silbenkonstituenten Kopf, Nukleus und Koda sind keine Positionen anzugeben, da diese in den Definitionen der Konstituenten enthalten sind.

```
positive_constraint('Struktur':'Silbe','Position':[]).
```

Dies ist ein Fall der Anwendung des auf Seite 43 besprochenen drastischen Mittels, einen Knoten komplett auszublenden.

Ebensowenig macht es Sinn, bei silbeninternen Kookkurrenzen relative Positionen anzugeben.

```
positive_constraint('Struktur':'Silbe','Relative_Position':[]).
```

Bei silbeninternen Kookkurrenzen kann nicht zweimal die gleiche Konstituente vorkommen.

```
negative_constraint(('Struktur':'Silbe','Konstituente':X),'Konstituente':[X]).
```

Bei silbeninternen Kookkurrenzen ist *kontakt_distanz* bereits durch die Konstituenten festgelegt. (z.B. ist die Kookkurrenz Kopf-Koda innerhalb einer Silbe immer *distanz*

```
positive_constraint('Struktur':'Silbe',kontakt_distanz:[]).
```

8 Rückblick und Ausblick

Klassifikationssysteme, Klassifikatoren und Anfragen im Sinne der Definitionen 10, 11 und 12 lassen sich sehr gut in einer Logikprogrammiersprache wie Prolog implementieren. An seine Grenzen dürfte dieser Ansatz erst bei größeren Datenmengen stoßen. Die werden allerdings beim Füllen der Datenbank mit Sprachdaten schnell erreicht.

Ein Problem war die *inhaltliche* Erstellung eines Klassifikationssystems. Wie die beiden Versuche zur Phonotaktik zeigen, sind die Ergebnisse sehr eingeschränkter Natur. Wie brauchbar das hier vorgestellte *Konzept* von Klassifikationssystemen ist, wird sich erst zeigen, wenn Linguisten damit umgehen.

Durch die Verwendung von Klassifikatoren und Beschreibungsschemata wird die Beschreibung linguistischer Phänomene in hohem Maße standardisiert. Da als Beschreibungsschemata jeder Prolog-Term dienen kann, ist zudem sichergestellt, daß die Schemata hinreichend ausdrucksstark erstellt werden können.

Als Knoten sind im Augenblick nur die ODER- bzw. UND-Knoten vorhanden. In einer früheren Implementation hatte ich auch sogenannte ITERATE-Knoten vorgesehen. Diese Knoten können beliebig oft hintereinander angelaufen werden. Dies schien mir sinnvoll, um Folgen (z. B. von Phonemen) in den Klassifikator einzubinden. Allerdings habe ich diese Idee nicht weiter verfolgt. Eine andere Art von Knoten, die ich ebenfalls in der finalen Version weggelassen habe, sind die von mir so genannten ODER-UND-Knoten. Die Nachfolgerliste eines solchen Knotens ist keine einfache Liste, sondern eine Liste von Listen. Man kann eine der Listen wählen, muß dann aber jeden Knoten der gewählten Liste durchlaufen (deswegen ODER-UND). Dadurch kann man sich einiges an sprachschöpferischer Tätigkeit sparen. Ein Beispiel wäre etwa

```
node(or_and,´Phonotaktik,´´,[[´Okkurrenz´,´Struktur´],
                             [´Kookkurrenz´,´Strukturen´]]).
node(or,´Okkurrenz´,´´,[erlaubt,verboten]).
node(or,´Kookkurrenz´,´´,[erlaubt,verboten]).
```

Es enfällt das unschöne `erlaubt_verboten`.

Der folgende Programmcode von 152 Zeilen enthält den Kernalgorithmus zur Erstellung von Klassifikatoren und zur Erstellung von Anfragen inklusive der Hilfsprädikate.

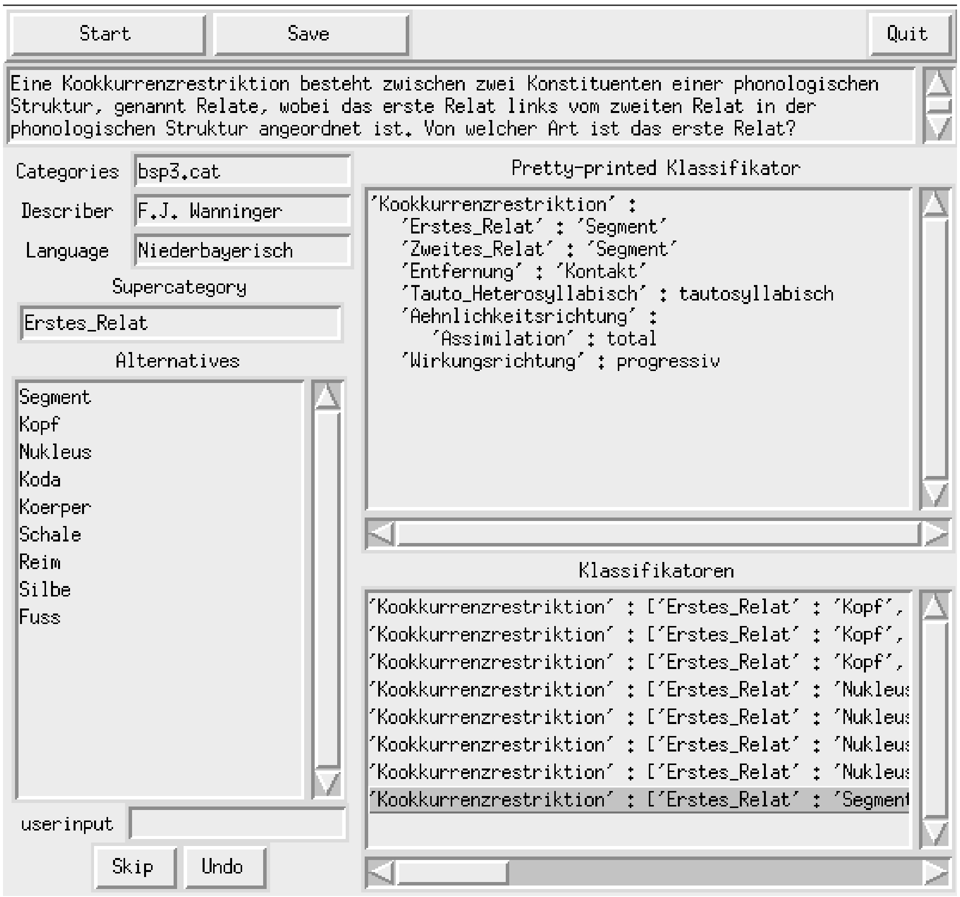
```
:- dynamic ´:´/2,
           node/5,
           eintrag/4,
           scheme/2,
           positive_constraint/2,
           negative_constraint/2,
           constraint/2.
:- lib(tk), tk([]), tcl(´source new.tcl´).
:- op(600,xfy,:).
:- op(1000,xfy,&).

build_classifier :-
        initialize_constraints,
        retract_all(_:_),
        root(StartNode),
        traverse_node(StartNode:Category),
        findall(Scheme,(scheme(Condition,Scheme),Condition),Schemata),
        tcl_string(StartNode:Category,TclCategory),
        tcl(´fill_listbox .cat.right.terms.list2 { ## }´,[TclCategory]),
        tcl_string(Schemata,TclSchemes),
        tcl(´lappend Schemes  ## ´,[TclSchemes]),
        tcl(´show_cat  ## ´,[TclCategory]).

traverse_node(Node:Category) :-
        node(Type,Node,Message,OldSubCats),
        check_constraints(Node,OldSubCats,SubCats),
        (SubCats == [] ->
             Category = []
        ;
        (Type == or ->
             display_message(Message),
             get_choice(Node,SubCats,SubCat),
             (SubCat == skip ->
                  Category = Category
             ;
             assert_and_retract_on_backtracking(Node:SubCat),
             (node(or,SubCat,_,[]) ->
                  Category = SubCat
             ;
             Category = [SubCat:Cat],
             traverse_node(SubCat:Cat))))
        ;
        Type == and,
        display_message(Message),
        traverse_nodes(Node,SubCats,Category)).

traverse_nodes(_,[],[]).
```

```
traverse_nodes(Node,[SubCat|SubCats],Category) :-
        traverse_node(SubCat:Cat),
        (Cat == [] ->
             traverse_nodes(Node,SubCats,Category)
        ;
        assert_and_retract_on_backtracking(Node:SubCat),
        Category = [SubCat:Cat|Cats],
        traverse_nodes(Node,SubCats,Cats)).

check_constraints(Node,OldSubCats,SubCats) :-
        (setof(NewCats,check_constraint(Node,NewCats),SubCatList) ->
             map_intersection(SubCatList,SubCats)
        ;
        SubCats = OldSubCats).

check_constraint(Node,SubCats) :-
        constraint(Conditions,Node:SubCats),
        call(Conditions).

check_alternatives([],[]).
check_alternatives([Node|Nodes],NewList) :-
        (check_constraint(Node,[]) ->
             check_alternatives(Nodes,NewList)
        ;
        NewList = [Node|RestList],
        check_alternatives(Nodes,RestList)).

get_choice(Node,List,Choice) :-
        check_alternatives(List,NewList),
        tcl_string(Node,TclNode),
        tcl_string(NewList,TclList),
        repeat,
        tcl(´get_choice ## ##´,[TclNode,TclList],TclChoice),
        (TclChoice == "undo" ->
             !,fail
        ;
        (string(TclChoice) ->
             atom_string(Choice,TclChoice)
        ;
        (compound(TclChoice) ->
             concat_string(TclChoice,Choice)
        ;
        Choice = TclChoice))).

display_message(Message) :-
        ((Message == "" ; Message == ´´) ->
              true
        ;
        tcl_string(Message,TclMessage),
        tcl(´message  ## ´,[TclMessage])).
```

```
initialize_constraints :-
        doall((
               retract(positive_constraint(Paths,Cat)),
               unfold_paths(Paths,NewCats),
               assert(constraint(NewCats,Cat))
               ;
               retract(negative_constraint(Paths,Cat2:NegAlts)),
               node(_,Cat2,_,Alts),
               (var(NegAlts) ->
                    delete(NegAlts,Alts,PosAlts)
               ;
               subtract(Alts,NegAlts,PosAlts)
               ),
               unfold_paths(Paths,NewCats),
               assert(constraint(NewCats,Cat2:PosAlts))
             )).

unfold_paths((X,Y),(E1,E2)) :-
        unfold_paths(X,E1),
        unfold_paths(Y,E2),
        !.
unfold_paths((X;Y),(E1;E2)) :-
        unfold_paths(X,E1),
        unfold_paths(Y,E2),
        !.
unfold_paths(X:Y,X:Y) :-
        var(Y),
        !.
unfold_paths(X:Y:Z,(X:Y,R)) :-
        unfold_paths(Y:Z,R),
        !.
unfold_paths(X,X).

assert_node :-
        open(dscr,read,ID),
        read(ID,Node),
        close(ID),
        assert(Node).

init_database :-
        retract_all(node(_,_,_,_,_)),
        retract_all(constraint(_,_)),
        retract_all(scheme(_,_)).

doall(C) :- C,fail;true.

map_intersection([],[]).
map_intersection([X],X) :- !.
map_intersection([X,Y|Z],IS) :-
        intersection(X,Y,I),
        map_intersection([I|Z],IS).
```

```
assert_and_retract_on_backtracking(X) :- assert(X).
assert_and_retract_on_backtracking(X) :- retract(X),!,fail.
```

Zu guter Letzt:



Eine Kookkurrenzrestriktion besteht zwischen zwei Konstituenten einer phonologischen Struktur, genannt Relate, wobei das erste Relat links vom zweiten Relat in der phonologischen Struktur angeordnet ist. Von welcher Art ist das erste Relat?

Categories: bsp3.cat
Describer: F.J. Wanninger
Language: Niederbayerisch
Supercategory: Erstes_Relat

Alternatives

Segment
Kopf
Nukleus
Koda
Koerper
Schale
Reim
Silbe
Fuss

userinput

Pretty-printed Klassifikator

```
'Kookkurrenzrestriktion' :
   'Erstes_Relat' : 'Segment'
   'Zweites_Relat' : 'Segment'
   'Entfernung' : 'Kontakt'
   'Tauto_Heterosyllabisch' : tautosyllabisch
   'Aehnlichkeitsrichtung' :
      'Assimilation' : total
   'Wirkungsrichtung' : progressiv
```

Klassifikatoren

```
'Kookkurrenzrestriktion' : ['Erstes_Relat' : 'Kopf',
'Kookkurrenzrestriktion' : ['Erstes_Relat' : 'Kopf',
'Kookkurrenzrestriktion' : ['Erstes_Relat' : 'Kopf',
'Kookkurrenzrestriktion' : ['Erstes_Relat' : 'Nukleu
'Kookkurrenzrestriktion' : ['Erstes_Relat' : 'Nukleu
'Kookkurrenzrestriktion' : ['Erstes_Relat' : 'Nukleu
'Kookkurrenzrestriktion' : ['Erstes_Relat' : 'Nukleu
'Kookkurrenzrestriktion' : ['Erstes_Relat' : 'Segmen
```

Literatur

[Backofen, 1994] Backofen, R. (1994). Regular path expressions in feature logic. *J. Symbolic Computation*, 17:421–455.

[Backofen et al., 1995] Backofen, R., Rogers, J., and Vijay-Shanker, K. (1995). A first-order axiomatization of the theory of finite trees.

[Backofen and Smolka, 1992] Backofen, R. and Smolka, G. (1992). A complete and recursive feature theory. Technical Report RR-92-30, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH.

[Bird and Blackburn, 1991] Bird, S. and Blackburn, P. (1991). A logical approach to arabic phonology. In *Proceedings of the Fifth Meeting of the European Chapter of the Association for Computational Linguistics*, pages 89–94.

[Bird and Klein, 1994] Bird, S. and Klein, E. (1994). Phonological analysis in typed feature structures. *Computational Linguistics*, 20(3).

[Brink and Schmidt, 1992] Brink, C. and Schmidt, R. A. (1992). Subsumption computed algebraically. *Computers Math. Applic.*, 23(2-5):329–342.

[Bußmann, 1990] Bußmann, H. (1990). *Lexikon der Sprachwissenschaft, 2. Auflage.* Kröner-Verlag, Stuttgart.

[Busemann, 1993] Busemann, S. (1993). *Einführung in die künstliche Intelligenz*, chapter Sprachverarbeitung. Addison-Wesley.

[Carpenter, 1992] Carpenter, B. (1992). *The Logic of Typed Feature Structures.* Cambridge University Press.

[Comrie and Smith, 1977] Comrie, B. and Smith, N. (1977). Lingua Descriptive Studies: Questionnaire. *Lingua*, 42:1–72.

[Cremers et al., 1994] Cremers, A., Griefahn, U., and Hinze, R. (1994). *Deduktive Datenbanken. Eine Einführung aus der Sicht der logischen Programmierung.* Vieweg.

[Crystal, 1987] Crystal, D. (1987). *The Cambridge Encyclopedia of Language.* Cambridge University Press.

[Crystal, 1991] Crystal, D. (1991). *A Dictionary of Linguistics and Phonetics.* Blackwell Publishers, 3rd. edition.

[Das, 1992] Das, S. K. (1992). *Deductive Databases and Logic Programming.* Addison-Wesley.

[Dorn, 1980] Dorn, G. J. W. (1980). Klassifikation. In Speck, J., editor, *Handbuch wissenschaftstheoretischer Begriffe*, volume 2. Vandenhoek & Ruprecht.

[Dürr and Schlobinski, 1994] Dürr, M. and Schlobinski, P. (1994). *Einführung in die deskriptive Linguistik, 2. Auflage.* Westdeutscher Verlag.

[Evans and Gazdar, 1995] Evans, R. and Gazdar, G. (1995). Datr: A language for lexical knowledge representation. Technical Report cogs csrp 382, Cognitive & Computing Sciences, University of Sussex.

[Glück, 1993] Glück, H. (1993). *Metzler Lexikon Sprache.* Verlag J. B. Metzler, Stuttgart, Weimar.

[Goldsmith, 1995] Goldsmith, J. (1995). *The Handbook of Phonological Theory.* Blackwell.

[Görz, 1988] Görz, G. (1988). *Strukturanalyse natürlicher Sprache.* Addison Wesley Publishing Company, Bonn.

[Görz, 1991] Görz, G. (1991). Wissensrepräsentation und die verarbeitung natürlicher sprache. In Struß, P., editor, *Wissensrepräsentation.* Oldenburg Verlag.

[Görz, 1993] Görz, G., editor (1993). *Einführung in die künstliche Intelligenz.* Addison-Wesley.

[Johnson, 1991] Johnson, M. (1991). Features and formulae. *Computational Linguistics*, 17:131–151.

[Kaplan and Kay, 1994] Kaplan, R. M. and Kay, M. (1994). Regular models of phonological rule systems. *Computational Linguistics*, 20(3):331–378.

[Kenstowicz, 1994] Kenstowicz, M. (1994). *Phonology in Generative Grammar.* Blackwell Publishers.

[Klein, 1991] Klein, E. (1991). Phonological data types. In Bird, S., editor, *Declarative Perspectives on Phonology*, pages 127–138. University of Edinburgh.

[Krieger, 1994] Krieger, H. U. (1994). Typed feature formalisms as a common basis for linguistic specification. Technical Report RR-94-39, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH.

[Lewandowski, 1985] Lewandowski, T. (1985). *Linguistisches Wörterbuch, 4. Auflage.* Quelle & Meyer.

[Lühr, 1986] Lühr, R. (1986). *Neuhochdeutsch.* W. Fink Verlag München.

[Linke et al., 1994] Linke, A., Nussbaumer, M., and Portmann, P. R. (1994). *Studienbuch Linguistik, 2. Auflage.* Niemeyer Verlag Tübingen.

[Lyons, 1989] Lyons, J. (1989). *Einführung in die moderne Linguistik, 7. Auflage.* C. H. Beck.

[Malmkjaer, 1991] Malmkjaer, K. (1991). *The Linguistics Encyclopedia.* Routledge.

[Miller, 1995] Miller, G. A. (1995). *Wörter.* Zweitausendeins.

[Pollard and Sag, 1994] Pollard, C. and Sag, I. (1994). *Head-Driven Phrase Structure Grammar.* Studies in Contemporary Linguistics. University of Chicago Press.

[Rühl, 1970] Rühl, P. (1970). *Türkische Sprachlehre.* Julius Groos Verlag Heidelberg.

[Scherer and Wollmann, 1986] Scherer, G. and Wollmann, A. (1986). *Englische Phonetik und Phonologie.* Grundlagen der Anglistik und Amerikanistik. Erich Schmidt Verlag.

[Trask, 1996] Trask, R. L. (1996). *A Dictionary of Phonetics and Phonology.* Routledge.

[Ullman, 1988] Ullman, J. (1988). *Principles of database and knowledge-base systems*, volume 1. Computer science press.

[Ullman, 1989] Ullman, J. (1989). *Principles of database and knowledge-base systems*, volume 2. Computer Science Press.

[von Luck and Owsnicki-Klewe, 1991] von Luck, K. and Owsnicki-Klewe, B. (1991). KL-ONE: Eine Einführung. In Struß, P., editor, *Wissensrepräsentation.* Oldenburg Verlag.

[Wendt, 1972] Wendt, H. F. (1972). *Langenscheidts praktisches Lehrbuch Türkisch.* Langenscheidt.

[Wunderlich, 1976] Wunderlich, D., editor (1976). *Wissenschaftstheorie der Linguistik.* Athenäum Verlag Kronberg.

[Wunderlich, 1981] Wunderlich, D. (1981). *Grundlagen der Linguistik, 2. Auflage.* Westdeutscher Verlag Opladen.