Generierung von natürlichsprachlichen Texten aus semantischen Strukturen im Prozeß der maschinellen Übersetzung - Allgemeine Strukturen und Abbildungen

Dipl. Inf. Lutz Rosenpflanzer
Bandholzstr. 9/1
D-69207 Sandhausen/Baden
email: lutz.rosenpflanzer@sap-ag.de

Prof. Dr.-Ing. habil. Hans-Ulrich Karl
Technische Universität Dresden
Fakultät Informatik
Institut für Softwaretechnik I
D-01062 Dresden
email: huk@inf.tu-dresden.de

Inhalt

0 VORWORT 2

1 ALLGEMEINER ABLAUF DER GENERIERUNG 3

1.1 AUFGABE DER GENERIERUNG 3
1.2 EINORDNUNG DER GENERIERUNG IN DIE MASCHINELLE ÜBERSETZUNG 4
1.3 REALISIERUNG 4
1.4 MORPHOLOGISCHE GENERIERUNG 6

2 STRUKTUREN UND ABBILDUNGEN 8

2.1 UNIVERSELLE STRUKTUR: DEFINITION VON GRAPHEN 8
2.2 FORMALISIERUNG SPEZIELLER SEMANTISCHER STRUKTUREN ALS GRAPHEN 9
2.3 ABBILDUNG VON STRUKTUREN 11
2.3.1 Strukturtyperhaltende Funktionen 12
2.3.2 Strukturtypverändernde Funktionen 19
2.3.3 Komplexe Funktionen 20
2.3.4 Abbildung eines gesamten Generierungsprozesses 21

3 PROTOTYP EINES GENERIERUNGSWERKZEUGS 22

4 BEISPIEL: GENERIERUNG VON TEXTEN AUS PRÄDIKATENLOGISCHEN AUSDRÜCKEN (INKREMENTELLER ALGORITHMUS) 23

4.1 ABLAUF 23
4.2 BEISPIELE VON REGELSTRUKTUREN 27

5 ZUSAMMENFASSUNG 28

6 QUELLENVERZEICHNIS 30

0 Vorwort

Bei der maschinellen Übersetzung natürlicher Sprache dominieren mehrere Probleme. Man hat es immer mit sehr großen Datenmengen zu tun. Auch wenn man nur einen kleinen Text übersetzen will, ist diese Aufgabe in umfänglichen Kontext eingebettet, d.h. alles Wissen über Quell- und Zielsprache muß - in möglichst formalisierter Form - zur Verfügung stehen. Handelt es sich um gesprochenes Wort treten Spracherkennungs- und Sprachausgabeaufgaben sowie harte Echtzeitforderungen hinzu. Die Komplexität des Problems ist - auch unter Benutzung moderner Softwareentwicklungskonzepte - für jeden, der eine Implementation versucht, eine nicht zu unterschätzende Herausforderung.

Ansätze, die die Arbeitsprinzipien und Methoden der Informatik konsequent nutzen, stellen ihre Ergebnisse meist nur prototyisch für einen sehr kleinen Teil der Sprache -etwa eine Phrase, einen Satz bzw. mehrere Beispielsätze- heraus und folgern mehr oder weniger induktiv, daß die entwickelte Lösung auch auf die ganze Sprache erfolgreich angewendet werden kann, wenn man nur genügend „Lemminge“ hat, die nach allen Seiten ausschwärmend, die „noch notwendigen Routinearbeiten“ schnell und bienenfleißig ausführen könnten.

Produktorientierte Ansätze, die marktfähige Übersetzungshilfen für jedermann anbieten, verfolgen oftmals eine weniger aufwendige Strategie, die insbesondere heuristische Verfahren integriert. Der Grad der Formalisierung ist geringer, auch hier ist die Sprache eingeschränkt etwa auf Touristenbedürfnisse oder wissenschaftlichen Fachtext.

Letztlich erfolgversprechend scheint die erste Vorgehensweise zu sein, eine konsequente Umsetzung der Aufgabe in syntaktische und semantische Regelsysteme, die bis zu „lernenden Systemen“ führen, die z.B. in der Lage sind, teilautomatisch Regeln zu erweitern, die erforderlichen Konsistenzprüfungen auszuführen bzw. für neue Textphrasen selbsttätig Regeln zu generieren.

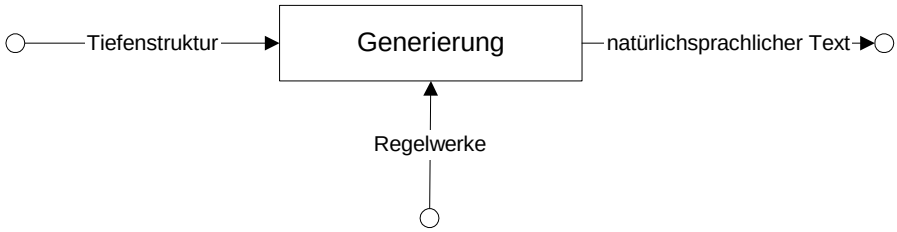
Übersetzungen bestehen immer aus einer Analyse- und einer Synthesephase. Wenn natürliche Sprachen übersetzt werden, kann die Analysephase als die aufwendigere angesehen werden. Ergebnis ist - das wird in der vorliegenden Arbeit unterlegt - ein semantisches Netz. Es wird gezeigt, in welcher Form in der Synthese- oder Generierungsphase aus einer solchen sprachunabhängigen Struktur sprachabhängige Baumstrukturen extrahiert und nach einer morphologischen Behandlung Sätze der Zielsprache aufgebaut werden können.

Die vorgeschlagene Lösung basiert auf Baum-zu-Baum-Transformationsschritten, wie sie mit Hilfe von Treetransducer formuliert werden können. Dazu werden die Eignung verschiedener Realisierungen von Treetransducern für diese Aufgabe verglichen und die erforderlichen Operationen über den verwendeten Bäumen formal beschrieben. Es wird der Prototyp eines Generierungswerkzeuges vorgestellt und anhand spezieller Beispiele die Folgerichtigkeit der Vorgehensweise demonstriert.

1 Allgemeiner Ablauf der Generierung

1.1 Aufgabe der Generierung

Die Generierung ist ein Prozeß, bei dem aus Tiefenstrukturen Texte einer natürlichen Sprache erzeugt werden. Diese Tiefenstrukturen können durch eine Analyse eines natürlichsprachlichen Textes gewonnen worden sein.



Die Tiefenstruktur ist eine formale Beschreibung des zu generierenden Textes. Um eine korrekte Generierung zu gewährleisten, muß die *Semantik* der Tiefenstruktur und die des Ergebnistextes identisch sein. Während die Tiefenstruktur Texteigenschaften mehr explizit (in der Form Eigenschaft = Wert) darstellt, sind die Eigenschaften des natürlichsprachlichen Textes nahezu ausschließlich implizit (d.h., aus dem Text kann man Eigenschaft = Wert *ableiten*, aber nicht *ablesen*) in diesem enthalten.

Die Tiefenstrukturen können unterschiedliche Abstraktionsgrade haben. Mit zunehmendem Abstraktionsgrad werden mehr Texteigenschaften explizit statt implizit im Rahmen der Struktur dargestellt.

Bei den Texteigenschaften unterscheidet man zwei wesentlich Gruppen: semantische (d.h. inhaltliche) Merkmale und syntaktische (d.h. die Darstellungsform) beschreibende Merkmale. Während die Semantik aller Strukturen gleich sein muß, ist die Syntax des Textes keine Eigenschaft dieser Strukturen, sondern der Sprache des zu generierenden Textes. Wenn die Tiefenstruktur eine rein semantische Beschreibung des Textes ist, ergeben sich die syntaktischen Merkmale als Produkt aus dieser Beschreibung und den syntaktischen Regeln der Sprache. Weil dieser Abbildungsprozeß sehr komplex ist, kann es Tiefenstrukturen geben, die bereits bestimmte wesentliche (aber nie alle) syntaktische Merkmale enthalten. Die restliche syntaktischen Merkmale ergeben sich dann wieder aus dem bestehenden Merkmalen der Tiefenstruktur und der allgemeinen Syntax der natürlichen Sprache.

Die Generierung selbst ist ein Abbildungsprozeß, bei dem aus der eingehenden Tiefenstruktur und den Regeln natürlichsprachlicher Text produziert wird. Die Generierung wird als abschließende Phase in der maschinellen Übersetzung benutzt, aber auch für Ausgabekomponenten eines natürlichsprachlichen Mensch-Maschine-Schnittstellen z.B, um Ergebnisse eine Datenbankanfrage natürlichsprachlich auszugeben.

1.2 Einordnung der Generierung in die maschinelle Übersetzung

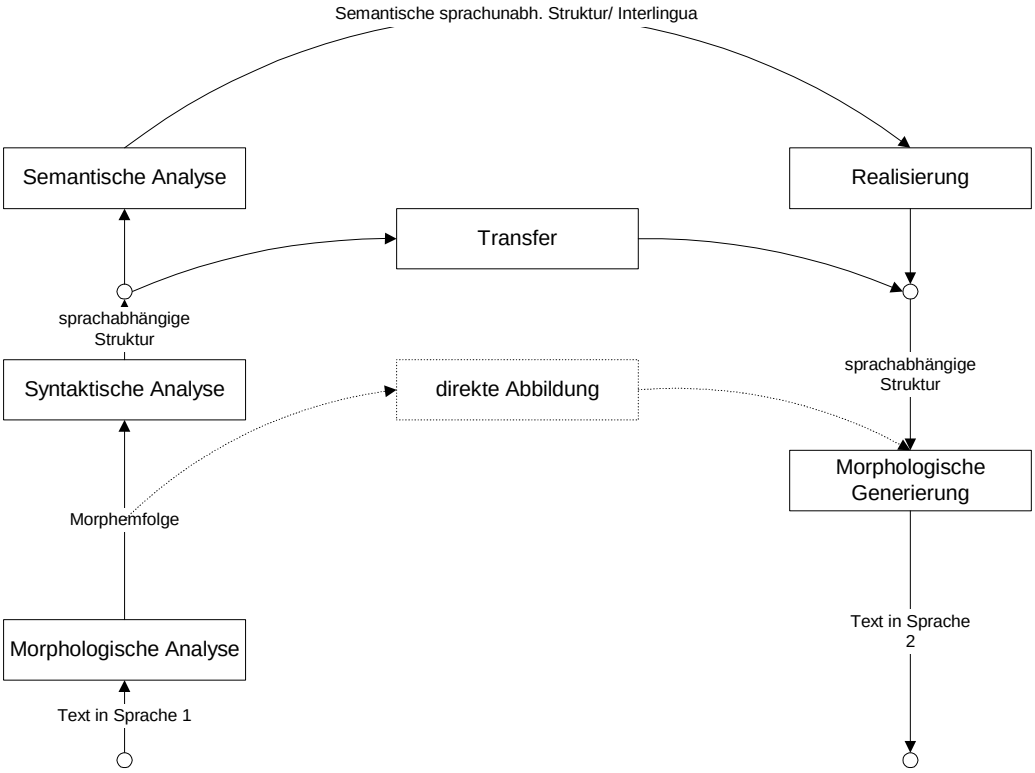


Abbildung 1 Die semantische Übersetzung im Modell

Die maschinelle Übersetzung ist ein Prozeß, der in mehreren aufwendigen Schritten abläuft. Diese Schritte kann man in Analyse- und in Generierungs(Synthese-)schritte unterteilen. Bei den Analyseschritten versucht man, die Tiefenstruktur (den Inhalt) zu erkennen. Man unterscheidet:

- morphologische Analyse (Erkennung der Morpheme aus dem Text)
- syntaktische Analyse (Bestimmung einer sprachabhängigen Baumstruktur)
- semantische Analyse (Bestimmung einer semantischen sprachunabhängigen Struktur [*Interlingua-Ansatz*])

Außer beim Interlingua-Ansatz muß ein Transfer sprachabhängiger Strukturen durchgeführt werden.

Bei den Syntheseschritten sollen Oberflächenstrukturen (Text) generiert werden. Man unterscheidet zwei wesentliche Schritte:

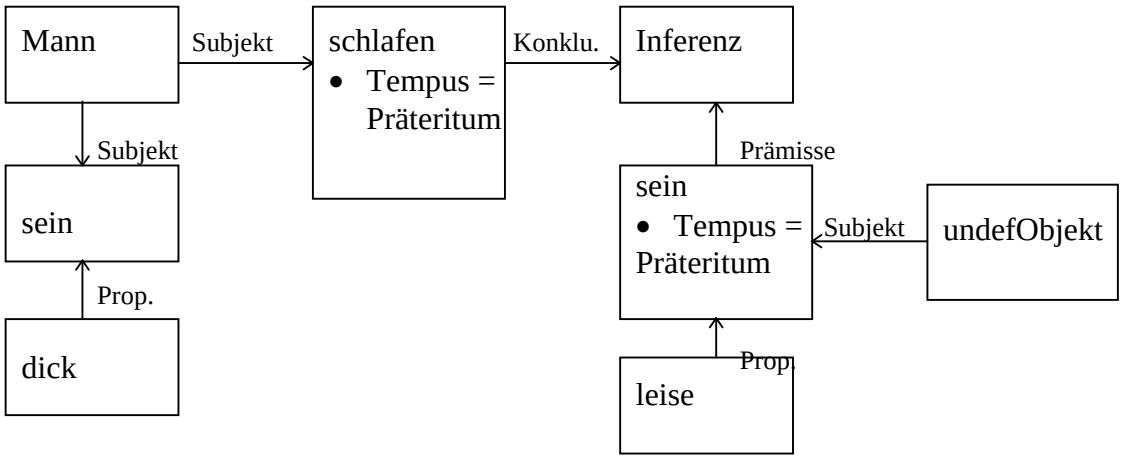
- Realisierung (Generierung einer sprachabhängigen Struktur)
- Morphologische Generierung (Texterzeugung)

1.3 Realisierung

Die Realisierung hat die Aufgabe, aus der sprachunabhängigen semantischen Struktur eine sprachabhängige Struktur[1] zu erzeugen. Diese enthält bereits den Satzrahmen, Wortradikale und grundlegende Textattribute wie Tempus, Modus etc.

[1] synonymer Begriff: rhetorische Struktur

Als Illustration möge folgendes semantisches Netz[2] dienen:



Dieses Netz könnte den Satz: Der dicke Mann schlief, weil es leise war. erzeugen.
Die Verfahren zur Generierung gehen aus linguistischer Sicht unterschiedlich vor. In allen Fällen erfolgt jedoch die Abbildung der netzartigen Struktur in eine einfachere Baumstruktur (z.B. Konstituenz- oder Dependenzbaum). Die Abbildung aus funktionaler Sicht muß bestimmte formale Unterschiede zwischen Bäumen und Graphen überwinden (siehe Kap. 2.3.2).
Um aus einem Graphen einen Baum zu erzeugen, muß man die Wurzelknotenzahl und die Vorgänger eines Knotens auf eins begrenzen. Die Reduzierung der Wurzelknoten erfolgt durch die Bildung *minimaler Graphen* aus dem Ursprungsgraphen.
Durch diese Minimalisierung entsteht oft bereits aus der vernetzten Struktur eine Baumstruktur.
Wenn das nicht der Fall ist, muß die Zahl der Vorgänger eines Knotens auf maximal einen reduziert werden, indem der *isomorphe Baum* des Graphen bestimmt wird.
Alle Verfahren erzeugen jedoch nicht nur formal aus dem Netz eine Baumstruktur, sondern es werden bestimmte Teile der Struktur durch andere Teile ersetzt. Dieses wird als (Tree)-Transducing bezeichnet. Immer benötigt man dafür ein (unterschiedlich notiertes) Regelwerk, bei dem Knoten oder ganze Teilbäumen durch andere ersetzt werden.
Bei vielen Verfahren wird beim Treetransducing dieser sprachabhängige Baum top-down bearbeitet.
Als Beispiel sei eine Regel eines Ersetzungs- (Rewrite-) Regelwerkes definiert und die Ersetzung demonstriert:

Element	Regel (Quellteil)	Regel (Zielteil)
Inferenz	Inferenz *Konklusion* *Prämisse* A B	weil A B

Abbildung 2 Beispiel für eine Ersetzungsregel[3]

[2] Die semantischen Symbole sind sprachunabhängig, zum besseren Verständnis wurden allerdings deutsche Bezeichnungen gewählt.

[3] A und B sind die Variablen für die Unterstrukturen.

Nachdem derartige Regeln am obigen Beispiel angewendet wurden, könnte man folgende Dependenzbaumstruktur erhalten:

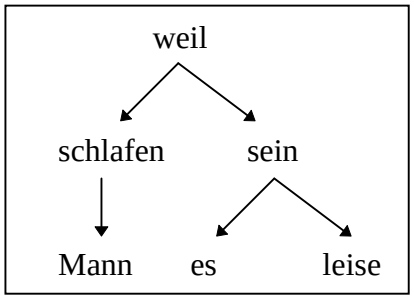


Abbildung 3 Dependenzbaum, realisiert aus dem semantischen Netz

Die Referenzen, d.h. die Elemente, die in mehreren Teilnetzen gleichzeitig auftreten, bleiben z.B. über entsprechende Attribute erhalten.
Andere Regelwerke sind auf der Grundlage von Entscheidungsnetzen oder von Unifikationsgraphen notiert. [Pangl],[Bat90]

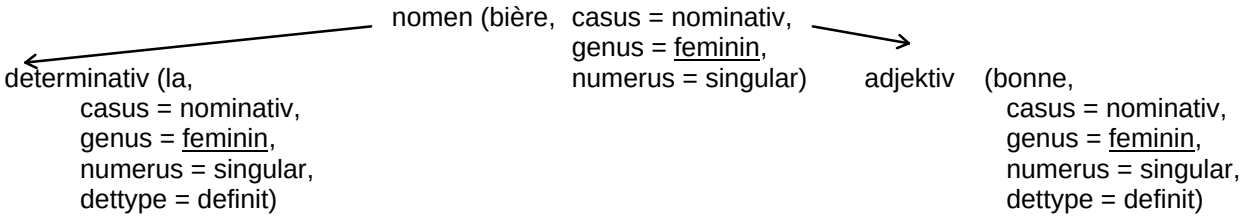
1.4 Morphologische Generierung

Die morphologische Generierung[4] hat die Aufgabe, aus dem Dependenzbaum den konkreten Text zu bestimmen. Folgende Probleme sind dabei zu lösen:

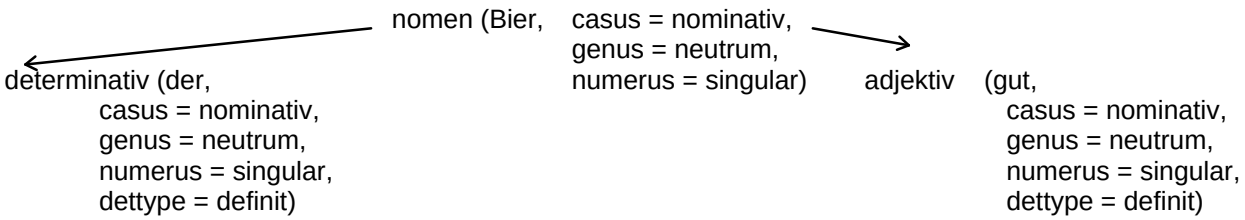
- Festlegung der Wortformen bei flektierenden Wortarten über ein Lexikon
- Festlegung der Satzstellung der Satzglieder:
  Die Position der Satzglieder richtet sich nach allgemeinen Folgeregeln, im Deutschen aber auch nach der Betonung von Textteilen. Betonte Textteile befinden sich im Deutschen entweder im Vorfeld, d.h. am Anfang des Satzes, oder am Ende im Nachfeld. Die Position von Wörtern in Phrasen ist dagegen kaum variabel.

Beispiel zur Illustration:

Bei der Übersetzung der französischen Phrase la bière bonne

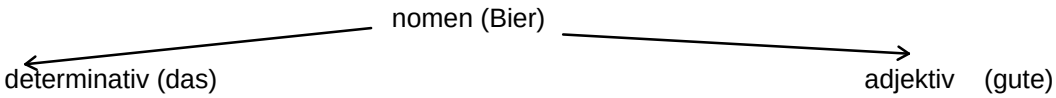


ins Deutsche das gute Bier erhält man nach der Realisierung, der Übernahme der Attributierung der Lexikoneinträge und der Attributverteilung folgenden Dependenzbaum:



[4] synonyme Begriffe: Satzgenerierung, Oberflächenrealisierung

Nun können über ein Lexikon die Wortformen bestimmt werden:



Durch die Konstituenzregel NP ⇒ det adj n erhält man schließlich folgenden Text:

das gute Bier

Die meisten Verfahren bearbeiten die Inputstruktur top-down knotenweise von der Wurzel zu den Blättern. Die Attribute des Baumes, die bei der Wortformenerzeugung insbesondere von Bedeutung sind, werden bei fast allen Verfahren zunächst im Dependenzbaum verteilt. (z.B.: vererbt bzw. unifiziert.) Meist stehen die Attribute nur an den Wurzeln ihrer Herkunft zur Verfügung[5].

Wortform

Die Festlegung der Wortform erfolgt mit Hilfe eines Lexikons (und meist zusätzlicher Bildungsregeln). Das Lexikon enthält für alle Worte und Attributausprägungen die Wortform oder eine Vorschrift zur Erzeugung der Wortform. Dadurch, daß die Zahl der Worte einer Sprache in der Größenordnung 10^4 und die der Wortformen in der Größenordnung 10^5 liegt, ist es notwendig, bei ernsthaften Anwendungen das Lexikon in professionellen Datenbanken zu speichern.

Formal werden einem Paar aus Grundform (Radikal) des Wortes und einem Attributvektor eine Wortform zugeordnet, z.B.:

- (gut, {genus=feminin, numerus=singular,casus=dativ,dettype=definiert}) ⇒ besseren
- (lauf, {modus=präteritum, numerus=singular, Person=3}) ⇒ lief
- (d, {genus=feminin, numerus=singular,casus=dativ}) ⇒ der

Satzstellung der Satzglieder:

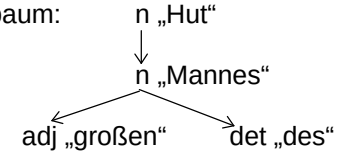
Bei manchen Verfahren ist die Inputstruktur bereits ein geordneter Baum. Wenn das nicht der Fall ist, benötigt man ein Ordnungsmerkmal, ein Positionsattribut oder ein Regelwerk. Dieses Regelwerk befähigt einen Treetransducer, aus einem ungeordneten Baum einen geordneten zu erzeugen.

Bei einer einfachen möglichen Realisierung mit Knoten-Subknoten-Positionsregeln wird der Baum top-down bearbeitet. Für jeden Knoten gibt es mindestens eine Regel, die angibt, in welcher Reihenfolge seine (ggf. vorhandenen) Subknoten (oder ganze Subbäume) und der Knoten selbst (sog. Head) stehen sollen. Dadurch wird der gesamte Baum in eine Reihenfolge gebracht. ([Sme91]).

Beispiel (Attribute oder Bedingungen sind hier nicht vorhanden):

Regelmenge: { (n, det adj *head* n), (det, *head*), (adj, *head*)}

Dependenzbaum:



[5] Z.B. in Nominalphrasen steht der Kasus und Genus zunächst nur am (Wurzel-)Nomen und muß noch zu den ggf. vorhandenen Komplementen (Adjektiv, Determinativ, ...) übertragen werden.

Baumordnen[6]: Die Unterbäume eines jeden Knotens im Dependenzbaum werden geordnet.

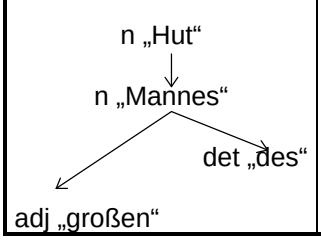
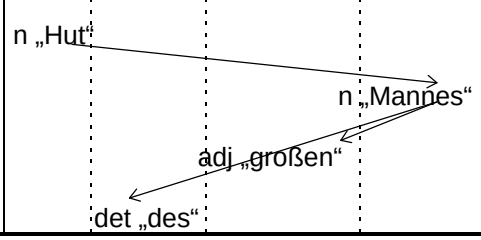
Dependenzbaum	angewandte Regel	geordneter Dependenzbaum
	~~(det adj~~ „Hut“ n) (det adj „Mannes“ ~~n~~) („des“) („großen“)	

Tabelle 1 Ordnung des Dependenzbaums

Der Dependenzbaum zeigt in seiner linearen Darstellung die folgende Wortordnung:
(„Hut“ ((„des“) („großen“) „Mannes“)) = „Hut des großen Mannes“.

Obwohl die Wörter an sich bereits in der Realisierung zu definieren sind, kann es notwendig sein, den Prozeß der Wortbearbeitung z.T. erst in der Morphologischen Generierung durchzuführen. Bestimmte syntaktische Eigenschaften (z.B. der Tempus) haben Einfluß auf die Wortbildung (z.B. sind separable Verbpräfixe unter bestimmten Bedingungen abzutrennen). Erst in dieser Phase steht ggf. alle Information zur Verfügung, um letzte Struktur- und Wortänderungen vorzunehmen.

2 Strukturen und Abbildungen

Alle Verfahren zur Generierung in der maschinellen Übersetzung kann man dahingehend verallgemeinern, daß aus einer Inputstruktur funktional eine Outputstruktur erzeugt wird.
Die Inputstruktur ist netzartig und wird über Transformationen erst in eine Baum- und dann in eine lineare Struktur gebracht. Daher soll hier ein Instrumentarium an Funktionen und an Strukturen definiert werden. Die Generierung ist dann eine Mehrfachanwendung bestimmter Funktionen auf diese Strukturen, insbesondere auf die Inputstruktur.

2.1 Universelle Struktur: Definition von Graphen

An dieser Stelle soll versucht werden, die gemeinsamen Merkmale aller aus natürlicher Sprache stammenden Strukturen zu benennen.

1. Man stellt als erstes fest, daß es sich immer um eine *Datenstruktur* handelt. Das heißt, es werden Beziehungen zwischen speziellen Entitäten festgelegt, die für einen Zeitpunkt gelten und für diesen invariant sind. (Fotoprinzip)
2. In allen Strukturen existieren *atomare Elemente (Entitäten)*, die über *Relationen* in Beziehung stehen. (z.B.: Entitäten: Wort, Lexemknoten, semantisches Objekt; Relation: Nachfolgerrelation, syntaktische Beziehung, semantische Beziehung)
3. Alle aus sprachlichen Äußerungen resultierenden semantischen Strukturen sind *endlich.*

Diese Eigenschaften treffen auf Graphen zu, so daß o.B.d.A. angenommen werden kann, daß Graphen geeignet sind, alle semantischen Strukturen formal darzustellen.
Nun sollen Graphen formal definiert werden, und spezielle semantische Strukturen als Graph dargestellt werden.

[6] Durchgestrichenen Konstituenten entfallen, wenn sie nicht auftreten.

Ein endlicher bewerteter Graph $G = (V, E, f_V, f_e)$ ist wie folgt definiert:

- V ist die nichtleere endliche Knotenmenge,
- E ist die Adjazenzrelation, vom Typ V x V also V^2,
- f_V ist die Knotenbewertungsfunktion, eine Abbildung von V in eine nicht weiter definierte Menge,
- f_e ist die Kantenbewertungsfunktion, eine Abbildung von E in eine nicht weiter definierte Menge.

Spezielle Eigenschaften von Graphen

1. azyklischer Graph: azyklischer Graph: (informal) Ein Graph ist azyklisch, wenn es keinen Weg durch den Graphen gibt, bei dem ein Knoten mehr als einmal berührt wird
2. (ungeordneter) Baum: Ein azyklischer Graph G heißt Baum, wenn gilt: Es gibt genau ein $v_w \in V$ für das kein $v_i \in V$ angegeben werden kann, für das gilt: $(v_i , v_w) \in E$, v_w heißt Wurzelknoten. Für alle $v_j \in V$, mit j <> w gilt: Es gibt genau ein $v_j \in V$, so daß $(v_i , v_j) \in E$, v_i heißt Superknoten von v_j. G ist zusammenhängend.
3. geordneter Baum: Ein Graph G heißt geordneter Baum, wenn gilt: G ist ein Baum. Es gilt eine Ordungsrelation $O_j(v_i , v_k)$ für jeden Superknoten v_j , so daß für zwei beliebige v_i ,v_k mit $(v_j , v_i) \in E$ und $(v_j , v_k) \in E$ gilt: Wenn nicht $O_j(v_i , v_k)$, dann $O_j(v_k , v_i)$. Zu $O_j(v_k , v_i)$ sagt man, " v_k ist links von v_i".
4. Folge: Ein Graph G heißt Folge, wenn gilt: G ist ein Baum und es gibt genau ein $v_e \in V$ für das kein $v_i \in V$ angegeben werden kann, für das gilt: $(v_e , v_i) \in E$, v_e heißt letzter Knoten. Für alle $v_j \in V$, mit j <> e gilt: Es gibt genau ein $v_i \in V$, so daß $(v_j , v_i) \in E$, v_i heißt Nachfolgerknoten von v_j.

2.2 Formalisierung spezieller semantischer Strukturen als Graphen

Nachdem Graphen und deren Eigenschaften formal definiert wurden, sollen nun bekannte semantische Strukturen damit klassifiziert werden.

1. Wortsequenz:
- Eine Wortsequenz ist eine Folge.
- f_v heißt Wortzuweisungsfunktion (jeder Knoten ist mit einem Wort beschriftet).
- f_e ist leer.

2. Morphemsequenz:
- Eine Morphemsequenz ist eine Folge.
- f_v heißt Morphemzuweisungsfunktion (jeder Knoten ist mit einem Tupel aus dem Morphem und dessen Attributen beschriftet).
- f_e ist leer.

3. Konstituenzbaum:
- Ein Konstituenzbaum ist ein geordneter Baum.
- f_v heißt Vokabularzeichenzuweisungsfunktion (jeder Knoten ist entweder mit einem Alphabetzeichen oder einem Nichtterminalzeichen und Attributen beschriftet).
- f_e ist leer.

4. Dependenzbaum:
- Ein Dependenzbaum ist ein (ungeordneter) Baum.
- f_v heißt Morphemzuweisungsfunktion (jeder Knoten ist mit einem Morphem und Attributen beschriftet).
- f_e ist die semantische Abhängigkeitsrelation

5. semantisches Netz:
- Ein semantisches Netz ist ein gerichteter Graph mit folgenden Eigenschaften:
- f_V ist die Knotenbewertungsfunktion, eine Abbildung von V in die Menge der Attribut/Wertpaare $O = N \times F = \{o_i\} = \{(n_i, f_i)\}$.
- N ist die Bezeichnung,
- F ist der Wert
- f_e ist die Relationenbewertungsfunktion, eine Abbildung von V in die Menge der Attribut/Wertpaare, eine Abbildung von V x V nach O

6. Frames:

Frames sind eine andere Darstellung für semantische Netze. Ein Frame besteht aus einem Kopf und Slots. Die Slots sind wieder mit Frames belegt und haben einen Typ. Frames ohne Slots sind Blattframes. Wenn mehrere Slots mit einunddemselben Frame belegt sind, liegt eine Netzstruktur, sonst nur eine Baumstruktur vor.

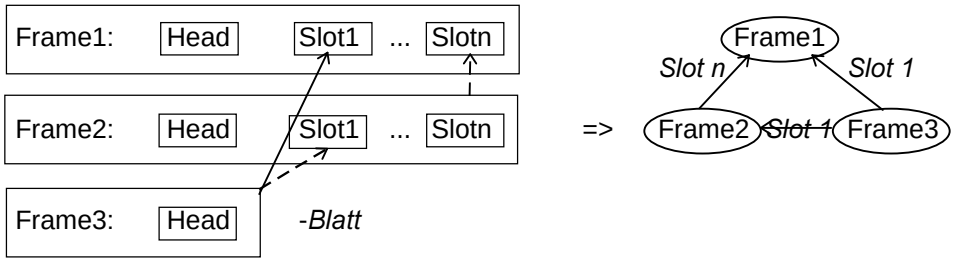
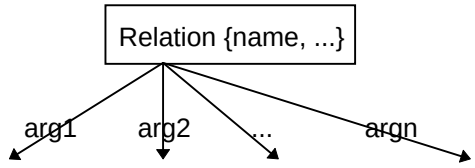
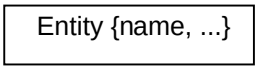


Abbildung 4 Struktur von Frames, Darstellung als Frame und als Netz

7. Prädikatenlogische Ausdrücke:
Auch prädikatenlogische Ausdrücke können als Graph dargestellt werden.
Für die Transformation eines solchen Ausdrucks ist der Wirkungsbereich der Quantoren entscheidend. Jedem Quantor ist sein zugehöriger Ausdruck als Teilgraph im Graphen untergeordnet. Die Schachtelung von Ausdrücken führt zur Schachtelung von Teilgraphen.
Aussagenlogische Formeln, Funktoren und Quantoren sind mathematisch gesehen Relationen (sie können nicht ohne zugeordnete Objekte existieren), die durch Teilgraphen der Form:



repräsentiert werden.
Konstanten und Variablen sind unabhängige Objekte, die durch Knoten der Form:



repräsentiert werden können.

Beispiel:
$\forall a . (\exists b. ((person(a) \wedge person(b)) \Rightarrow vater(a, b)))$

Dieser Ausdruck mit der Semantik „Jeder hat einen Vater“ wird durch folgenden Graphen repräsentiert:

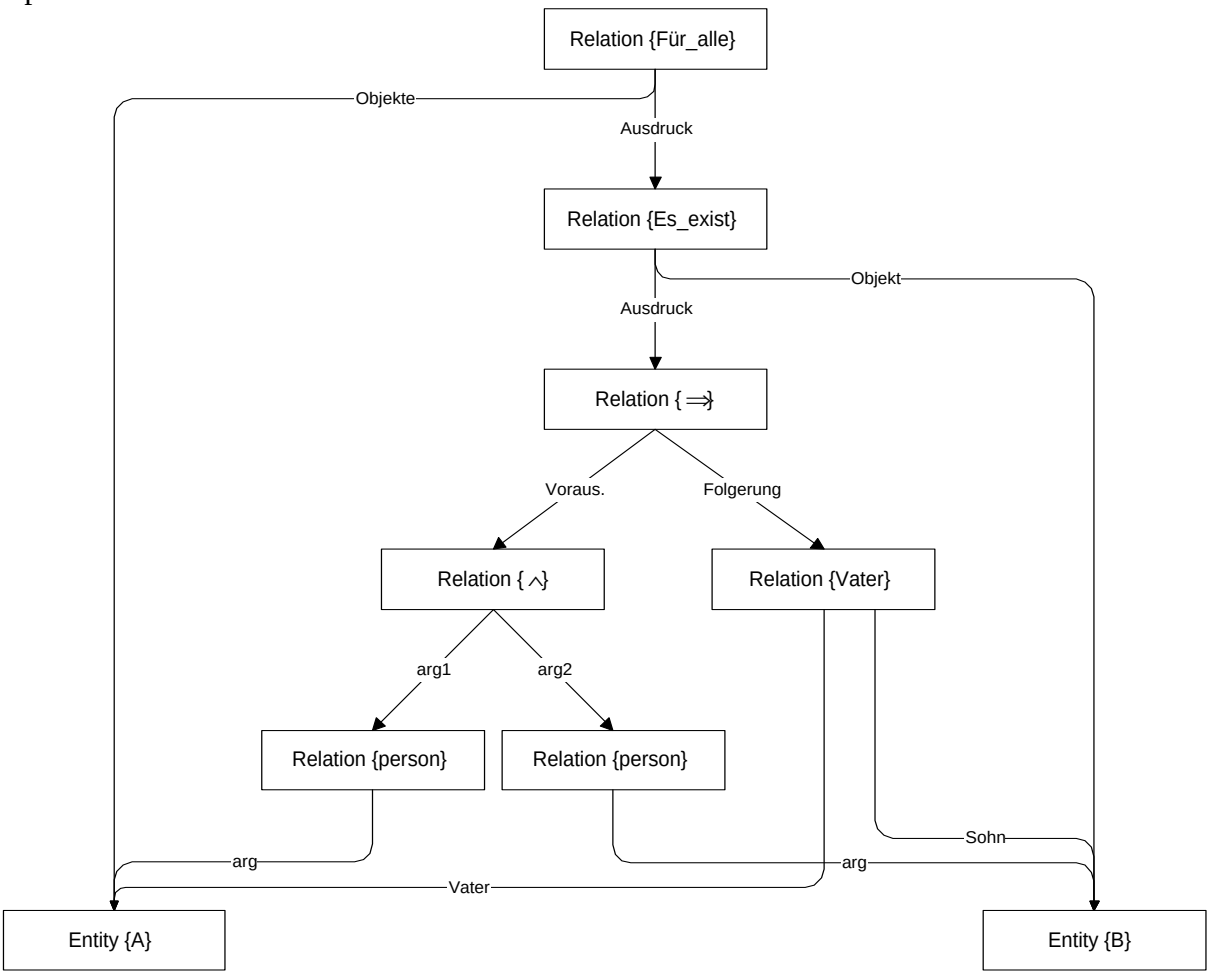


Abbildung 5 Graph eines prädikatenlogischen Ausdrucks

2.3 Abbildung von Strukturen

Bei der Generierung können die Input- und Outputstrukturen typisiert werden. Weiterhin kann die Zahl der Inputparameter betrachtet werden.
Im folgenden wird eine Matrix aller Strukturen der Verfahren zur Generierung vorgestellt. Durch diese Matrix wird die Menge der (theoretisch) möglichen unären Abbildungen beschrieben. Einige Abbildungen finden nur im Parsing, andere nur in der Generierung und weitere gemischt Anwendung.

von : nach	1-dim (Text, Morphemsequenz)	2-dim (Syntax-, Dependenzbaum)	3-dim (Graph)
1-dim (Text, Morphemsequenz)	Morphologische Analyse, Textbildung	Syntaktische Analyse	?
2-dim (Syntax-, Dependenzbaum)	Morphologische Generierung	Treetransducing	Semantische Analyse
3-dim (Graph)	?	Realisierung	Graphentransducing/ Unifikation

Tabelle 2 Matrix der Abbildungen

Die Generierung kann als ein Pfad durch obige Matrix angesehen werden, ebenso das Parsing.
Wenn statt einem zwei Inputparameter benutzt werden, entsteht aus der Matrix ein Würfel aller möglichen binären Abbildungen[7].
Funktionen, die neben der (komplexen) Inputstruktur noch einen weiteren (konstanten) Parameter wie z.B. ein Regelwerk haben, zählen nicht zur Menge der binären Abbildungen. Formal gesehen, kann der Parameter weggelassen werden, wenn man ihn mit der binären Abbildung zu einer neuen unären Abbildung vereinigt :

f: X × P -> Y wird zu f': X -> Y

2.3.1 Strukturtyperhaltende Funktionen

2.3.1.1 Abbildungen von Graphen

- Erzeugen minimaler Graphen

Hier werden alle Wurzelkoten K_r des Graphen G selektiert und dann durch die dadurch beschriebenen Teilgraphen die Menge **G** minimaler Graphen G_i definiert. Diese Funktion hat daher folgende abstrakte Struktur:

f_{GGM}: $G \Rightarrow \{G_1, G_2, ..., G_n\} = \mathbf{G}$

- Eine weitere Abbildung ist die hier nur kurz erwähnte Graphenunifikation:
Die Unifikation ist eine Operation, bei der zwei Strukturen zu einer Ergebnisstruktur vereinigt werden. Ggf. ist diese Struktur leer, d.h. die Unifikation ist fehlgeschlagen.

2.3.1.2 Abbildungen von Baumstrukturen

Abbildungen von Baumstrukturen heißen *Treetransducer*. Treetransducer können auf unterschiedliche Weise realisiert sein.

Abstrakte Struktur von Treetransducern:

f_{TT}: $T \Rightarrow T'$

Die Treetransducer können auf sehr unterschiedliche Art und Weise arbeiten. Beim Studium der Literatur stellt man fest, daß man drei wesentliche Arten unterscheiden muß:

[7] Aus dieser Menge sind die Klassen T x T -> T und G x G -> G am bedeutendsten. Instanzen dieser Klasse sind die Baumkomposition und die Unifikation zweier (nicht konstanter) Strukturen.

Regelbasierte Treetransducer
Prinzip: Wenn in dem Quellbaum T^Q ein Teilbaum T' enthalten ist und eine Regel r(T', T'') existiert, soll T'' im Zielbaum T^Z eingefügt werden, es entspricht T' im Baum T^Q dem Baum T'' im Baum T^Z.

Unifikationsbasierte Treetransducer
Prinzip: Wenn in dem Quellbaum T^Q ein Teilbaum T' enthalten ist und eine mit diesem Teilbaum unifizierbare Regel r(T'') existiert, soll T'' im Zielbaum T^Z eingefügt werden, es entspricht T' im Baum T^Q dem Teilbaum T'' im Baum T^Z.

entscheidungsnetzbasierte Treetransducer
Prinzip: Wenn in dem Regelbaum T^R eine Alternative T'_r enthalten ist, deren Entscheidungsausdruck (Chooser) den Wert TRUE bezogen auf die Position p^Q im Quellbaum T^Q liefert, dann wird die mit der Alternative verknüpfte Aktion (Successor) P^{Rr} bezogen auf den bisherigen Zielbaum T^Z ausgeführt. Diese Treetransducer zeichnen sich durch ihre Dezentralität aus; fast alle Konstruktionsschritte werden durch spezielle Funktionen einer Regel durchgeführt. (wie bei konzeptuellen Graphen [Sow81])

Diese drei Klassen von Treetransducern verfolgen zunächst sehr unterschiedliche Ansätze. Letztlich ist der Prozeß des Treetransducing jedoch immer ein (i.allg. nichtdeterministischer) Konstruktionsprozeß bestehend aus drei wesentlichen Situationen: [Wed88]

- Startsituation: leerer Zielbaum, vollständig unbearbeiteter Quellbaum, Startposition im Quellbaum und im Regelwerk
- Konstruktionssituation: teilkonstruierter Zielbaum, teilbearbeiter Quellbaum, Position ungleich Start und Ende im Quellbaum und im Regelwerk
- Endsituation: fertiger Zielbaum, vollständig verarbeiteter Quellbaum, Endposition im Quellbaum und im Regelwerk

Der Konstruktionsprozeß ist daher beschreibbar über ein den Status definierendes Wertetupel:

$s(W^1, W^2, W^3, ..., W^n)$

und n Übergangsfunktionen, die jedem dieser Werte die Folgewerte zuweisen:

$f_1(W^1_i, W^2_i, ..., W^n_i,) = W^1_{i+1}$;
$f_2(W^1_i, W^2_i, ..., W^n_i,) = W^2_{i+1}$;
....
$f_n(W^1_i, W^2_i, ..., W^n_i,) = W^n_{i+1}$;

Im allg. sind diese Übergangsfunktionen von allen Werten abhängig.

Jetzt sollen diese Werte konkret definiert werden. Für alle drei Verfahrensklassen genügen die folgenden Werte:

- p^Q: Position im Quellbaum T^Q
- M^Q: Menge aller bereits bearbeiteten Knoten des Quellbaumes T^Q

- p^R: Position bzw. Regel im Regelwerk R
- T^Z: Zielbaum
- R^{QZ}: eindeutige Zuordnung Position im Quellbaum T^Q zu Position im Zielbaum T^Z

Die Situation nach j Bearbeitungsschritten ist dann beschrieben durch:

$s_j(p^Q_j, M^Q_j, p^R_j, T^Z_j, R^{QZ}_j)$

speziell die Startsituation: $s_0(p^Q_0, M^Q_0 = \varnothing, p^R_0, T^Z_0 = T_\varnothing, R^{QZ}_0 = \varnothing)$ [8]
und die Endsituation: $s_E(p^Q_E, M^Q_E = I, p^R_E, T^Z_E = T^Z, R^{QZ}_E)$ [9]

Für diese fünf Werte müssen nun fünf Übergangsfunktionen beschrieben werden:

- $next_{pR}$: Folgeposition bzw. -regel im Regelwerk R
  $p^R_{j+1} = next_{pR}(p^R_j, M^Q_j, p^Q_j)$
  Die nächste auszuwählende Regel(-position) hängt ab von der vorhergehenden und den bereits insgesamt bearbeiteten Quellknoten und der Folgeposition im Input.

- $next_{MQ}$: Folgemenge aller bereits bearbeiteten Knoten des Quellbaumes T^Q:
  $M^Q_{j+1} = next_{MQ}(M^Q_j, p^R_{j+1}) = M^Q_j \cup next_{MI\,j}'(p^R_{j+1})$
  Die Menge der bereits bearbeitetenden Knoten des Quellbaums wird erhöht um die Menge der durch die aktuelle Regel betroffenden Knoten des Quellbaumes.

- $next_{TZ}$: Folgezielbaum
  $T^Z_{j+1} = next_{TZ}(T^Z_j, p^R_{j+1}, p^Q_{j+1})$
  Der Folgezielbaum hängt ab vom vorherigen Zielbaum, von der vorhergehenden Regel(-position) und der vorhergehenden Position im Quellbaum.

- $next_{RQZ}$: Folgeabbildungsmenge
  $R^{QZ}_{j+1} = next_{RQZ}(p^R_{j+1}, p^Q_j, R^{QZ}_j) = R^{QZ}_j \cup next_{RQZ'}(p^R_{j+1}, p^Q_j)$
  Die Abbildungsmenge wird um die durch die aktuelle Regel betroffenen Quellbaumteile und die erzeugten neuen Teile des Zielbaumes erhöht.

- $next_{pQ}$: Folgeposition im Quellbaum T^Q:
  $p^Q_{j+1} = next_{pQ}(p^Q_j, M^Q_{j+1}, p^R_{j+1})$
  Der nächste auszuwählende Quellbaumknoten hängt ab vom vorhergehenden und den bereits insgesamt bearbeiteten Quellknoten und von der aktuellen Regel-(position).

Diese Übergangsfunktionen müssen beim Abbildungsprozeß in bestimmter Reihenfolge ausgeführt werden. Es gelten folgende Präzedenzen beim Prozeß der Abbildung:
1. Bestimmung der Regelposition
2. Konstruktion des Folgezielbaumes, der Menge des bearbeiteten Inputs, der Zuordnungsfunktion
3. Bestimmung der nächsten Quellposition
4. (Gehe zu 1.)

[8] $T_\varnothing$ ist der leere Baum.
[9] I ist die Menge aller Knoten des Quellbaumes.

Formal: $next_{pR} \subset \begin{bmatrix} next_{MQ} \\ next_{TZ} \\ next_{RQZ} \end{bmatrix} \subset next_{pQ}$

Diese Übergangsfunktionen sind für die drei Treetransducerklassen unterschiedlich ausgeprägt.

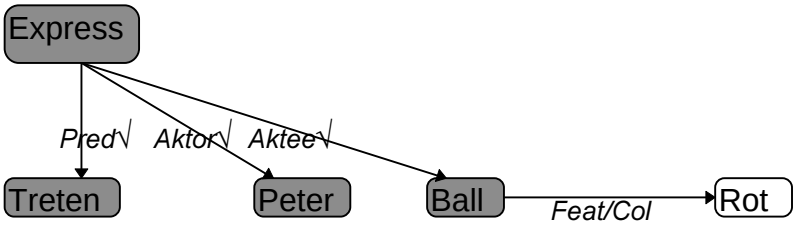
An dieser Stelle soll nur der regelbasierte Treetransducer und der entscheidungsnetzbasierte Treetransducer genauer betrachtet werden:

Funktion	regelbasierter Treetransducer	entscheidungsnetzbasierter Treetransducer
$next_{pQ}$: Folgeposition im Quellbaum:	$p^Q_{j+1} = next_{pQ}(p^Q_j, M^Q_{j+1}, p^R_{j+1}) = sel(T^Q - M^Q_{j+1})$[10] Der nächste auszuwählende Quellbaumknoten ist ein beliebiger bisher unbearbeiteter Knoten.	$p^Q_{j+1} = next_{pQ}(p^Q_j, M^Q_{j+1}, p^R_{j+1})$ Der nächste auszuwählende Quellbaumknoten wird von den Successoren der aktuellen Regel bestimmt.
$next_{MI}$: Folgemenge aller bereits bearbeiteten Knoten des Quellbaumes T^Q:	$M^Q_{j+1} = M^Q_j \cup next_{MI\,j}'(p^R_{j+1}) = M^Q_j \cup f_{quell}(p^R_{j+1})$ Die Menge der bereits bearbeiteten Knoten des Quellbaums wird erhöht um die Menge der in der aktuellen Regel enthaltenen Quellknoten.	wie regelbasierter TT
$next_{pR}$: Folgeposition bzw. -regel im Regelwerk R	$p^R_{j+1} = next_{pR}(p^R_j, M^Q_j, p^Q) = map(source(p^Q), r) \wedge r = sel(R)$ Die nächste auszuwählende Regel ist eine beliebige auf die aktuelle Quellbaumposition passende Regel.	$p^R_{j+1} = next_{pR}(p^R_j, M^Q_j, p^Q_j) = map(p^Q_j, next(p^R_j))$ Die nächste auszuwählende Regelposition ist einer der Folgeknoten des aktuellen Knotens eine auf die aktuelle Quellbaumposition passende Regel (Checker = TRUE).
$next_{TZ}$: Folgezielbaum	$T^Z_{j+1} = next_{TZ}(T^Z_j, p^R_{j+1}, p^Q_{j+1}) = build(target(p^R_{j+1}), T^Z_j, f_{map}(p^Q_{j+1}))$ Der Folgezielbaum hängt ab vom vorherigen Zielbaum, von der vorhergehenden Regel und der vorhergehenden Position im Quellbaum, zu der die korrespondierende Position im Zielbaum bestimmt wird.	$next_{TZ}$: Folgezielbaum $T^Z_{j+1} = next_{TZ}(T^Z_j, p^R_{j+1}, p^Q_{j+1}) = build(p^R_{j+1}, T^Z_j, f_{map}(p^Q_{j+1}))$ Der Folgezielbaum wird von den Successoren der aktuellen Regel bestimmt, hängt ab vom vorherigen Zielbaum und der vorhergehenden Position im Quellbaum, zu der die korrespondierende Position im Zielbaum bestimmt wird.
$next_{RQZ}$:	$R^{QZ}_{j+1} = R^{QZ}_j \cup next_{RQZ}(p^R_{j+1},$	wie regelbasierter TT

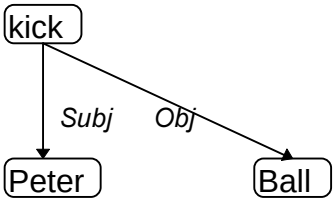
[10] Die Funktion **sel** selektiert nichtdeterministisch ein beliebiges Element aus einer Menge.

Folgeabbildungs-menge	p^{Q}_{j}) Die Abbildungsmenge wird um die durch die aktuelle Regel betroffenen Quellbaumteile und die erzeugten neuen Teile des Zielbaumes erhöht.	

Beispiel Konstruktionsschritt regelbasierter TT „Dependenzbaum aus semantischem Netz“:
Quellbaum (die grauen bzw. gehakten Teile sind bereits bearbeitet (Menge M^{Q})) :

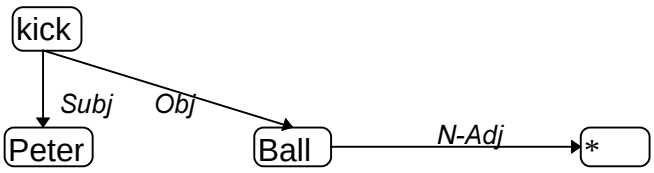


Zielbaum/aktuell:



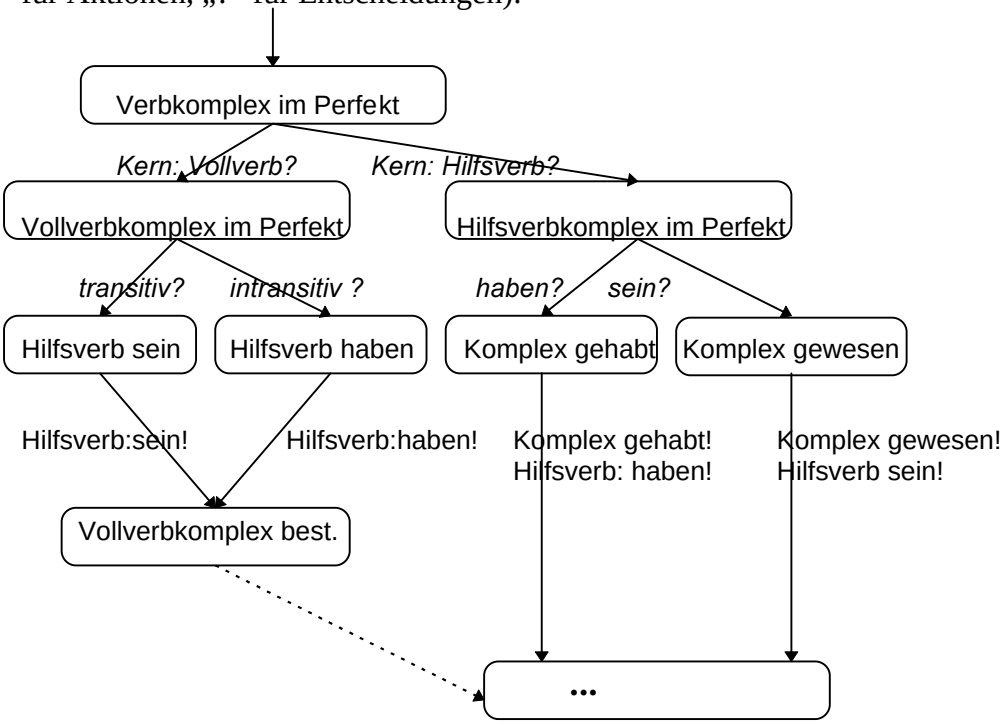
Regel:

Neuer Zielbaum:



Die Menge M^{Q} wird nun um die Relation *Feat/Col* erhöht.

Beispiel Regelnetz entscheidungsnetzbasierter TT „Auswahl des passenden Hilfsverbes“ („!“ stehen für Aktionen, „?“ für Entscheidungen):



Spezielle Treetransducer

-Attributdiffusion

Jeder Knoten des Baumes kann Attribute enthalten, die ggf. verteilt werden müssen. Bei dieser Operation bleibt die Struktur erhalten, lediglich die Attribute verändern sich (*kontextfreier TT*[11]).

f_{TTA}: $T \Rightarrow T'$

Man unterscheidet *vererbte* und *synthetisierte* Attribute.
Vererbte Attribute sind von Knoten, die in Superposition zum betrachteten Knoten liegen (näher an der Wurzel), übernommene Attribute. Vererbung erfolgt meist implizit, d.h. *alle* Attribute werden nach unten zu den Blättern hin übertragen.
Synthetisierte Attribute werden von Knoten, die in Subposition zum betrachteten Knoten liegen (entfernter von der Wurzel) übernommen. Synthese erfolgt meist explizit, d.h. ein Knoten definiert eine Übertragungsregel.

- Weitere Abbildungen sind das *Ordnen von ungeordneten Bäumen* und die *Pronomialisierung*:
Ein ungeordneter Baum wird geordnet, indem die Gültigkeit einer Ordnungsrelation $O(k_s)$ auf alle Subknoten k_{si} eines Knotens k_s hergestellt wird:

f_{TTO}: $T \times O \Rightarrow T'$

[11] Dies ist ein Treetransducer, dessen Regeln im Quellteil nur Einzelknoten anstelle von Teilbäumen haben.

Bei der Pronomialisierung (Referenzbildung) sucht man nach gleichen Strukturen, die durch einen einzelnen Knoten abgekürzt werden können. Damit vermeidet man Mehrfachnennungen langer Phrasen. Die Referenzbildung ist ein Treetransducer, der zunächst nach gleichen ersetzbaren Teilstrukturen in einem geordneten Baum sucht und dann versucht, diese durch neue Knoten abzukürzen bzw. die Teilstrukturen zu reduzieren.[12] Dabei sind bestimmte Regeln zur Ersetzung zu beachten. Es können nur ausgewählte Phrasen bzw. Teilbäume durch Referenzen wie z.B. Pronomen ersetzt werden. Als wichtigste Erweiterung zum normalen Treetransducer gibt es eine Abhängigkeit der Regelanwendung von der Reihenfolge; Im Gegensatz zum Standardtreetransducer wird erst ab dem zweiten Auftreten einundderselben Phrase diese entsprechend der Regel ersetzt.

2.3.1.3 Abbildungen von Sequenzen

- Sequenztransducer

Der Sequenztransducer wird zumeist für die morphologische Komponente benutzt. Er wird durch schrittweise Knotenabbildungen anhand der Morphologieregeln R_M realisiert:

f_{SSM}: $K \times R_M \Rightarrow K'$

Es ist jedoch auch möglich, die morphologische Generierung als *kontextfreien Treetransducer* zu realisieren, bei dem alle Knoten des Baumes in beliebiger Reihenfolge mittels R_M bearbeitet werden (f_{TTM}: $K \times R_M \Rightarrow K'$).

-Morphologische Komponente

Die Morphologische Komponente, d.h. das Lexikon, stellt durch seine enorme Mächtigkeit eine Besonderheit dar. Aus funktionaler Sicht handelt es sich eigentlich „nur“ um einen einfachen Sequenztransducer oder gar Knotentransducer (0-dim). Für professionelle Anwendungen ist es jedoch notwendig, diesen Sequenztransducer in mehrere Teile zu zerlegen, um der großen Wortformenzahl einer natürlichen Sprache (Größenordnung 10^5) Herr zu werden.
Es sind im Deutschen drei grundlegende Flexionsformen zu unterscheiden:

- regelmäßige Flexion,
- regelmäßige Flexion mit vorangehender Stammänderung,
- unregelmäßige Flexion.

Eine mögliche relationale Realisierung [Milo95]:

Relation[13]	Bezeichnung	Beispiele
(W, A)	Wort zu Wortart	(kind, noun)
(W, T)	Wort auf Flexionstyp	(kind, 2)
(W, P, S)	Wort auf Sekundärstamm	(lauf, {prät}lief)
(T, A, P, M)	Flexionsmuster eines Kontextes aus Flexionstyp, Wortart und Paradigma	(2, noun, {pl,1},*er)
(W, P, A, F)	Flexion bei Ausnahmen	(gams, {pl,1},gemsen)
(W,M,W_f)	Flektiertes Wort	(lief,*en,liefen)

Tabelle 3 Relationen bei der morphologischen Generierung im Lexikon

[12] Wenn in der Realisierung in der Phase des Abbauens der Graphenstruktur in eine isomorphe Baumstruktur Information über die ursprüngliche Identität von Knoten z.B. in den Attributen abgelegt wurde, kann jetzt durch einfaches Suchen gleicher Attribute die Pronomialisierung durchgeführt werden.
[13] W .. Wort, A .. Wortart, T .. Flexionstyp, P .. Paradigma, S .. Sekundärstamm

2.3.2 Strukturtypverändernde Funktionen

2.3.2.1 Formale Unterschiede von Strukturen

Bei den strukturtypverändernden Funktionen müssen unabhängig von der konkreten Anwendung formale Unterschiede der Strukturen überwunden werden.

	Graph (n dim)	Baum (2 dim)	Folge (1 dim)
Wurzelknotenzahl	1 .. n	1	1
Vorgänger eines Knotens	0 .. n	0 .. 1	0 .. 1
Nachfolger eines Knotens	0 .. n	0 .. n	0 .. 1

Tabelle 4 Formale Unterschiede zwischen Strukturen

Der Abbau der Dimensionen, der bei der Generierung erfolgt, ist dabei weniger komplex, als ihr Aufbau, der bei der Analyse erfolgt. Beim Abbau findet ein Verlust impliziter[14] Information statt, der entweder gewollt ist oder durch Konvertierung in explizite[15] Information aufgefangen wird. Eine Funktionskette aus strukturtypverändernden Funktionen f_{sv} und strukturtyperhaltenden Funktionen f_{se}: ... f_{sv} x f_{se} ... kann dann transformiert werden in eine Kette aus *formalen* strukturtypverändernden Funktionen $f_{sv'}$ und *anwendungsspezifischen* strukturtyperhaltenden Funktionen f_{se}: ... $f_{sv'}$ x f_{se} ...
Damit kann man diese formalen Funktionen $f_{sv'}$ allgemein definieren. Die Anwendung muß nun nur noch die strukturtypverändernden Funktionen f_{se} ggf. modifizieren.

2.3.2.2 Abbildung eines (minimalen) Graphen in seinen isomorphen Baum

f_{GT}: G $\Rightarrow$ T'

Diese Abbildung muß den formalen Unterschied zwischen Bäumen und minimalen Graphen überwinden: Vorgänger eines Knotens bei Graphen: 0 .. n, bei Bäumen: 0 .. 1

Diese Funktion kann technisch am einfachsten durch ein *Preorder-Graphendurchlauf*[16] realisiert werden, bei dem der isomorphe Baum entsteht.

2.3.2.3 Abbildung eines geordneten Baumes in eine Morphemsequenz

Diese Funktion erzeugt aus dem Baum die isomorphe Sequenz des Baumes, den Text.

f_{TS}: T $\Rightarrow$ S

Diese Funktion kann technisch durch einen *Preorder-Baumdurchlauf* realisiert werden, bei dem die isomorphe Sequenz entsteht. Dazu ist zum Einordnen des übergeordneten Knotens in die Reihenfolge seiner Unterknoten ein Positionsattribut wie beim Ordnen von Bäumen notwendig.

[14] Implizite Information ist die durch die Struktur gespeicherte Information.
[15] Explizite Information sind z.B. Features, d.h. Attribut-Wert-Paare.
[16] Dabei wird der Graph rekursiv durchquert. Dabei wird jeder Schritt und jedes Backtracking aufgezeichnet. Das Resultat ist der isomorphe Baum. Teilgraphen mit mehr als einem Vorgänger werden entsprechend oft kopiert.

2.3.3 Komplexe Funktionen

2.3.3.1 Binäre Funktion: Komposition von Bäumen

Die Komposition von Bäumen wird notwendig, wenn mehrere kleine Sätze zu einem großen zusammengefügt werden sollen. (z.B. Nebensatzeinfügung). Funktional handelt es sich um die Verbindung von zwei Bäumen. Diese kann unterschiedliche Komplexität aufweisen.

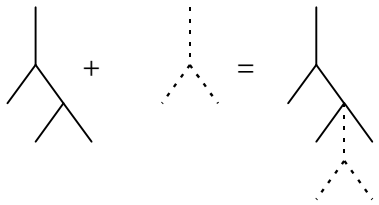
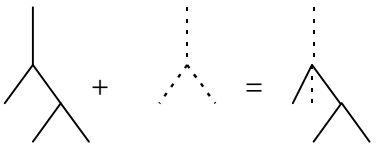
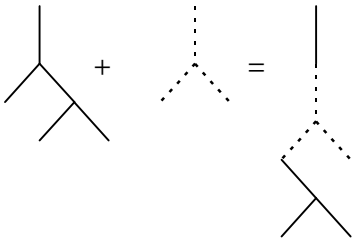
Konkatenation	Unifikation	Insertion/ Adjunktion TAG[17]
einfache Verbindung zweier Bäume, indem ein Baum mit der Wurzel an einen Knoten des anderen Baumes geknüpft wird	Vereinigung zweier Strukturen anhand Gleichheit von Knoten, Kanten und Attributen	ein Baum wird in einen zu spaltenden Knoten des anderen Baumes eingefügt
		

Tabelle 5 Verbindungsarten von Bäumen

Dieser Prozeß kann durch die folgende Funktion beschrieben werden:

f_{TTK}^{RK}: $T_1 \times T_2 \Rightarrow T'$

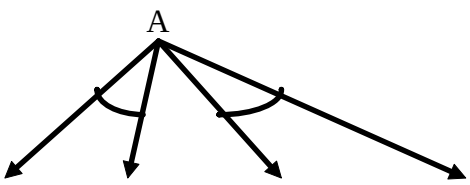
R_K steht für die Art bzw. die Regeln der Komposition.

2.3.3.2 Entscheidungsnetze

[DUD S.212 ff.]

Entscheidungsnetze sind Regelwerke, die beliebige Inputstrukturen hinsichtlich Anzahl und Komplexität verarbeiten können. Bei Treetransducern wurde ihre Nutzung bereits gezeigt.

Ein Entscheidungsnetz ist ein geordneter Graph. Jeder Knoten stellt ein atomares Entscheidungsproblem dar, wird daher *Entscheidungsknoten* genannt. Die Kanten zwischen einem Knoten und seinen Söhnen (Subknoten) stellen Entscheidungsalternativen dar. Die Auswahl eines Weges von der Wurzel zu einem Blatt stellt eine *Entscheidung* dar. Jede Kante kann mit auszuführenden Aktionen beschriftet sein. Die Entscheidung ist also eine Anweisungsfolge. Entscheidungsnetze können auch in Form von UND/ODER Graphen notiert sein:



[17] TAG steht für Tree Adjoning Grammar und stellt ein Verfahren zur Beschreibung von Zeichenfolgen durch zu verbindende Einzelbäume dar. Der Verbindungsprozeß läuft durch wiederholtes Einfügen von -Bäumen ab, bis die zu realisierende Zeichfolge abgebildet ist. Siehe auch [Har88]

B C D E

„Entweder wird mit B und C oder mit D und E fortgesetzt“

Entscheidungsnetze stellen im Gegensatz zu den anderen bisher betrachteten Regelformalismen die Abbildung als solche in den Vordergrund, unabhängig vom Typ der betrachteten Strukturen. Die Struktur muß weder nach einer bestimmten Vorschrift passiert werden, noch muß irgendwelchen Bedingungen genügen. Das kann als Vor- oder Nachteil gesehen werden: Vorteilig ist die strukturelle Unabhängigkeit. Es gibt aber keine vom Verfahren garantierte Vollständigkeit des Abbildungsprozesses.

2.3.4 Abbildung eines gesamten Generierungsprozesses

Die Generierung ist, nachdem Abbildungen und Strukturen formal definiert worden sind, die Verkettung dieser Strukturen und Abbildungen nach folgendem Prinzip:

$Abb_n(... Abb_{i+1}(Abb_i(In)))$ mit Typ out(Abb_i) = Typ in(Abb_{i+1})

d.h. die Strukturtypen zweier verketter Abbildungen müssen übereinstimmen.

Als Beispiel soll ein inkrementelles Modell zur Generierung vorgestellt werden.

Bei der Betrachtung der *Realisierung* wurde festgestellt, daß oft drei Zwischenschritte notwendig sind, um aus dem semantisches Netz eine sprachabhängige Baumstruktur zu erzeugen:

1. Erzeugen minimaler Graphen f_{GGM}
2. Abbildung der Graphen in die isomorphen Bäume f_{GT}
3. Treetransducing; Erzeugung der sprachabhängigen Strukturen (Dependenzbäume) f_{TT}

Zusammenfassend kann man die *Realisierung* als eine Funktion folgender Struktur auffassen:

$$(f_{GGM} \times \bigcup_{i=0}^{n} (sel(i) \times f_{GT} \times f_{TT})) (G) = \underline{\mathbf{T}}$$

Die *morphologische Generierung* soll in folgenden Schritten durchgeführt werden:

1. Vererben von Attributen f_{TTA}
2. Ordnen von ungeordneten Bäumen f_{TTO}
3. Pronomialisierung (Referenzbildung) f_{TTP}
4. Erzeugen eines Textes (isomorphe Sequenz des Baumes) f_{TS}
5. Morphologische Generierung f_{SSM}

Zusammenfassend kann man die *morphologische Generierung* als eine Funktion folgender Struktur auffassen:

$(f_{TTA} \times f_{TTO} \times f_{TTP} \times f_{TS} \times f_{SSM})$ $(\underline{\mathbf{T}}) = \underline{\mathbf{S}}$ [18]

Der gesamte Generierungsprozeß lautet dann:

[18] $\underline{\mathbf{X}}$ sind Mengen von Strukturen.

$$(f_{GGM} \times \bigcup_{i=0}^{n} (sel(i) \times f_{GT} \times f_{TT} \times f_{TTA} \times f_{TTO} \times f_{TTP} \times f_{TS} \times f_{SSM})) (G) = \underline{\mathbf{S}}$$

Dabei werden die Funktionen f_{TT} und f_{SSM} auf der Grundlage von komplexen Regelwerken definiert. Aus dem Inputnetz entsteht eine Menge von Outputsätzen.

3 Prototyp eines Generierungswerkzeugs

Aus den allg. Strukturen und Abbildungen wurde ein Prototyp in C realisiert, der Funktionen und Datenstrukturen zum Prozessieren dieser Abbildungen bereithält.
Durch die starke Verallgemeinerung der Verfahren zur Generierung ist es möglich, mit einem sehr kleinen Satz unterschiedlicher Datenobjekte auszukommen:

- Strukturen
- Entitäten mit Attributen
- Relationen mit Attributen

Alle Strukturen wie: Graphen, Bäume, Folgen, Regeln etc. werden mit diesen Objekten abgebildet.
In C werden diese Strukturen durch Tabellen (streng relational) realisiert:

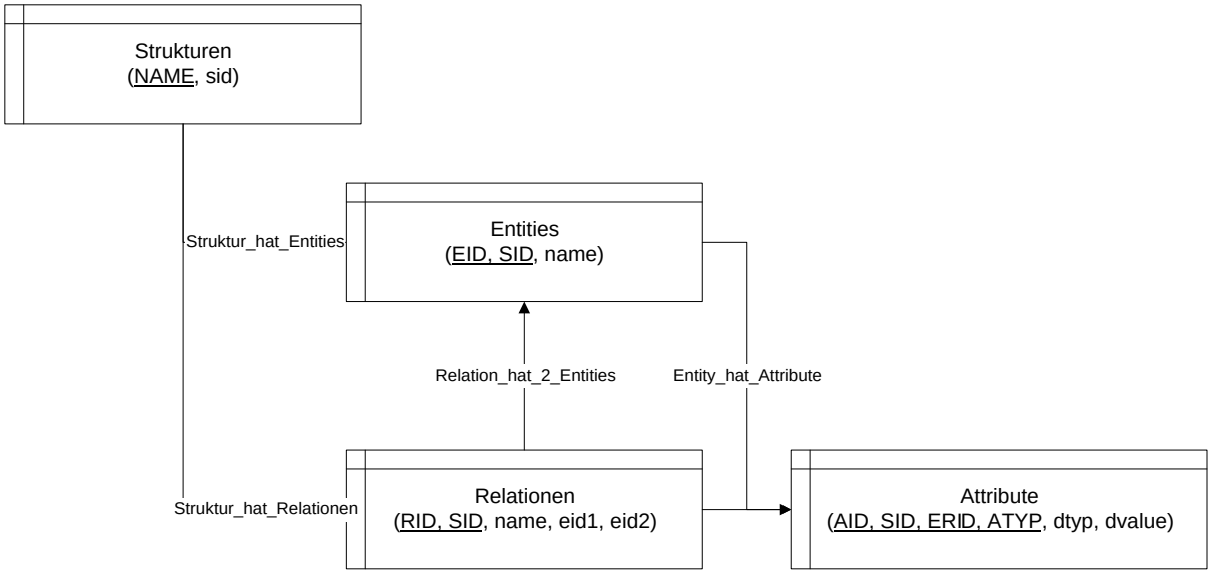


Abbildung 6 ERM der benötigten Datenstrukturen

Über diesen Objekten sind Funktionen definiert:

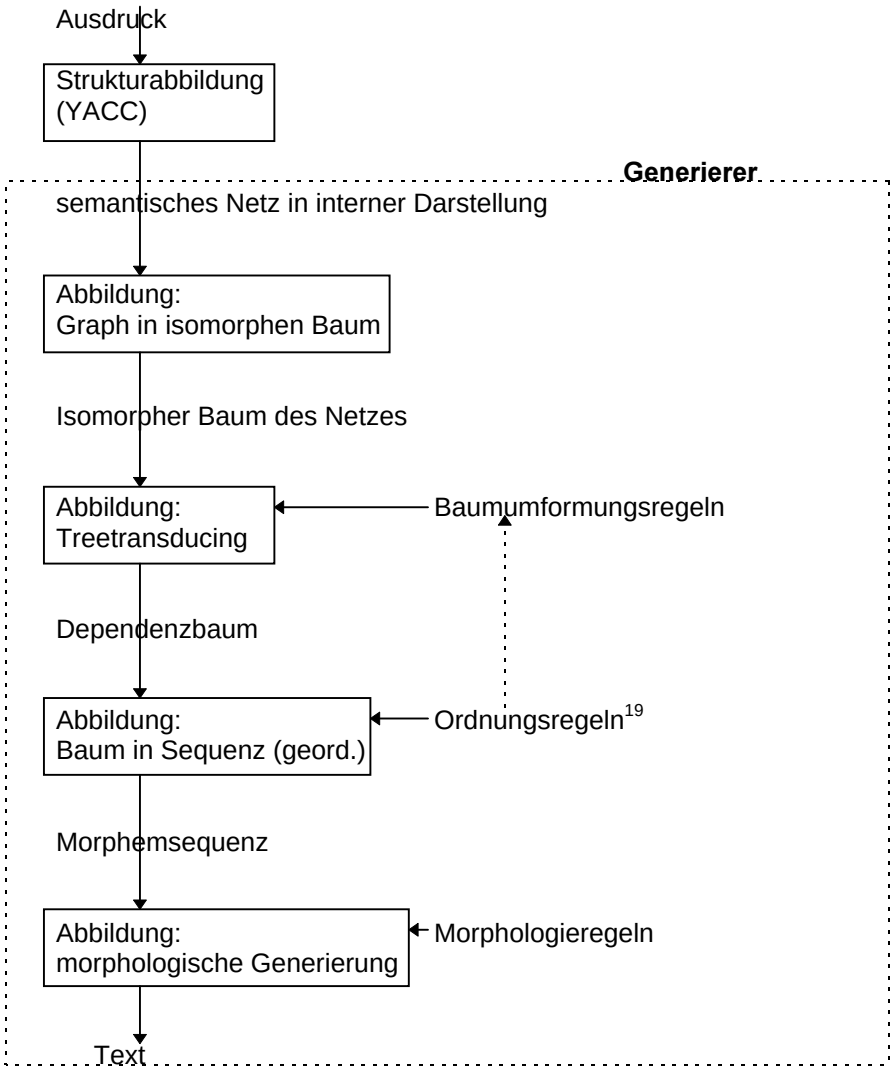
- Funktionen zur einfachen Strukturmanipulation (Anzeigen, Anlegen, Ändern, Löschen) jeweils für Strukturen, Entitäten, Relationen und Attribute, d.h. für alle Tabellen
- Strukturtypverändernde Funktionen: Graph zu Baum, Baum zu Folge usw.
- Strukturtyperhaltende Funktionen: Graph1 zu Graph2 , Baum1 zu Baum2 usw.
- Sonstige Funktionen

Die komplexen Funktionen greifen über eine Datenverwaltungsschicht auf die Strukturen zu.
Die Reihenfolge der Funktionsaufrufe bzw. die Abarbeitungsweise ist dem Benutzer überlassen. Dieser muß die Funktionen anhand seines Verfahrens zu einem sinnvollen Ablauf kombinieren. Dadurch kann der Generierer als virtuelle Maschine angesehen werden.

4 Beispiel: Generierung von Texten aus prädikatenlogischen Ausdrücken (inkrementeller Algorithmus)

4.1 Ablauf

In diesem Kapitel soll anhand eines Beispiels der Ansatz der allg. Strukturen und Abbildungen mit getestet werden.
Zunächst muß der prädikatenlogischer Ausdruck in eine Graphendarstellung gebracht werden.
Dann können die Schritte der Realisierung und die der morphologischen Generierung erfolgen. Die Nutzung von Regelwerken ist in der folgenden Ablauf-Skizze angegeben:



Formal kann dieser Ablauf[20] wie folgt beschrieben werden:

[19] Die Ordnungsregeln werden beim Treetransducing in den Dependenzbaum implizit über Positionsattribute eingebettet.

[20] P .. prädikatenlogischer Ausdruck, G .. semantisches Netz (DAG), T_s .. Isomorpher Baum des semantischen Netzes S, T_d .. Dependenzbaum, S_m .. Morphemsequenz, S_t .. Ausgabetext, R_{spez} ..YACC-Spezifikation, R_t ..Transducing-Regelwerk

$G = f_{YACC}(P,R_{spez})$	Semantische Analyse mit yacc
$T_S = f_{GT}(G)$	isomorpher Baum zum Graphen
$T_d = f_{TT}(T_S,R_t)$	Dependenzbaum aus semantischem Baum, Realisierung
$S_M = f_{TS}(T_D)$	Morphemsequenz aus Dependenzbaum, morphologische Generierung (I)
$S_T = f_{SSmor}(S_M)$	Text aus Morphemsequenz, morphologische Generierung (II)

$f_{GENER} = f_{YACC} \times f_{GT} \times f_{TT} \times f_{TS} \times f_{SSmor}$

Sämtliche Funktionen arbeiten inkrementell ohne Backtracking und mit einelementigen Lösungsmengen.
Es werden prädikatenlogische Ausdrücke betrachet, die der folgenden Syntax in BNF genügen:

PA =:: QA '.' AA
QA =:: '∃' VA | '∀' VA
AA =:: (PA) | (FA) | (AA AOP AA)
FA =:: FC '(' FL ')'
FL =:: FE | FE ',' FL
FE =:: VA | AA

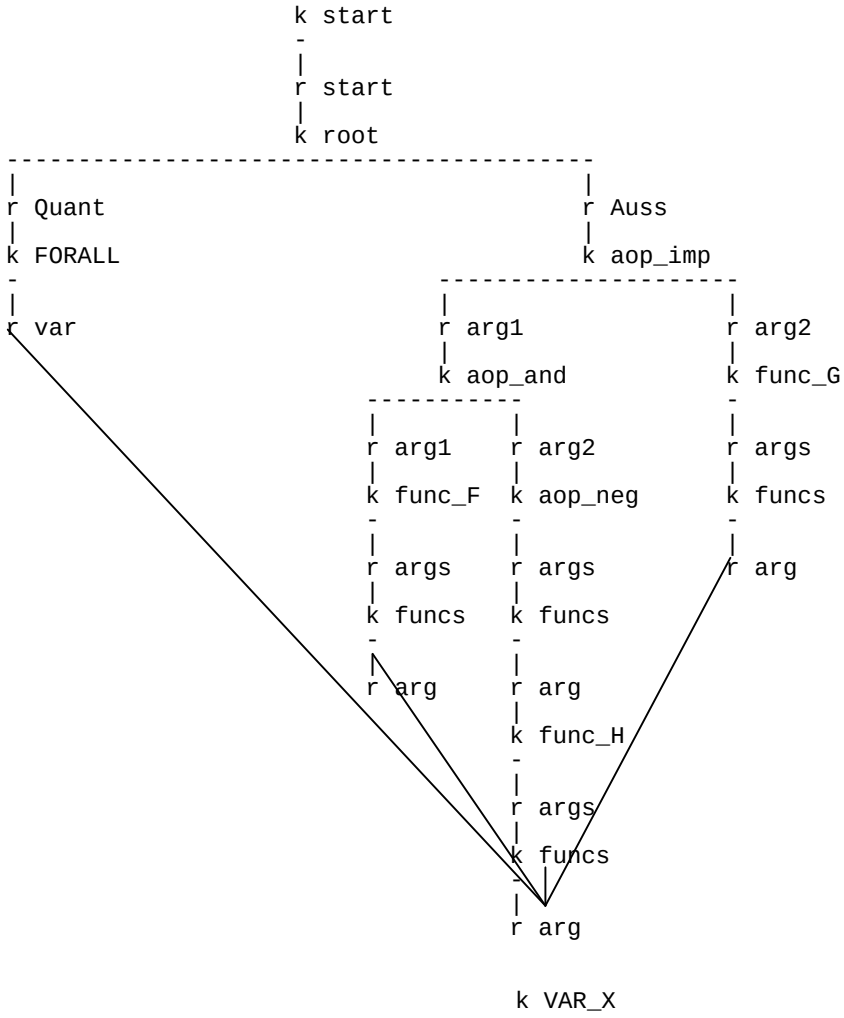
Abkürzung	Langtext	Beispiel
PA	Prädikatenlogischer Ausdruck	∃X.(∀Y.((vogel(X) ∧ fliegen(Y))) → ¬pinguin(X))
QA	Quantorenausdruck	∃X
AA	Aussagenlogischer Ausdruck	vogel(X) ∧ fliegen(Y)
VA	Variable (terminales Symbol)	X
FA	Funktionsausdruck	vogel(X)
FC	Funktor (terminales Symbol)	vogel
FL	Funktionsargumentliste	X,Y
FE	Funktionsargument	X
AOP	aussagenlogischer Operator (terminales Symbol)	∧
VA	Variable (terminales Symbol)	X, Y
FC	Funktor (terminales Symbol)	vogel, pinguin, heissen, fliegen
AOP	aussagenlogischer Operator (terminales Symbol)	und, oder, folgt, nicht

Tabelle 6 Beschreibung der Syntaxsymbole

Als Beispiele sollen ein prädikatenlogischer Ausdruck, dessen semantisches Netz, der isomorphe Baum, der daraus erzeugte Dependenzbaum und der dann generierte Text dargestellt werden.

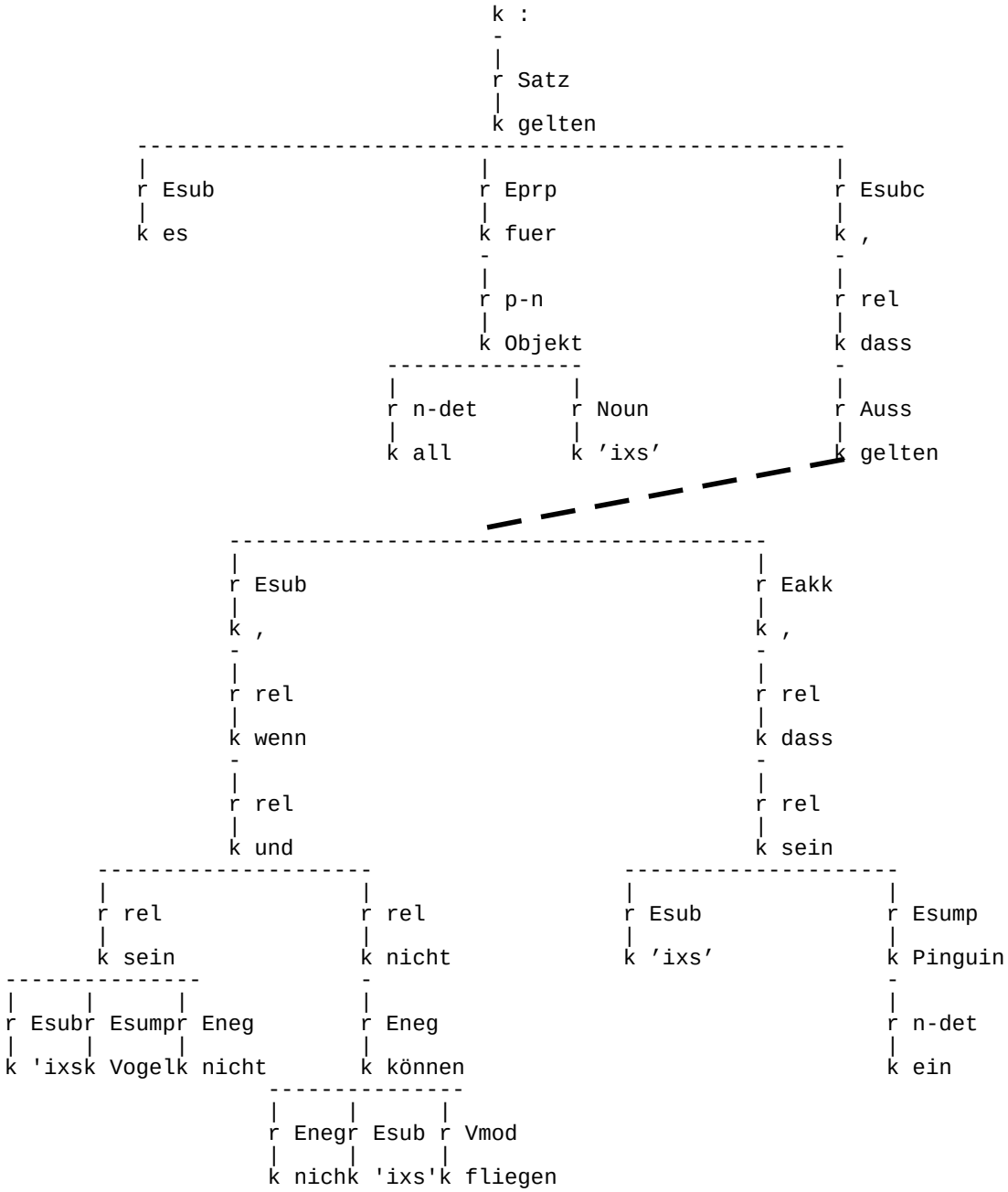
1. prädikatenlogischer Ausdruck: ∀X.(vogel(X) ∧¬fliegen(X))→pinguin(X)
Dieser Ausdruck wird mit einem mittels YACC erzeugten Parser analysiert und dadurch eine semantische Struktur erzeugt.

2. Semantische Struktur:



Die Struktur wird in ihren isomorpher Baum abgebildet, der in den Dependenzbaum in deutscher Sprache überführt wird. Dazu werden der allgemein in Kap. 2.3.1 definierte Treetransducer benötigt, zu dem ein konkretes Regelwerk definiert wurde.

3. Dependenzbaum



Anhand von Positionsattributen erfolgt nun in einem Schritt die Sortierung des Baumes und die Abbildung in die entsprechende Folge. Nach der Diffusion der Attribute (Vererbung und Synthese) stehen alle Positionsattribute zur Verfügung (Beispielteilbaum):

```
                    k sein headpos=10
                      |
-----------------------------------------
|                                       |
r Esub position=1                       r Esump position=8
|                                       |
k 'ixs' headpos=5                       k Pinguin headpos=2
                                        -
                                        |
                                        r n-det position=1
                                        |
                                        k ein headpos=2
```

Obige Attribute führen bei rekursiver Anwendung zu: `((‘ixs’)((ein) pinguin)sein.`
`Position` gibt an, welche Position der zugehörige Subbaum bzgl. der anderen Subbäume eines bestimmten Knotens hat. `Headpos` plaziert den Superknoten dieser Bäume in diese Reihe.
Diese Sequenz wird dann elementweise morphologisch generiert und ergibt dann den Ausgabetext:

4. Ausgabetext

```
es gilt fuer alle Objekte 'ixs' , dass , wenn 'ixs' Vogel ist und
'ixs' nicht fliegen kann gilt , dass 'ixs' ein Pinguin ist
```

weiteres Beispiel:

- Ausdruck 1: $\forall X.(vogel(X) \rightarrow heissen(X,Y))$ `->` „es gilt fuer alle Objekte 'ixs' , dass , wenn 'ixs' ein Vogel ist gilt , dass 'ixs' 'ypsi' heisst“

- Ausdruck 2: $\exists X.(vogel(X) \wedge pinguin(X))$ `->` „es existiert ein objekt 'ixs', so dass gilt : 'ixs' ist ein Vogel und 'ixs' ist ein Pinguin“

- Ausdruck 3: $\exists X.((vogel(X) \wedge pinguin(X)) \rightarrow heissen(X,Y))$ `->` „es existiert ein objekt 'ixs' , so dass gilt : wenn 'ixs' ein Vogel ist und 'ixs' ein Pinguin ist gilt , dass 'ixs' 'ypsi' heisst“

Der dritte Ausdruck ist das resolvierte Ergebnis der beiden ersten. (Bei einem geeigneten Parser und einem Resolvierer könnte man also nun Wissen eines Textes ableiten und kompakt wieder natürlichsprachlich ausgeben).

4.2 Beispiele von Regelstrukturen

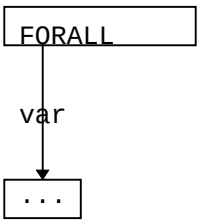
- Semantische Analyse

Das semantische Netz wird in Form von Anweisungen dargestellt, die der Generator direkt verarbeiten kann. Diese Anweisungen werden beim Parsen des Inputs (YACC) erzeugt.

Beispiel: YACC-Regel für Quantoren (in der Eingabe als EX oder AL dargestellt):

```
qa:             EX va             {$$=node("EX");
                                    rela("var",$$,$2);}   |
                AL va             {$$=node("FORALL");
                                    rela("var",$$,$2);}
 ;
```

Diese YACC-Regel sagt aus, daß ein Quantorausdruck mit einem All- oder Existenzquantor beginnt, gefolgt von einer Variable. Zu dieser Regel wird folgender Teilbaum bestehend aus einem Knoten und einer Relation konstruiert (durch die Anweisungen in den geschweiften Klammern):

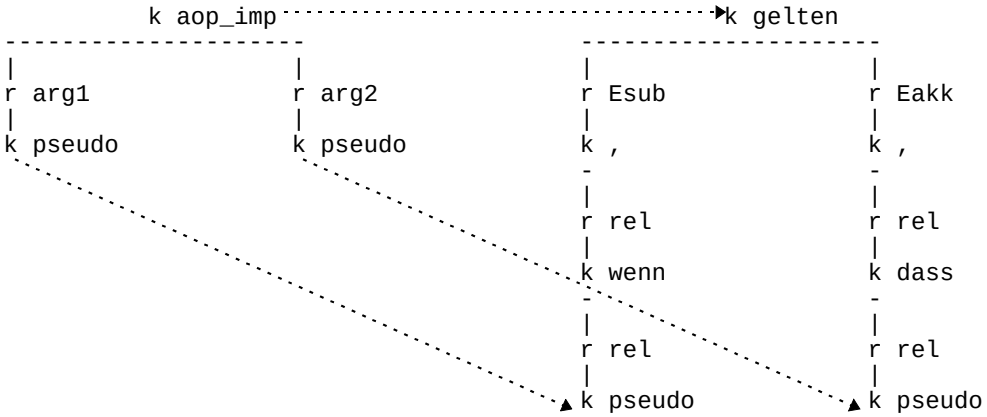


Durch die Anwendung aller der durch die semantische Analyse entstandenen Anweisungsfolgen entsteht dann der gesamte Graph.

- Treetransducer

Beim Treetransducing werden aus bestimmten Teilbäumen (linke Regelseite) neue Teilbäume (rechte Regelseite) konstruiert. Dabei wird jedem Knoten im Inputbaum genau ein Knoten im Zielbaum zugeordnet. Nur dadurch ist es möglich, daß die Unterbäume eines Knotens nicht zur gleichen Zeit, sondern nacheinander (d.h. mit verschiedenen Regeln) eindeutig bearbeitet werden können.

Als Beispiel soll die Abbildung des zum Implikationsoperator gehörenden Quellbaumes in seinen sprachabhängigen Zielbaum betrachtet werden.



5 Zusammenfassung

Es existieren viele Verfahren und Systeme zur Generierung. Bisher sind jedoch wenig Untersuchungen gemacht worden, diese Verfahren allgemein und vergleichend in einer strengen Sicht der Informationsverarbeitung (Verallgemeinerung der Datenstrukturen, Funktionen, Algorithmen) zu beschreiben. Mit Hilfe dieser allg. Betrachtungsweise kann man jedoch viele Verfahren formaler beschreiben, klassifizieren und versuchen, das Wesentliche des Verfahrens zu erkennen.

Dadurch ist es möglich, eine allgemeine Generierungsmaschine zu definieren, die in Lage ist, mehrere Generierungsverfahren, deren Regelwerke und Algorithmen, zu prozessieren. Man erhält dann ein Werkzeug, bei dem Änderbarkeit und Austauschbarkeit hinsichtlich Daten- und Steuerfluß durch die Trennung von Daten, Formalismus und Verarbeitung möglich werden.

Dazu müssen über eine funktionale Sicht und mit Hilfe der Graphentheorie die unterschiedlichen Strukturen und die Abbildungen zwischen diesen identifiziert werden, um dann eine Menge (abstrakter) Strukturen und Abbildungen zu erhalten.
Durch die Klassifizierung der Strukturen anhand ihrer Komplexität erreicht man Klassen[21] von Strukturen und Klassen von Abbildungen zwischen diesen Strukturen. Unter Komplexität wird die Art der Vernetzung verstanden (Sequenz, Baum, Graph).
Ein konkretes Verfahren ist dann die Verkettung von bestimmten Strukturen durch Abbildungen und die Definition eine Abarbeitungsvorschrift über dieser Kette.
Die Strukturen und den Abbildungen sind eine virtuelle Maschine (Generierer). Dieser ist in der Lage, viele Generierungsverfahren prozessieren zu können.

Wenn man den Abstraktionsgrad als Maß nimmt, kann man drei Ebenen[22] von (computerlinguistischen) Anwendungen definieren:

1. anwendungsorientierte Ebene (Computerlinguistisches Verfahren in der Praxis) z.B. *Parser auf LFG-Basis* (konkret realisierte Strukturen und Funktionen für ein spezielles Verfahren in einem speziellen Umfeld)
2. abstrakte anwendungsorientierte Ebene (Theoretische Informatik wird benutzt zur allg. Beschreibung computerlinguistischer Prozesse) z.B. *Attributdiffusion in einem Dependenzbaum*
3. theoretisch-abstrakte Ebene (Theoretische Informatik, Linguistik) z.B. *Kellerautomat, Turing Maschine* (theoretisch definierte Strukturen und allg. Algorithmen (u.U. nicht vollständig und verfahrensunabhängig))

Diese Arbeit beschäftigt sich mit der Definition der zweiten Ebene. Diese ist weitgehend unabhängig von einem konkretem System/Verfahren, stellt die Verbindung zwischen konkretem Verfahren und abstrakter theoretischer Informatik/Linguistik dar.

Heute sind in der Computerlinguistik oftmals Probleme vorhanden bei essentiell notwendigen Eigenschaften jeglicher Software: Wiederbenutzbarkeit, Modularität, Änderbarkeit, Kompatibilität.

Obwohl die Verfahren oft nur leicht verändert sind, sind diese Eigenschaften durch die zu geringe Verallgemeinerung bei zu hoher Anwendungskomplexität oft nicht gegeben.[23] Dadurch tritt ein unnötiger Verlust von Wissen und Zeit auf. Diese Probleme können durch den Ansatz der Einführung einer abstrakten anwendungsorientierten (Zwischen-) Ebene beim Design überwunden werden.
Weitere Verallgemeinerung ist über die Generierung hinaus möglich.

[21] Der „allgemeinste Punkt“ der abstrakten Strukturen/ Abbildungen ist der Graph als Struktur und der n-dimensionale Graphentransducer als Funktion. Diese beiden Elemente sind die beiden einzigen theoretisch unbedingt notwendigen Objekte der abstrakten Maschine.

[22] Mit dem funktionaler Ansatz ist eine Ebene die Gesamtheit von Strukturen (Daten) und Abbildungen (Funktionen) = eine abstrakte Maschine.

[23] „Nach 10 Jahren muß man alles wegwerfen und neu programmieren“

6 Quellenverzeichnis

[Bat90] John A. Bateman, Eduard H. Hovy: An Overview of Computational Text Generation,Infom. Science Ins., Univ. of South. California Marina del Rey 1990;; The KOMET multilingual text generation system: http://www.darmstadt.gmd.de/publish/komet/komet-system.html

[Dud89] Autorenkollektiv: Duden Informatik, Dudenverlag Mannheim, Wien, Zürich 1989

[Eng94] Ulrich Engel Syntax der deutschen Gegenwartssprache, Erich Schmidt Verlag 1994

[EnzMa86] Kleine Enzyklopädie Mathematik, Bibliografisches Institut Leipzig 1986, 13. Auflage S 361 ff.

[Freu88] Rudolf Freundlich Einführung in die Semantik, Wissenschaftliche Buchgesellschaft, Darmstadt, 2. Aufl. 1988

[Hall81] M.A.K. Halliday Options and Functions in the English Clause, in: Halliday and Martin: Readings in systemic linguistics, Batsford Academic Press, pp. 138-145, 1981

[Har88] Karin Harbusch Effiziente Analyse natürlicher Sprache mit TAG's, in: I.S. Bátori: Computerlinguistik und ihre theoretischen Grundlagen, Informatik Fachberichte 195, Springerverlag Heidelberg 1988

[Jabl90] Konrad Jablonski, Armin Rau, Johannes Ritzke: Wissensbasierte Textgenerierung, Tübinger Beiträge zur Linguistik Nr. 338, Gunter Narr Verlag 1990

[Jack93] Ray Jackendoff X-Bar-Semantics, in: Semantics and the Lexikon, James Pustejovsky, Kluwer Academic Publishers 1993, S. 15-27

[Lan1987] Landsbergen Montague Grammar & Machine Translation: in Whitelock et al.: Linguistic Theory and Computer Applications, Academic Press 1987

[McDon85] David D. McDonald, James D. Pustejovsky: Description-Directed Natural Language Generation, in: Proceedings of the 9th IJCAI, S. 799-805, IJCAI 1985.

[McDon91] David D. McDonald: Lexical Choice, in [Par91]

[McKeo85] Kathleen McKeown: Text generation, in: Studies in Natural Language Processing, Cambridge University Press 1985, S. 133 ff.

[McKeo91] Kathleen McKeown: A contrastive Evaluation of FUG for Surface Language Generation, in: [Par91] S. 351 ff.

[Milo95] Marija Milosevic: Lexikonstrukturen für die maschinelle Übersetzung natürlicher Sprachen, Diplomarbeit, Diplomarbeit, TU Dresden 1995

[Mont80] D. Kalish, Richard Montague, Gary Mar: LOGIC, Techniques of Formal Reasoning, Harcourt Brace Jovanovich Publ., 2.Aufl. New York 1980

[Nir87] Sergei Nirenburg, Victor Raskin, Allen B. Tucker: The Structure of Interlingua in TRANSLATOR, in: Sergei Nirenburg: Machine Translation, Cambridge University Press 1987

[Par91] Cécile Paris, William R. Swartout, William C. Mann: Natural Language Generation in Artificial Intelligence and Computer Linguistics, Kluwer Academic Publishers 1991

[Pangl] Penman: English Sentence and Phrase generation (PANGLOSS Translation System): http://crl.nmsu.edu:81/users/pangloss/p...node27.html

[Per80] F.C.N. Pereira u. D.H.D. Warren: Definite Clause Grammars for Language Analysis - A Survey of the Formalism and a Comparison with ATN's, in: Artificial Intelligence 13(1980) S. 231-278

[Reit91] Norbert Reithinger: POPEL: A Parallel and Incremental Language Generation System, in: [Par91] S. 179 ff.

[Rös86] Dietmar Rösner: Ein System zur Generierung von deutschen Texten aus semantischen Repräsentationen, Dissertation, Universität Stuttgart 1986

[Ros95] Lutz Rosenpflanzer: Konzeption eines Syntaxanalysators am Beispiel der deutschen Sprache, Diplomarbeit, Technische Universität Dresden 1995

[Sow81] John F. Sowa Generating Language from Conceptual Graphs in: Comp. & Maths. Vol. 9 No. 1 S. 29-43 , 1983

[Sow93] John F. Sowa Lexical and Conceptual Structures, in: Semantics and the Lexikon, James Pustejovsky, Kluwer Academic Publishers 1993

[Sme91] Koenraad de Smedt u. Gerard Kempen: Segment Grammar: a Formalism of Incrementell Sentence Generation, in: [Par91] S. 329 ff.

[Wed88] Jürgen Wedekind: Generation as Structure Driven Derivation, in: Proceedings of the 12th international Conference on Computational Linguistics, Budapest Aug. 1988